Politechnika Świętokrzyska
Wydział Elektrotechniki, Automatyki i Informatyki

Projekt
Technologie obiektowe

Grupa:	Autorzy:	Data:
1ID21A	Dawid Kosmala Agnieszka Rogowska	20.06.2020

Temat Projektu:

Relacyjno - obiektowe bazy danych

1. Cel projektu

Celem projektu było omówienie elementów występujących w bazie danych związanych z obiektowością oraz opracowanie dla wybranego problemu rozwiązania wykorzystującego wybraną relacyjno-obiektową bazę danych oraz relacyjną bazę danych.

2. Wykorzystane środowiska i narzędzia

W projekcie wykorzystano:

- Oracle Database 11g
- Język PHP
- Bootstrap
- HTML
- CSS
- Visual Studio Code

Język PHP został wybrany z uwagi na jego elastyczność która pozwala również na projektowanie aplikacji w sposób obiektowy jak i zarówno na jego ogromną popularność wśród twórców serwisów internetowych. PHP pozwala również na proste łączenie się z bazami danych dostarczanych przez wielu producentów. Baza danych Oracle Database 11g została wybrana przez jej wsparcie zarówno dla relacyjnych jak i relacyjno-obiektowych baz danych. Jest ona również mocno wspierana przez twórców języka PHP więc wszystkie narzędzia do pracy z nią są bardzo łatwo dostępne. Do stworzenia interfejsu graficznego został użyty język HTML wraz z kaskadowymi arkuszami stylów CSS. Do łatwiejszej konfiguracji interfejsu został wykorzystany również Bootstrap który znacząco ułatwia tworzenie takich struktur jak formularze. Do edycji kodu został użyty program Visual Studio Code który wspiera tworzenie aplikacji internetowych z użyciem wybranych narzędzi.

3. Charakterystyka obiektowo-relacyjnej bazy danych

```
CREATE OR REPLACE TYPE FiguraGeometryczna
AS OBJECT(
  id Number(7),
  Pole NUMBER(7),
  Obwod NUMBER(7),
  MEMBER PROCEDURE obliczObwod,
  MEMBER PROCEDURE obliczPole,
  MEMBER FUNCTION getObwod RETURN NUMBER,
  MEMBER FUNCTION getPole RETURN NUMBER)NOT FINAL;
```

rys. 1 Tworzenie typów

Na rys. 1 przedstawiono sposób tworzenia typów obiektowego wewnątrz bazy Oracle. Typy wewnątrz bazy danych tworzymy w podobny sposób jak w większości obiektowych języków programowania. Główną różnicą jest to, że interfejs typu oraz jego implementację należy specyfikować oddzielnie. Jak widać na rys. 1 na końcu deklaracji interfejsu została użyta klauzula *NOT FINAL* która umożliwia dziedziczenie, domyślnie możliwość dziedziczenia w bazie danych Oracle jest wyłączona więc aby korzystać z tej opcji należy pamiętać o użyciu tej klauzuli.

```
CREATE TYPE BODY FiguraGeometryczna AS
  MEMBER FUNCTION getObwod RETURN NUMBER IS
    BEGIN
      RETURN(Obwod);
    END getObwod;
  MEMBER FUNCTION getPole RETURN NUMBER IS
    BEGIN
      RETURN(Pole);
    END getPole;
  MEMBER PROCEDURE obliczObwod IS
    BEGIN
     Obwod:=1;
    END obliczObwod;
  MEMBER PROCEDURE obliczPole IS
    BEGIN
     Pole:=1;
    END obliczPole;
END;
```

rys. 2 Implementacja metod

Na **Błąd! Nie można odnaleźć źródła odwołania.** przedstawiono implementację metod zadeklarowanych wcześniej w interfejsie typu. Aby określić zachowanie metod należy zastosować klauzule *BEGIN* oraz *END.* Zachowanie danej metody należy umieścić pomiędzy tymi dwoma klauzulami.

```
CREATE OR REPLACE TYPE KoloTyp UNDER FiguraGeometryczna(
    Promien NUMBER(7),
    Srednica NUMBER(7),
    OVERRIDING MEMBER PROCEDURE obliczObwod,
    OVERRIDING MEMBER PROCEDURE obliczPole);
```

rys. 2 Typ *KoloTyp*

Na rys. 3 przedstawiono typ *KoloTyp*, który dziedziczy po typie *FiguraGeometryczna* używając słowa kluczowego *UNDER*. Typ *KoloTyp* posiada wszystkie pola zawarte w typie *FiguraGeometryczna* jak również dodaje swoje własne pola. W deklaracji typu zostało również określone przysłanianie metod zawartych w nadtypie za pomocą słowa kluczowego *OVERRIDING*.

```
CREATE TYPE BODY KoloTyp AS
    OVERRIDING MEMBER PROCEDURE obliczPole IS
        BEGIN
            Pole := 3.14*Promien*Promien;
        END obliczPole;
    OVERRIDING MEMBER PROCEDURE obliczObwod IS
        BEGIN
            Obwod :=  3.14*Promien*2;
        END obliczObwod;
END;
```

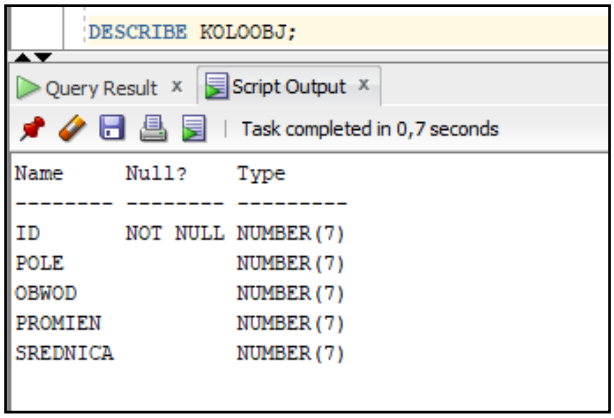
rys. 3 Implementacja metod

Na rys. 4 przedstawiono implementacje metod przesłoniętych w specyfikacji typu *KoloTyp.* Metody zostały dostosowane do tego jak powinny być obliczane pole i obwód w kole.

```
CREATE TABLE KwadratObj OF KwadratTyp
(id PRIMARY KEY);
```

rys. 4 Tworzenie tabeli

Na rys. 5 przedstawiono natomiast sposób tworzenia tabeli przy użyciu utworzonego wcześniej typu. rys. 5 przedstawia tabele która została utworzona za pomocą polecenia na rys. 5. Jak widać tabela poza polami określonymi w typie *KoloTyp* posiada również pola która zostały sprecyzowane w typie *FiguraGeometryczna* po którym on dziedziczy.



rys. 5 Tabela KoloTyp

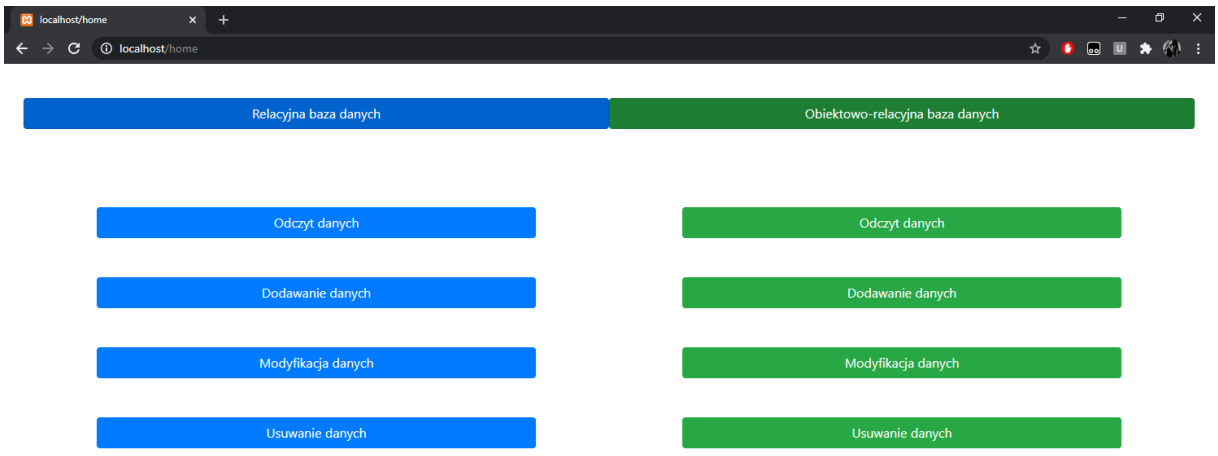
Reszta potrzebnych typów oraz tabel dla tych typów została wykonana w sposób analogiczny do tego, jak zostało to wykonane dla typu *KoloTyp.*

4. Ogólny opis serwisu

4.1. Frontend serwisu

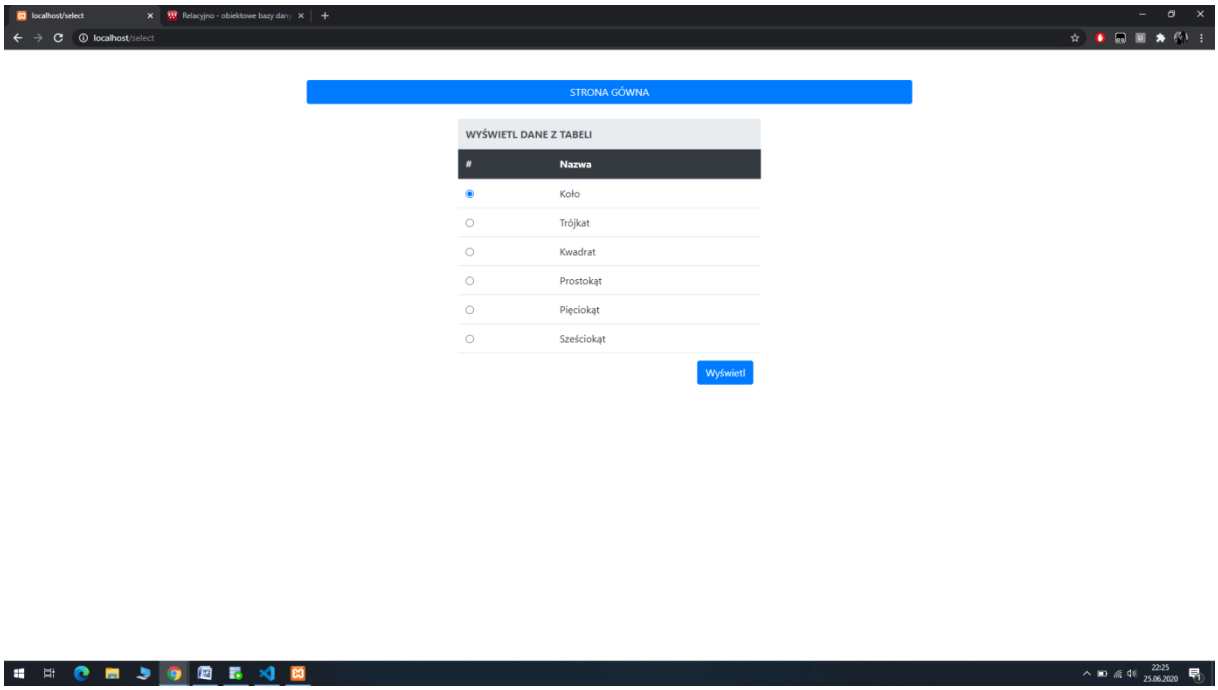
Frontend serwisu został wykonany przy pomocy języka HTML oraz kaskadowych arkuszy styli CSS. Do formatowania stron serwisu został również wykorzystany Framework Bootstrap, który zapewnia dostęp do zaawansowanych narzędzi formatowania wyglądu stron internetowych.

Na rys. 6 przedstawiono stronę główną serwisu, z której użytkownik może uzyskać dostęp do każdej jego funkcjonalności.



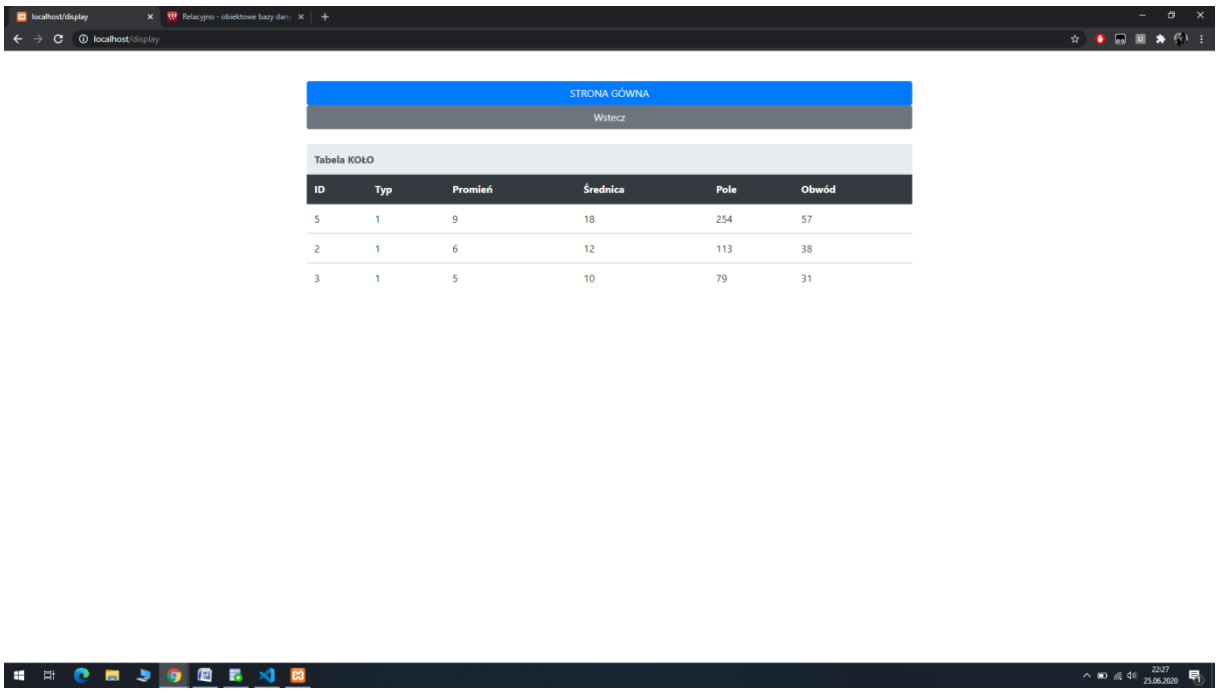
rys. 6 Strona główna serwisu

Na rys. 7 przedstawiono stronę służącą do wyboru, z której tabeli mają zostać wyświetlone dane.



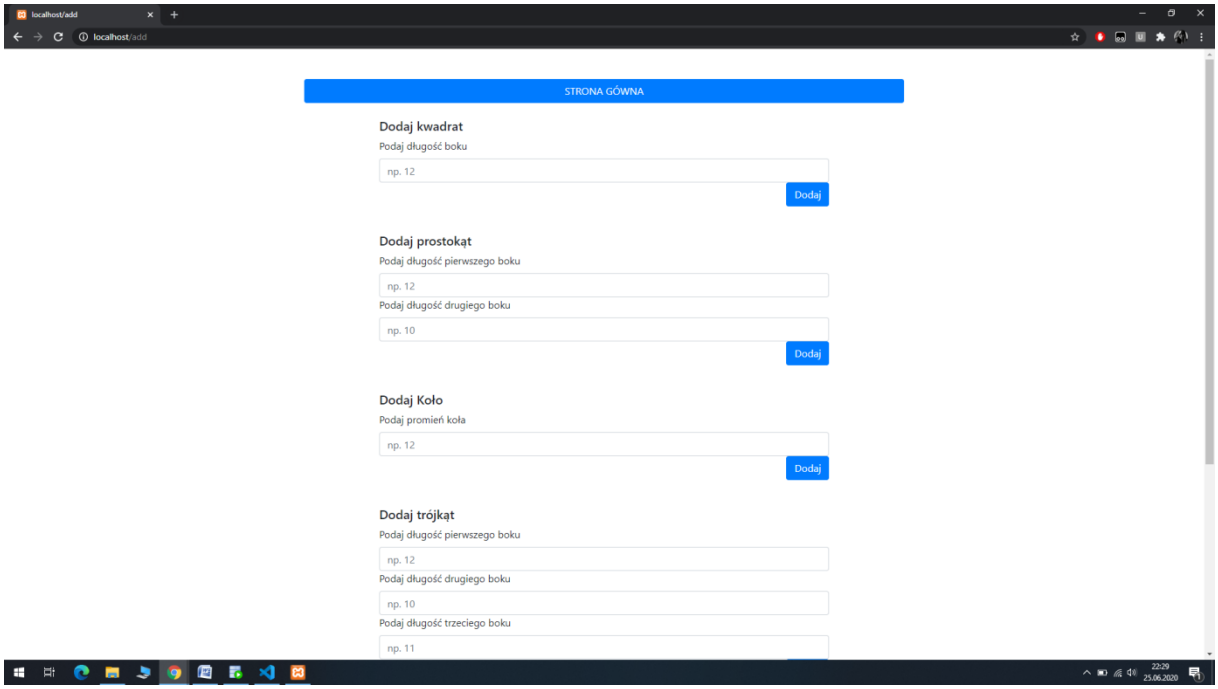
rys. 7 Strona służąca do odczytu danych

Na rys. 8 przedstawiono stronę wyświetlającą wszystkie dane z konkretnej tabeli.



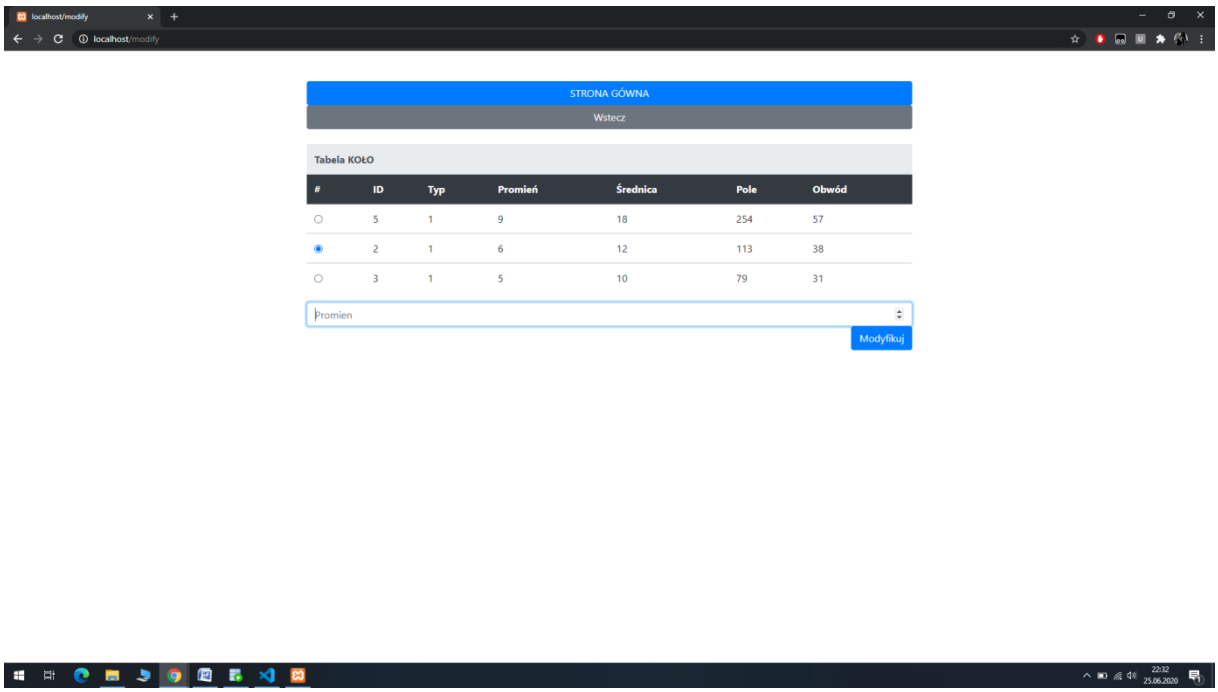
rys. 8 Wyświetlenie danych z tabeli

Na rys. 9 przedstawiono stronę, na której użytkownik może dodać dane do wybranej tabeli.



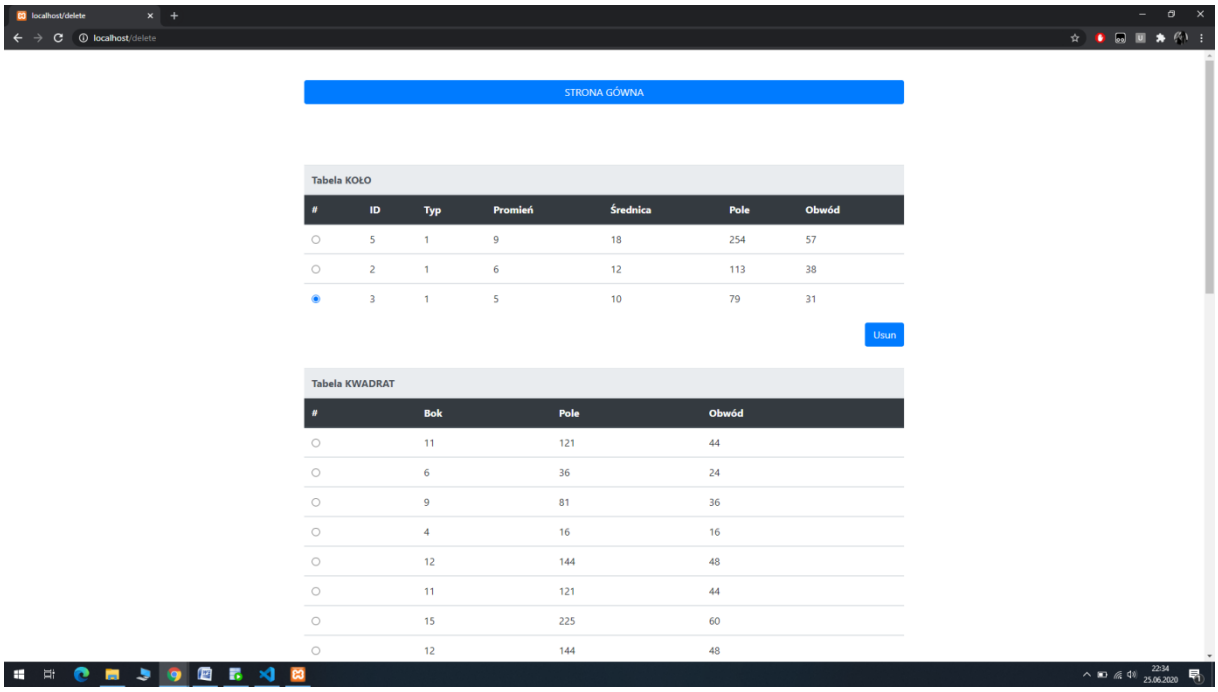
rys. 9 Strona z formularzami służącymi do dodawania danych do relacyjnej bazy danych

Na rys. 10 przedstawiono przykładową stronę służącą do modyfikowania rekordów w wybranej tabeli.



rys. 10 Modyfikacja wybranego rekordu z wybranej tabeli

Na rys. 11 przedstawiono stronę służącą do usuwania wybranego rekordu z bazy danych.



rys. 11 Strona do usuwania rekordów z bazy danych

4.2. Backend serwisu

Backend serwisu wykonano przy użyciu języka *PHP* oraz serwera *Apache*. W projekcie zastosowano wzorzec *MVC*(*Model, View, Controller*), czyli podział ma model, widok oraz kontrolery. Kontrolery użyte w projekcie wykorzystują klasy oraz ich metody w celu współpracy z bazą danych oraz przetwarzania informacji pobranych od użytkownika. Następnie kontrolery udostępniają wyniki wykonanych działań do widoków, tak, aby można ich swobodnie używać. Połączenie z bazą danych realizowano za pomocą klasy *PDO*. W serwisie stworzono klasy reprezentujące wybrane figury geometryczne, w których określono ich zachowanie. Na rys. 13 przedstawiono przykładową klasę *Kolo*. Zawarto w niej pola opisujące jednoznacznie obiekt *Kolo*. Klasa ta zawiera również metody obliczające pole, obwód oraz średnicę danego koła, tak, aby użytkownik podczas dodawania nowego koła do bazy danych musiał jedynie podać promień koła, które chce dodać. Aby ułatwić komunikację z bazą danych klasa ta została wyposażona w metody takie jak: *modify, getFromDB, LoadToDB, LoadToDBOBJ.* Metody te ułatwiają interakcję z bazą danych przy użyciu klasy *Kolo.*

```php
<?php
class Kolo{
   private $id;
   private $typ;
   private $promien;
   private $srednica;
   private $pole;
   private $obwod;

   public function setId($id){
      $this->id = $id;
   }

   public function modify($app, $tabela){
      $app['database']->modifyKolo($this->id, $this->promien, $this->srednica, $this->obwod,
                                   $this->pole, $tabela);
   }
    private function obliczPole($promien){
      $this->pole = pow($promien, 2)*pi();
   }
   private function obliczObwod($promien){
      $this->obwod = $promien*2*pi();
   }
   public function getFromUser($promien){
      $this->promien=$promien;
      $this->srednica=2*$promien;
      $this->obliczObwod($promien);
      $this->obliczPole($promien);
   }
   public function getFromDB($id){
      $result = $app['database']->selectOne($id, "kolo");
      $this->id=$result['id'];
      $this->typ=$result['typ'];
      $this->promien=$result['promien'];
      $this->srednica=$result['srednica'];
      $this->obwod=$result['obwod'];
      $this->pole=$result['pole'];
   }

   public function displayKolo(){
      echo "Kolo o id: ".$this->id."<br>";
      echo "Promien to: ".$this->promien."<br>";
      echo "Srednica to: ".$this->srednica."<br>";
      echo "Pole to: ".$this->pole."<br>";
      echo "Obwod to: ".$this->obwod."<br>";
   }
   public function LoadToDB($app){
      $app['database']->addKolo($app, $this->promien, $this->srednica, $this->obwod, $this->pole);
   }
   public function LoadToDBOBJ($app){
      $app['database']->addKoloOBJ($app, $this->pole, $this->obwod, $this->promien,
                                   $this->srednica );
   }
}
```

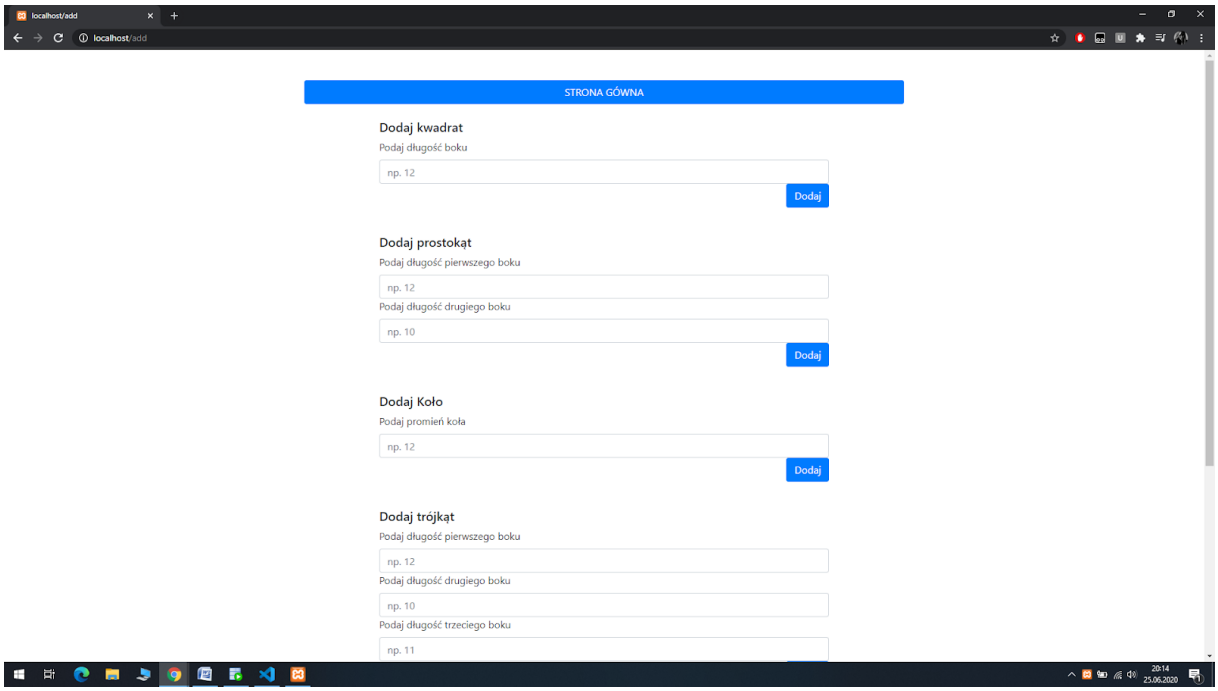
rys. 12 Klasa Kolo

5. Relacyjna baza danych

Relacyjna baza danych to rodzaj bazy danych, który pozwala przechowywać powiązane ze sobą elementy danych i zapewnia do nich dostęp. Relacyjne bazy danych są oparte na modelu relacyjnym — jest to prosty i intuicyjny sposób przedstawiania danych w tabelach. W relacyjnej bazie danych każdy wiersz tabeli jest rekordem z unikatowym identyfikatorem nazywanym kluczem. Kolumny tabeli zawierają atrybuty danych, a każdy rekord zawiera zwykle wartość dla każdego atrybutu, co ułatwia ustalenie relacji między poszczególnymi elementami danymi.

5.1. Operacje na bazie danych

Do wykonywania operacji na bazie danych służą stworzone w języku HTML widoki zawierające formularze (rys. 13) umożliwiające przeprowadzenie pożądanych działań wewnątrz bazy danych.



rys. 13 Przykładowy formularz służący do dodawania danych do relacyjnej bazy danych

5.1.1. Dodawanie

Na rys. 15 przedstawiono metodę z klasy *Queries*, który wykorzystywana jest przez klasę Kolo w celu załadowania danych do relacyjnej bazy danych. Analogiczne metody zostały wykorzystane dla każdej ze stworzonych klas w celu dodawania informacji do bazy danych. Metoda *addKolo* wykorzystuje przekazane jej parametry, które zostały przygotowane przez klasę Kolo na podstawie danych uzyskanych od użytkownika do tego, aby utworzyć nowy rekord w tabeli Kolo relacyjnej bazy danych. Następnie

przygotowywana jest kwerenda, która zostanie przekazana do bazy danych i wykonana wewnątrz niej.

```
public function addKolo($app, $promien, $srednica, $obwod, $pole){
    $nextId=$app['database']->maxId("kolo")+1;
    $query = $this->pdo->prepare(
        "INSERT INTO sys.kolo(id, typ, promien, srednica, obwod, pole)
        VALUES({$nextId}, 1, '{$promien}', '{$srednica}', '{$obwod}', '{$pole}')");
    $query->execute();
}
```

rys. 14 Metoda *addKolo* z klasy Queries

5.1.2. Usuwanie

Na rys. 16 przestawiono metodę *delete*, służącą do usuwania rekordu z bazy danych. Metoda ta wykorzystywana jest przez wszystkie rodzaje obiektów reprezentujących figury geometryczne wewnątrz naszego systemu. Metoda *delete* przyjmuje dwa parametry, pierwszym z nich jest nazwa tabeli, z której ma zostać usunięty dany rekord, a drugim jest identyfikator danego rekordu do usunięcia.

```
public function delete($tabela, $id){
    $query = $this->pdo->prepare("DELETE FROM sys.{$tabela} WHERE id={$id}");
    $query->execute();
}
```

rys. 15 Metoda *delete*

5.1.3. Modyfikacja

Na rys. 17 przedstawiono metodę modyfikującą istniejące rekordy wewnątrz relacyjnej bazy danych. Jest to metoda wykorzystywana przez klasę Kolo. Użytkownik podaje nowy promień wybranego koła, a następnie klasa Kolo dokonuje obliczeń pozostałych parametrów oraz wywołuje funkcję *modifyKolo* przekazując jej obliczone przez siebie uprzednio wartości parametrów.

```
public function modifyKolo($id, $promien, $srednica, $obwod, $pole, $tabela){
    $query = $this->pdo->prepare("
    UPDATE sys.{$tabela} SET promien = {$promien}, srednica = {$srednica}, pole = {
$pole}, obwod ={$obwod}
    WHERE id = {$id}
    ");
    $query->execute();
}
```

rys. 16 Metoda *modifyKolo*

5.1.4. Wyświetlanie

Na rys. 18 przedstawiono metodę służącą do wyświetlania danych z konkretnej tabeli. Metoda ta używana jest we wszystkich miejscach systemu, w których niezbędne jest wyświetlenie danych z danej tabeli takich jak: usuwanie danych, modyfikacja danych oraz samo wyświetlanie danych. Metoda ta jest metodą uniwersalną, pozwalającą na pobranie danych z dowolnej tabeli przy pomocy określenia nazwy tabeli poprzez przekazanie jej przez parametr. Dane zwrócone przez metodę można wyświetlić pojedynczo lub przy pomocy pętli foreach, wyświetlić wszystkie pobrane rekordy. Dane wyświetlane mogą być w dowolny sposób formatowane za pomocą nazw kolumn, tak aby wyświetlone zostały tylko potrzebne dane.

```
public function selectAll($table){
    $query = $this->pdo->prepare("SELECT * FROM {$table}");
    $query->execute();
    return $query->fetchAll(PDO::FETCH_CLASS);
}
```

rys. 17 Metoda *selectAll*

6. Obiektowo-relacyjna baza danych

Obiektowo-relacyjne bazy danych są hybrydowymi architekturami baz danych. Stanowią kompromis między relacyjnymi, a obiektowymi bazami danych. Obiektowe bazy danych nie mają tak szerokiej infrastruktury jak relacyjne, a budowanie ich od początku wiąże się z dużymi kosztami. Pozwalają na operowanie na danych jak na obiektach, posiadają jednak bazę relacyjną jako wewnętrzny mechanizm przechowywania danych. Systemy obiektowo-relacyjne są wyposażane w wiele cech umożliwiających efektywną produkcję aplikacji. Wśród nich można wymienić przystosowanie do multimediów, dane przestrzenne, abstrakcyjne typy danych, metody (funkcje i procedury) definiowane przez użytkownika w różnych językach, kolekcje (zbiory, sekwencje, zagnieżdżone tablice, tablice o zmiennej długości), typy referencyjne, przeciążanie funkcji, późne wiązanie i inne.

6.1. Operacje na bazie danych

Do wykonywania operacji na bazie danych służą stworzone w języku HTML widoki zawierające formularze umożliwiające przeprowadzenie pożądanych działań wewnątrz bazy danych.

6.1.1. Dodawanie

```
public function addKoloOBJ($app, $pole, $obwod, $promien, $srednica ){
  $nextId=$app['database']->maxId("koloOBJ")+1;
  $query = $this->pdo->prepare(
    "Insert into sys.koloOBJ
    values({$nextId}, '{$pole}', '{$obwod}', '{$promien}', '{$srednica}')");
  $query->execute();
}
```

rys. 18 Metoda *addKoloOBJ*

Dodawanie nowych wierszy do tabel utworzonych na podstawie typów może zostać wykonane w bardzo podobny sposób do tego jak jest to wykonywane dla bazy relacyjnej. Znaczącą różnicą jest to że podanie nazw kolumn do których chcemy załadować dane nie jest wymagane, należy jedynie pamiętać aby zachować odpowiednią kolejność przekazywania danych.

6.1.2. Usuwanie

```
public function deleteObj($tabela, $id){
  $id=$id*1;
  $query = $this->pdo->prepare("delete from sys.{$tabela} o where o.id={$id}");
  $query->execute();
}
```

rys. 19 Metoda *deleteOBJ*

Usuwanie danych z tabeli obiektowej wykonuje się za pomocą wykonania kwerendy *DELETE.* Aby określić który obiekt chcemy usunąć trzeba podać jakąś wartość parametru, aby nie usunąć przypadkowo innych obiektów. Najlepszym wyjściem jest podanie jakiegoś rodzaju wartości unikatowej. Można to osiągnąć za pomocą OID(OBJECT IDENTIFIER), który jest automatycznie generowanym unikatowym identyfikatorem tworzonym podczas dodawania nowego obiektu do tabeli. Jednak można również podczas tworzenia tabeli typu obiektowego określić jedno z pól jako klucz główny i to rozwiązanie zostało użyte w naszym projekcie. Takie rozwiązanie zostało użyte głównie z powodu dziwnego zachowania bazy danych podczas próby pobrania OID z danej tabeli.

6.1.3. Modyfikacja

```
public function modifyKolo($id, $promien, $srednica, $obwod, $p
ole, $tabela){
  $query = $this->pdo->prepare("
Update sys.{$tabela} set promien = {$promien}, srednica = {$sre
dnica}, pole = {$pole}, obwod = {$obwod} where id = {$id}");
        $query->execute();
    }
```

rys. 20 Metoda modifikacji

6.1.4. Wyświetlanie

```
  public function selectAll($table){
    $query = $this->pdo->prepare("select * from {$table}");
    $query->execute();
    return $query->fetchAll(PDO::FETCH_CLASS);
  }
```

rys. 21 Metoda wyświelania

Pobieranie danych z tabel obiektowych można wykonywać w praktycznie identyczny jak w przypadku tabel bazy relacyjnej. Natomiast podczas pobierania danych z tabel obiektowych mamy możliwość wykonywania funkcji czy procedur składowanych wewnątrz bazy danych.

7. Porównanie relacyjnych oraz obiektowo-relacyjnych baz danych

Relacyjne bazy danych oraz relacyjno-obiektowe po odpowiedniej konfiguracji mogą zachowywać się w bardzo podobny sposób. Główną różnicą jest deklarowanie własnych typów oraz tworzenie tabel przy ich pomocy. Jest to rozwiązanie pozwalające na tworzenie tabel z tej samej kategorii, różniącymi się jedynie szczegółami, można to osiągnąć używając dziedziczenia tworząc nowe typy ściśle powiązane z nadtypem. Dużą różnicą jest możliwość składowania metod wewnątrz bazy danych co pozwala na zaoszczędzenie implementowania niektórych funkcjonalności wewnątrz systemów w porównaniu do systemów opartych na relacyjnych bazach danych. Wykonywanie zapytań nie różni się w dużym stopniu między oboma rodzajami baz danych. Największą różnicą w tych dwóch typach baz danych jest możliwość określenia własnych typów danych przez użytkownika.

8. Podsumowanie

Podczas tworzenia projektu zapoznaliśmy się z pojęciem obiektowo-relacyjnych baz danych. Tworząc projekt nauczyliśmy się tworzyć typy użytkownika wewnątrz bazy danych oraz korzystać z nich w celu tworzenia tabel. Porównując relacyjną bazę danych z obiektowo-relacyjną bazą danych zauważyliśmy, że obiektowo-relacyjna baza danych ułatwia użytkownikom wykonywanie niektórych zadań w porównaniu z relacyjną bazą. Użytkownik obiektowo-relacyjnej bazy danych dzięki możliwości definiowania własnych typów ma możliwość tworzenia za ich pomocą tabel, co jest bardzo przydatną funkcją w sytuacji gdy należy stworzyć tabele o zbliżonej do siebie strukturze, które jednak nie są ze sobą w żaden sposób połączone, w takiej sytuacji możemy korzystać z wcześniej utworzonych typów. Ten rodzaj bazy danych umożliwia również tworzenie metod składowanych wewnątrz bazy danych z których możemy korzystać np. podczas odczytywania danych z bazy czy aktualizowania ich. Obiektowy aspekt baz danych zapewnia takie funkcje jak przeciążanie, przysłanianie funkcji i dziedziczenie po nadtypach, co w dużym stopniu upodabnia pracę z bazą danych do tej wykonywanej w obiektowych językach programowania.

Pomimo swoich oczywistych zalet obiektowo-relacyjne bazy danych nie wyparły z rynku swojego czysto relacyjnego odpowiednika. Taka sytuacja ma miejsce głównie z dwóch powodów. Pierwszym z nich jest wysoki koszt migracji istniejących rozwiązań na używanie obiektowo-relacyjnych baz danych. Drugim powodem jest niski stopień wspierania tego rodzaju rozwiązania przez duże firmy zajmujące się rozwojem systemów bazodanowych co skutkuje małą ilością specjalistów zajmujących się rozwiązaniami tego typu.