

# POLITECHNIKA GDAŃSKA

## WYDZIAŁ FIZYKI TECHNICZNEJ I MATEMATYKI STOSOWANEJ

### STUDIA PODYPLOWE PROGRAMOWANIE I BAZY DANYCH 2015/16

#### TEMAT PROJEKTU: DOKUMENTACJA TECHNICZNA APLIKACJI „BLOG O FIRMIE”



WYKONAŁ: MGR INŻ. DAWID SARNECKI

copyright Dawid Sarnecki [sarnecki.dawid@wp.pl](mailto:sarnecki.dawid@wp.pl), 2016**SPIS TREŚCI**

1. Wstęp	4
2. Opis wykorzystanych technologii	4
3. Opis oprogramowania	4
4. Instrukcja użytkownika	5
4.1. elementy główne	5
4.2. użytkownicy	6
4.2.1. rejestracja użytkownika	6
4.2.2. logowanie i utworzenie sesji	6
4.2.3. panel użytkownika	7
4.2.4. informacje o użytkownikach	7
4.2.5. zarządzanie uprawnieniami i blokadą konta	7
4.3. wpisy	8
4.3.1. nowy wpis	8
4.3.2. zarządzanie wpisem	8
4.3.3. zmiana statusu	8
4.3.4. edycja wpisu	8
4.3.5. przenoszenie do archiwum wpisów	9
4.3.6. zarządzanie archiwum wpisów	9
4.4. komentarze	9
4.4.1. dodawanie komentarzy	9
4.4.2. usuwanie komentarzy	9
5. Struktura plików	10
6. Klasy	10
6.1. klasa connection	10
6.1.1. schemat	10
6.1.2. opis pól	10
6.1.3. opis metod	11
7. Opis algorytmów	12
7.1. komunikacja z bazą danych	12
7.2. wyświetlanie elementów głównych	12
7.3. użytkownicy	13
7.3.1. rejestracja użytkownika	13
7.3.2. logowanie i utworzenie sesji	14
7.3.3. panel zalogowanego użytkownika	14
7.3.4. informacje o użytkownikach	14
7.3.5. zarządzanie uprawnieniami	15

7.3.6.blokowanie konta użytkownika	16
7.4. wpisy	16
7.4.1.nowy wpis	16
7.4.2.edycja wpisu	16
7.4.3.zmiana statusu	17
7.4.4.wyświetlanie wpisów i komentarzy	17
7.4.5.przenoszenie do archiwum wpisów	18
7.4.6.wyświetlanie archiwum wpisów	18
7.4.7.przywracanie z archiwum	18
7.4.8.usuwanie wpisów	18
7.5. komentarze	19
7.5.1.dodawanie komentarzy	19
7.5.2.usuwanie komentarzy	19
8. struktura bazy danych	20
8.1. opis	20
8.2. diagram err	20
8.3. tabele	21
8.3.1.tabela user	21
8.3.2.tabela blogtext	21
8.3.3.tabela blogcommnet	21
8.3.4.tabela activity	22
8.4. wyzwalacze	23
8.5. widoki	25
8.5.1. widok: all_blog	25
8.5.2. widok: blog_public	25
8.5.3. widok: all_comments	26
8.5.4. widok: archive	26
8.5.5. widok: show_user	27
9. wykaz literatury	28

## 1. WSTĘP

Dokumentacja opisuje aplikację sieciową – „Blog O firmie” i jest przeznaczona dla użytkowników i programistów.

- Co dokumentacja zawiera
  - część użytkową : opis aplikacji i jej przeznaczenie, wymagania, instrukcja obsługi
  - część techniczną: struktura plików, opis algorytmów dla poszczególnych zadań, struktura bazy danych

## 2. OPIS WYKORZYSTANYCH TECHNOLOGII

Aplikacja wykorzystuje następujące technologie

- PHP 5
- MySQL 5.6
- HTML 5
- CSS3
- javascript
- Bootstrap Framework
- AJAX

## 3. OPIS OPROGRAMOWANIA

Aplikacja pozwala wyświetlać stronę internetową składającą się z 3 zakładek:

- „O Firmie” : wyświetlanie grafiki i informacji tekstowych
- „Blog” : wyświetlanie wpisów, rejestracja i logowanie użytkownika, zarządzanie użytkownikami, dodawanie, edycja wpisów, zarządzanie wpisami, dodawanie i usuwanie komentarzy
- „Kontakt”: wyświetlanie aktualnej lokalizacji użytkownika na mapie Google

Wymagania dla działania aplikacji:

- dla serwera:
  - serwer Apache 2.0
  - serwer MySQL 5.6
- dla klienta:
  - aktualna przeglądarka internetowa (Mozilla Firefox, IE, Google Chrome)

Wyświetlany widok aplikacji automatycznie dostosowuje się do rozdzielczości wyświetlacza na urządzeniu, co zapewnia technologia responsywności wykorzystana w aplikacji.

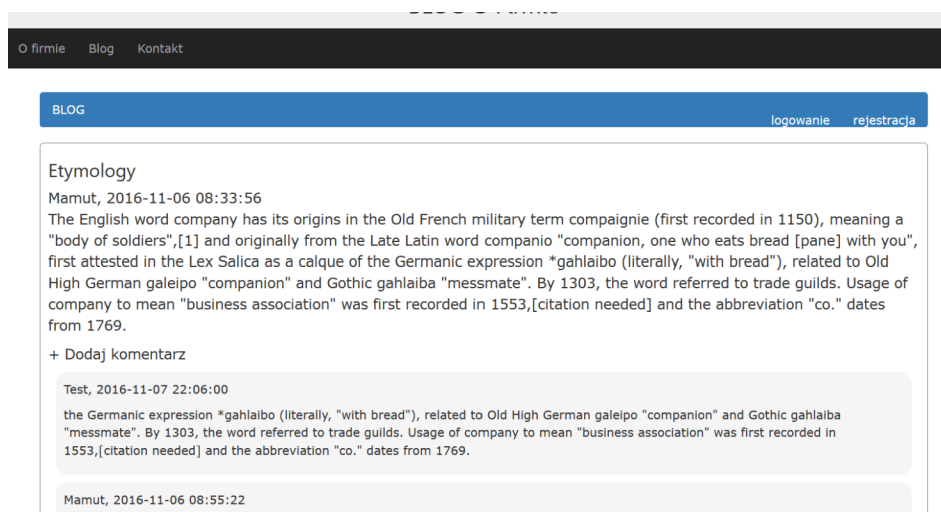
## 4. INSTRUKCJA UŻYTKOWNIKA

### 4.1. ELEMENTY GŁÓWNE

- „O Firmie” : wyświetlanie grafiki i informacji tekstowych, widok łąduje się jako domyślny przy otwieraniu strony



- „Blog” : wyświetlanie aktywnych wpisów i dodanych komentarzy. Dodawanie komentarzy jest możliwe po zalogowanie się na konto



- „Kontakt”: wyświetlanie aktualnej lokalizacji użytkownika na mapie Google



## 4.2. UŻYTKOWNICY

### 4.2.1. rejestracja użytkownika

Formularz rejestracji wyświetla się po kliknięciu na link „Rejestracja”. Rejestracja użytkownika jest możliwa po uzupełnieniu wszystkich pól formularza rejestracji.

Wprowadź swoje dane, aby się zarejestrować.

### NOWE KONTO

**Login:**   
 X Login już istnieje.

**email:**

**Hasło:**

[Załącz konto](#) Posiadasz już konto? Kliknij aby się zalogować.

Po wprowadzeniu loginu formularz sprawdza „w tle” jego dostępność i wyświetla informację o stanie : „Login już istnieje” lub „Login jest dostępny”.

W przypadku nieuzupełnienia wszystkich pól pojawia się komunikat:

**! Nie wszystkie pola zostały wypełnione.**

W przypadku gdy wprowadzony login już istnieje w bazie danych pojawia się komunikat:

**! Użytkownik o takiej nazwie już istnieje.**

### 4.2.2. logowanie

Formularz logowanie wyświetla się po kliknięciu na link „Logowanie”, lub po założeniu nowego konta. Rejestracja użytkownika jest możliwa po uzupełnieniu wszystkich pól formularza logowania.

O firmie Blog Kontakt

### LOGOWANIE

**login:**

**hasło:**

[Zaloguj](#) Nie posiadasz konta? Kliknij tutaj.

W przypadku nieuzupełnienia wszystkich pól pojawia się komunikat:

**! Błędna nazwa użytkownika lub hasło.**

W przypadku gdy konto użytkownika zostało zablokowane pojawia się komunikat:

**! Konto [ MAMI ] zostało zablokowane. Wymagany kontakt z administratorem, e-mail: admin@blog.pl.**

#### 4.2.3. panel użytkownika

#### 4.2.4. informacje o użytkownikach

#### 4.2.5. zarządzanie uprawnieniami i blokadą konta

Użytkownik z rolą ADMIN ma możliwość zarządzania uprawnieniami. Przycisk „Zmień Rolę” umożliwia zmianę roli USER/ADMIN. Przycisk „Zablokuj/Odblokuj” umożliwia zablokowanie/odblokowanie konta.

## 4.3. WPISY

### 4.3.1. nowy wpis

Aby dodać nowy wpis należy uzupełnić pola Tytuł, Treść i kliknąć przycisk „Dodaj wpis”.

Nowy Wpis

Tytuł:

The English word company has its origins

Treść:

The English word company has its origins in the Old French military term *compaignie* (first recorded in 1150), meaning a "body of soldiers",[1] and originally from the Late Latin word *companiono* "companion, one who eats bread [pane] with you", first attested in the Lex *Salica* as a *calque* of the Germanic expression *\*gahlaibo* (literally, "with bread"), related to Old High German *galeipo* "companion" and Gothic *gahlaiba* "messmate". By 1303, the word referred to trade guilds. Usage of company to mean "business association" was first recorded in 1553,[citation needed] and the abbreviation "co." dates from 1769.

Dodaj wpis

### 4.3.2. zarządzanie wpisem

Użytkownik z rolą ADMIN może edytować wszystkie wpisy, użytkownik USER tylko swoje wpisy.

Do zarządzania wpisem służą przyciski pojawiające się pod każdym wpisem.

status: Wyświetlany

Ukryj

Edytuj wpis

Skomentuj

Do archiwum

### 4.3.3. zmiana statusu wpisu

Status wpisu określa czy dany wpis jest widoczny dla niezalogowanego użytkownika.

Domyślnym statusem dla nowych wpisów jest status „Ukryty”.

Pod każdym wpisem pojawia się aktualny status wpisu, a obok niego przycisk Ukryj(Wyświetl) zmieniający status wpisu.

### 4.3.4. edycja wpisu

Po kliknięciu na przycisk „Edytuj” pojawia się okno edycji wpisu

BLOG

witaj:) [ Mamut ] wyloguj

Edycja wpisu

Tytuł:

Etymology

Treść:

The English word company has its origins in the Old French military term *compaignie* (first recorded in 1150), meaning a "body of soldiers",[1] and originally from the Late Latin word *companiono* "companion, one who eats bread [pane] with you", first attested in the Lex *Salica* as a *calque* of the Germanic expression *\*gahlaibo* (literally, "with bread"), related to Old High German *galeipo* "companion" and Gothic *gahlaiba* "messmate". By 1303, the word referred to trade guilds. Usage of company to mean "business association" was first recorded in 1553,[citation needed] and the abbreviation "co." dates from 1769.

Zapisz zmiany

Anuluj zmiany

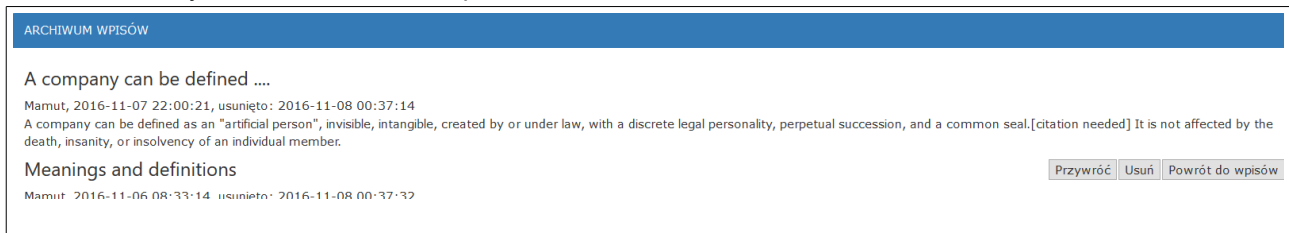


#### 4.3.5. przenoszenie do archiwum wpisów

Po kliknięciu na przycisk „Do archiwum” wpis zostaje przeniesiony do do Archiwum wpisów. Dostęp do zakładki archiwum ma tylko użytkownik z rolą ADMIN.

#### 4.3.6. zarządzanie archiwum wpisów

Po kliknięciu na przycisk „Archiwum” (widoczny tylko dla użytkownika z rolą ADMIN) zostaje wyświetlone Archiwum wpisów.



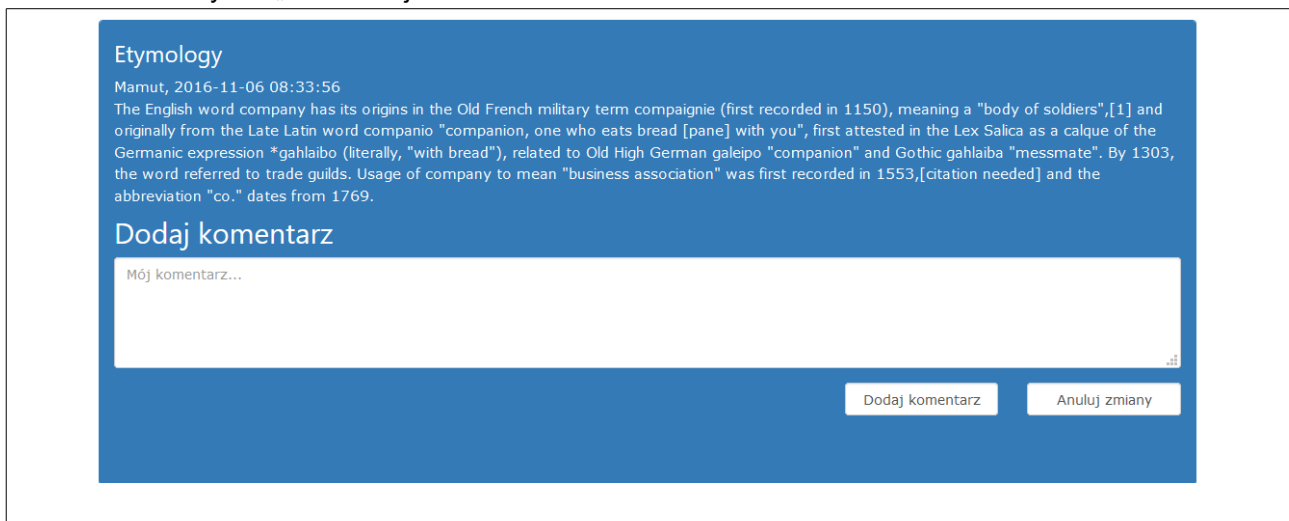
Przycisk „Przywróć” przywraca wpis wraz z przypisanymi do niego komentarzami.

Przycisk „Usuń” usuwa wpis z bazy danych wraz z przypisanymi do niego komentarzami.

### 4.4. KOMENTARZE

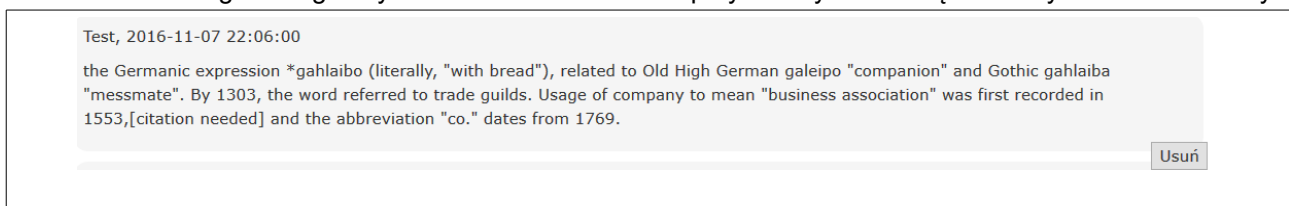
#### 4.4.1. dodawanie komentarzy

Przycisk „Skomentuj” otwiera widok dodawania komentarza.

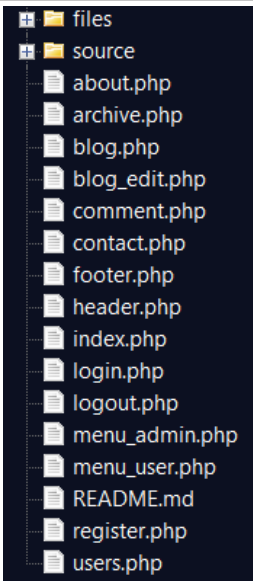
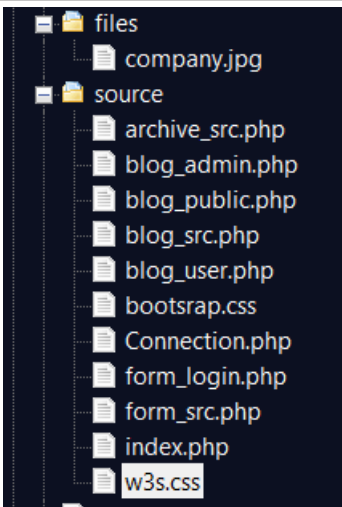


#### 4.4.2. usuwanie komentarzy

Przycisk „Usuń” umożliwia usuwanie komentarzy. Widoczność przycisku jest uzależniona od roli użytkownika. Rola USER pozwala na wyświetlanie przycisku tylko pod komentarzami zalogowanego użytkownika. Dla roli ADMIN przycisk wyświetla się dla wszystkich komentarzy.



## 5. STRUKTURA PLIKÓW

katalog główny	katalogi /source i /files
	

## 6. KLASY

### 6.1. KLASA CONNECTION

#### 6.1.1. schemat

```

private $dbhost;
private $dbname;
private $dbuser;
private $dbpass;
private $conn;
private $error;
private $query;

public function __construct()
public function query($query)
public function resultset()
public function execute()
public function bindValue($param, $value, $type)

```

#### 6.1.2. opis pól

private \$dbhost	nazwa hosta
private \$dbname	nazwa bazy danych
private \$dbuser	nazwa użytkownika bazy danych
private \$dbpass	hasło do bazy danych
private \$conn	pdo ( php data object)
private \$error	informacja o wystąpieniu błędu podczas inicjalizacji obiektu \$conn (pdo)
private \$query	zapytanie sql przygotowane do wykonania na bazie danych

#### 6.1.3. opis metod

public function __construct()	<p>przyjmuje: nic</p> <p>co robi: do pola \$conn przypisanie obiektu PDO, w przypadku wystąpienia błędu wydrukowanie komunikatu z treścią błędu na ekranie; ustawienie kodowania na utf-8</p> <p>zwraca: nic</p>
public function query(\$query)	<p>przyjmuje: \$query [string]</p> <p>co robi: metoda dla otrzymanej wartości \$query wywołuje metodę prepare zmiennej obiektowej \$conn, zwrócona wartość jest przypisywana do pola \$query</p> <p>zwraca: nic</p>
public function execute()	<p>przyjmuje: nic</p> <p>co robi: wykonuje wyrażenie sql przypisane do pola \$query na bazie danych</p> <p>zwraca: wynik wykonanego wyrażenia sql</p>
public function resultSet()	<p>przyjmuje: nic</p> <p>co robi: wywołuje metody execute()</p> <p>zwraca: tablicę asocjacyjną z wynikami zapytania</p>
public function bindValue (\$param, \$value, \$type)	<p>przyjmuje: (\$param, \$value, \$type)</p> <p>co robi: wywołanie metody bindValue(\$param, \$value, \$type) dla pola \$query (typ obiektowy PDO)(podmiana wartości)</p> <p>zwraca: tablicę asocjacyjną</p>

## 7. OPISY ALGORYTMÓW

## 7.1. KOMUNIKACJA Z BAZĄ DANYCH

Za komunikację z bazą danych odpowiada klasa *Connection* opisana w p. 6.1.

Plik nagłówkowy *header.php* otwiera sesję i tworzy nowy obiekt *\$connect* klasy *Connection*.

```
<?php
session_start();
require_once 'source/Connection.php';
$connect = new Connection();
?>
```

## 7.2. WYŚWIETLANIE ELEMENTÓW STAŁYCH

Elementy stałe to nagłówek strony, menu i stopka strony oraz strony: „O firmie” i „Kontakt”

Zawartość pliku *index.php*:

```
<?php
require_once 'header.php';
require_once 'about.php';
require_once 'footer.php';
?>
```

Plik nagłówkowy *header.php* sprawdza czy użytkownik jest zalogowany weryfikując czy do zmiennej sesji o nazwie *user* została przypisana jakaś wartość.

```
<?php
$userstr;
if (isset($_SESSION['user']))
{
    $user = $_SESSION['user'];
    $loggedin = TRUE;
    $userstr = " [ $user ]";
}
else $loggedin = FALSE;
?>
```

Zawartość strony *Blog* zależy od tego czy użytkownik jest zalogowany oraz czy jest administratorem. Sprawdzenie warunków zostało zaimplementowane w pliku *blog.php*

```
<?php
if ($loggedin)
{
    if ($_SESSION['IsAdmin'])
    {
        require_once 'menu_admin.php';
        require_once 'source/form_login.php';
        require_once 'source/blog_admin.php';
    }
    else
    {
        require_once 'menu_user.php';
        require_once 'source/form_login.php';
        require_once 'source/blog_user.php';
    }
}
else
    require_once 'source/blog_public.php';
?>
```

### 7.3. UŻYTKOWNICY

#### 7.3.1. rejestracja użytkownika

Za rejestrację użytkownika odpowiada plik `register.php`. Formularz kontaktowy rejestracyjny wysyła wymagane dane metodą POST do zmiennej tablicowej `$_SERVER['PHP_SELF']`. AJAX umożliwia dynamiczne sprawdzenie wprowadzonego loginu i wyświetlenie informacji o jego dostępności.

Następnie instrukcje warunkowe sprawdzają poprawność wprowadzonych danych:

- sprawdzenie wypełnienia wszystkich pól
  - sprawdzenie czy wprowadzony login istnieje już w bazie danych
- Kolejny krok to operacja INSERT dla tabeli *user*.

Do użytkownika wysyłane są trzy rodzaje komunikatu:

- „! Nie wszystkie pola zostały wypełnione”
- „! Użytkownik o takiej nazwie już istnieje.”
- „Konto zostało utworzone. Proszę się zalogować.”

Wprowadzone hasło użytkownika jest „solone” i hashowane algorytmem *whirlpool*.

```
$salt1="!@#?><";
$salt2="^%yt";
$token = hash( 'whirlpool', "$salt1$pass$salt2");
```

Kod PHP odpowiadający za obsługę formularza rejestracji:

```
<?php
$user = $pass = "";
if (isset($_SESSION['user'])) destroySession();

if (isset($_POST['user']))
{
    $user = ($_POST['user']);
    $pass = ($_POST['pass']);
    $email = ($_POST['email']);

    $salt1="!@#?><";
    $salt2="^%yt";
    $token = hash( 'whirlpool', "$salt1$pass$salt2");

    if ($user == "" || $pass == "" || $email == "" )
        echo "<h3><b> ! Nie wszystkie pola zostały wypełnione.<br></b></h3>";
    else
    {
        $connect->query("SELECT * FROM User WHERE login=:user");
        $connect->bindValue(':user', $user, PDO::PARAM_STR);
        $rows = $connect->resultset();

        if ($rows)
            echo "<h3><b> ! Użytkownik o takiej nazwie już istnieje.<br><br></b></h3>";
        else
        {
            $connect->query("INSERT INTO User (Login,Pass,Email) VALUES(:user, :pass, :email)");
            $connect->bindValue(':user', $user, PDO::PARAM_STR);
            $connect->bindValue(':pass', $token, PDO::PARAM_STR);
            $connect->bindValue(':email', $email, PDO::PARAM_STR);
            $connect->execute();

            die("<h4>Konto zostało utworzone<a href =login.php> Proszę się zalogować.</a><br>");
        }
    }
}
?>
```

### 7.3.2. logowanie i utworzenie sesji

Za logowanie i utworzenie sesji odpowiada plik *login.php*. Wysyłanie informacji z formularza logowania odbywa się analogicznie jak w p.7.3.1

W przypadku poprawnych logowania tworzone są 3 zmienne *sesji*: *user*, *ID* i *IsAdmin*:

```
$_SESSION['user'] = $user;
$_SESSION['ID'] = $rows[0]['ID'];
$_SESSION['IsAdmin'] = $rows[0]['IsAdmin'];
```

Po zalogowaniu użytkownik zostaje przekierowany na stronę z blogiem za co odpowiada kod:

```
header("Location: blog.php")
```

Za logowanie i utworzenie sesji odpowiada plik *login.php*. Wysyłanie informacji z formularza logowania odbywa się analogicznie jak w p.7.3.1

### 7.3.3. panel zalogowanego użytkownika

```
<ul class="nav navbar-nav navbar-right">
    <li><a>witaj:<?php echo $userstr; ?> </a></li>
    <li><a href="blog.php" >blog</a></li>
    <li><a href="users.php" >użytkownicy</a></li>
    <li><a href="logout.php" >wyloguj</a></li>
</ul>
</div>
<div class="panel-body">
```

### 7.3.4. informacje o użytkownikach

```
require_once 'menu_user.php';

foreach ($rows as $row) : ?>
    <div id ="blogItem">
        <h3><?php echo $row['Login']; ?></h3>

        <?php if ($row['IsAdmin']) : ?>
            <p>ROLA: ADMIN</p>
        <?php else : ?>
            <p>ROLA: USER</p>
        <?php endif; ?>

        <p>WPISÓW: <?php echo $row['Blogs']; ?> </p>
        <p>KOMENTARZY: <?php echo $row['Comments']; ?></p>
        <p>DOŁĄCZYŁ: <?php echo $row['X_CreateTime']; ?></p>
        <p>OSTATNIA AKTYWNOŚĆ: <?php echo $row['LastActivity']; ?></p>
    </div>
    <br>
<?php endforeach; ?>

<?php endif; ?>
```

**7.3.5. zarządzanie uprawnieniami**

```

if ($_SESSION['IsAdmin']) :

    require_once 'menu_admin.php';

foreach ($rows as $row) : ?>
    <div id ="blogItem">
    <h3><?php echo $row['Login']; ?></h3>
    <p>@mail: <?php echo $row['Email']; ?> </p>

        <?php if ($row['IsAdmin']) : ?>
                                <p>ROLA: ADMIN</p>
        <?php else : ?>
                                <p>ROLA: USER</p>
        <?php endif; ?>

    <p>WPISÓW: <?php echo $row['Blogs']; ?> </p>
    <p>KOMENTARZY: <?php echo $row['Comments']; ?></p>
    <p>DOŁĄCZYŁ: <?php echo $row['X_CreateTime']; ?></p>
    <p>OSTATNIA AKTYWNOŚĆ: <?php echo $row['LastActivity']; ?></p>

        <?php if ($row['IsBlocked']) : ?>
        <h3>KONTO ZABLOKOWANE</h3>
        <?php endif; ?>

    <form method="post" action="<?php $_SERVER['PHP_SELF']; ?>" >
    <ul class="nav navbar-nav navbar-right">
    <li>
    <input type="hidden" name="privileges_id" value="<?php echo $row['user_id']; ?>"/>
    <input type="hidden" name="isadmin" value="<?php echo $row['IsAdmin']; ?>"/>
    <input type="hidden" name="isadmin" value="<?php echo $row['IsAdmin']; ?>"/>
    <input type="submit" name="privileges" value="Zmień Rolę"/>
    </li>
    <li>
    <input type="hidden" name="block_id" value="<?php echo $row['user_id']; ?>"/>
    <input type="hidden" name="isblocked" value="<?php echo $row['IsBlocked']; ?>"/>
    <input type="submit" name="block" value="Zablokuj/Odblokuj"/>
    </li></ul>
    </form>
    <br>
    </div></br>
<?php endforeach; ?>

```

Wykoanie akcji dla zmiennej post w pliku *blog\_src.php*.

```

elseif($post['privileges'])
{
    $user_id = $post['privileges_id'];
    $isadmin = $post['isadmin'];

    if ($isadmin)
    {
        $isadmin = false;
    }
    else
    {

```

```

        $isadmin = true;
    }

    $connect->query("UPDATE User SET IsAdmin = :isadmin WHERE ID =:id");
    $connect->bindValue(':isadmin', $isadmin, PDO::PARAM_BOOL);
    $connect->bindValue(':id', $user_id, PDO::PARAM_STR);
    $connect->execute();

```

#### 7.3.6. blokowanie konta użytkownika

```

elseif($post['block'])
{
    $user_id = $post['block_id'];
    $isblocked = $post['isblocked'];

    if ($isblocked)
    {
        $isblocked = false;
    }
    else
    {
        $isblocked = true;
    }

    $connect->query("UPDATE User SET IsBlocked = :isblocked WHERE ID =:id");
    $connect->bindValue(':isblocked', $isblocked, PDO::PARAM_BOOL);
    $connect->bindValue(':id', $user_id, PDO::PARAM_STR);
    $connect->execute();
}

```

### 7.4. WPISY

#### 7.4.1. nowy wpis

```

if($post['add'])
{
    $title = $post['title'];
    $body = $post['body'];
    $user_id = $_SESSION['ID'];

    $connect->query(
        'INSERT INTO blogtext (ID, Title, Text, X_CreateUser)
        VALUES (uuid(), :title, :body, :user)');
    $connect->bindValue(':title', $title, PDO::PARAM_STR);
    $connect->bindValue(':body', $body, PDO::PARAM_STR);
    $connect->bindValue(':user', $user_id, PDO::PARAM_STR);
    $connect->execute();
}

```

#### 7.4.2. edycja wpisu

```

elseif($post['edit'])
{
    $edit_id = $post['edit_id'];
    header("Location: blog_edit.php?edit=$edit_id");
}

```



```
}

```

#### 7.4.3. zmiana statusu

```
elseif($post['change_status'])
{
    $change_id;
    $status;
    if ($post['hide_id'])
    {
        $change_id = $post['hide_id'];
        $status = 0;
    }
    else
    {
        $change_id = $post['show_id'];
        $status = 1;
    }
    $connect->query('UPDATE blogtext SET Status = :status WHERE ID =:id');
    $connect->bindValue(':id', $chane_id, PDO::PARAM_STR);
    $connect->bindValue(':status', $status, PDO::PARAM_INT);
    $connect->execute();
}
```

#### 7.4.4. wyświetlanie wpisów i komentarzy

Wyświetlanie realizuje zagnieżdżona pętla *foreach*.

```
<?php foreach ($rows_show as $row) : ?>
<div id="blogItem">

    <h3><?php echo $row['Title']; ?></h3>
    <h10><?php echo $row['Login'].", ".$row['X_UpdateTime']; ?></h10>
    <p><?php echo $row['Text']; ?></p>
    <p><a href="login.php"> + Dodaj komentarz</a><p>

        <?php $connect->query(
            'SELECT * FROM all_comments
            WHERE BlogItemID = :blog_id');

        $connect->bindValue(':blog_id', $row['blogtext_id'], PDO::PARAM_STR);
        $comments = $connect->resultSet();

        if ($comments) : ?>

            <?php foreach ($comments as $row) : ?>
                <div id="comment">
                    <p><?php echo $row['Login'].", ".$row['Time']; ?></p>
                    <p><?php echo $row['Text']; ?></p>
                </div>
            <?php endforeach;
```

```
endif; ?>

</div>
</br></br>
<?php endforeach;?>
```

#### 7.4.5. wyświetlanie archiwum wpisów

Za dostęp do archiwum wpisów odpowiada plik *archive.php*. Jeśli użytkownik jest zalogowany otrzymuje dostęp do wpisów, w przeciwnym wypadku (jeśli np. przypadkowo dostał się na stronę *archive.php* zostaje przekierowany do widoku bloga z wykorzystaniem funkcji *header*. Zawartość pliku *archive.php*:

```
<?php require_once 'header.php';
require_once 'source/blog_src.php';

    if ($loggedin)
    {
        require_once 'source/archive_src.php';
        require_once 'footer.php';
    }
    else
        header("Location: blog.php");
```

Zawartość pliku *archive\_src.php*:

```
<?php
    $connect->query(
        'SELECT * FROM archive');
    $rows_rem = $connect->resultset();
    ?>

    <div class="row">
    <div class="col-sm-12">
    <div class="panel panel-primary">
    <div class="panel-heading">ARCHIWUM WPISÓW</div>
    <div class="panel-body">

<?php if ($rows_rem) :
foreach ($rows_rem as $row) : ?>
            <div>
                <h3><?php echo $row['Title']; ?></h3>
<h10><?php echo $row['Login'].", ".$row['X_UpdateTime'].", usunięto: ".$row['X_RemoveTime']; ?></h10>
                <p><?php echo $row['Text']; ?></p>
                <form method="post" action="<?php $_SERVER['PHP_SELF']; ?>" >
                    <ul class="nav navbar-nav navbar-right">
                        <li>
                            <input type="hidden" name="restore_id" value="<?php echo $row['blogtext_id']; ?>" />
                            <input type="submit" name="restore" value="Przywróć"/>
                        </li>
                        <li>
                            <input type="hidden" name="delete_id" value="<?php echo $row['blogtext_id']; ?>" />
                            <input type="submit" name="delete" value="Usuń"/>
                        </li>
                        <li>
                            <input type="submit" name="abort" value="Powrót do wpisów"/>

```

```

        </li>
    </ul>

    </div>

    <?php endforeach;
    else : ?>
    <h2>BRAK WPISÓW</h2>

<?php endif; ?>

```

#### 7.4.6. przenoszenie do archiwum wpisów

```

elseif($post['archive'])
{
    $archive_id = $post['archive_id'];
    $connect->query('UPDATE blogtext SET X_RemoveTime = now()
                    WHERE ID =:id');
    $connect->bindValue(':id', $archive_id, PDO::PARAM_STR);
    $connect->execute();
}

```

#### 7.4.7. przywracanie z archiwum wpisów

```

elseif($post['restore'])
{
    $restore_id = $post['restore_id'];
    $connect->query('UPDATE blogtext SET X_RemoveTime = NULL
                    WHERE ID =:id');
    $connect->bindValue(':id', $restore_id, PDO::PARAM_STR);
    $connect->execute();
}

```

#### 7.4.8. usuwanie wpisów

```

elseif($post['delete'])
{
    $delete_id = $post['delete_id'];
    //echo $delete_id = $post['delete_id'];
    $connect->query('DELETE FROM blogtext WHERE ID =:id');
    $connect->bindValue(':id', $delete_id, PDO::PARAM_STR);
    $connect->execute();
}

```

## 7.5. KOMENTARZE

### 7.5.1. dodawanie komentarzy

```
elseif($post['comment'])
{
    $comment_id = $post['comment_id'];
    header("Location: comment.php?com=$comment_id");
}
```

### 7.5.2.usuwanie komentarzy

```
elseif($post['delcom'])
{
    $delete_id = $post['del_id'];
    //echo $delete_id = $post['delete_id'];
    $connect->query('DELETE FROM blogcomment WHERE ID =:id');
    $connect->bindValue(':id', $delete_id, PDO::PARAM_STR);
    $connect->execute();
}
```

## 8. STRUKTURA BAZY DANYCH

### 8.1. OPIS

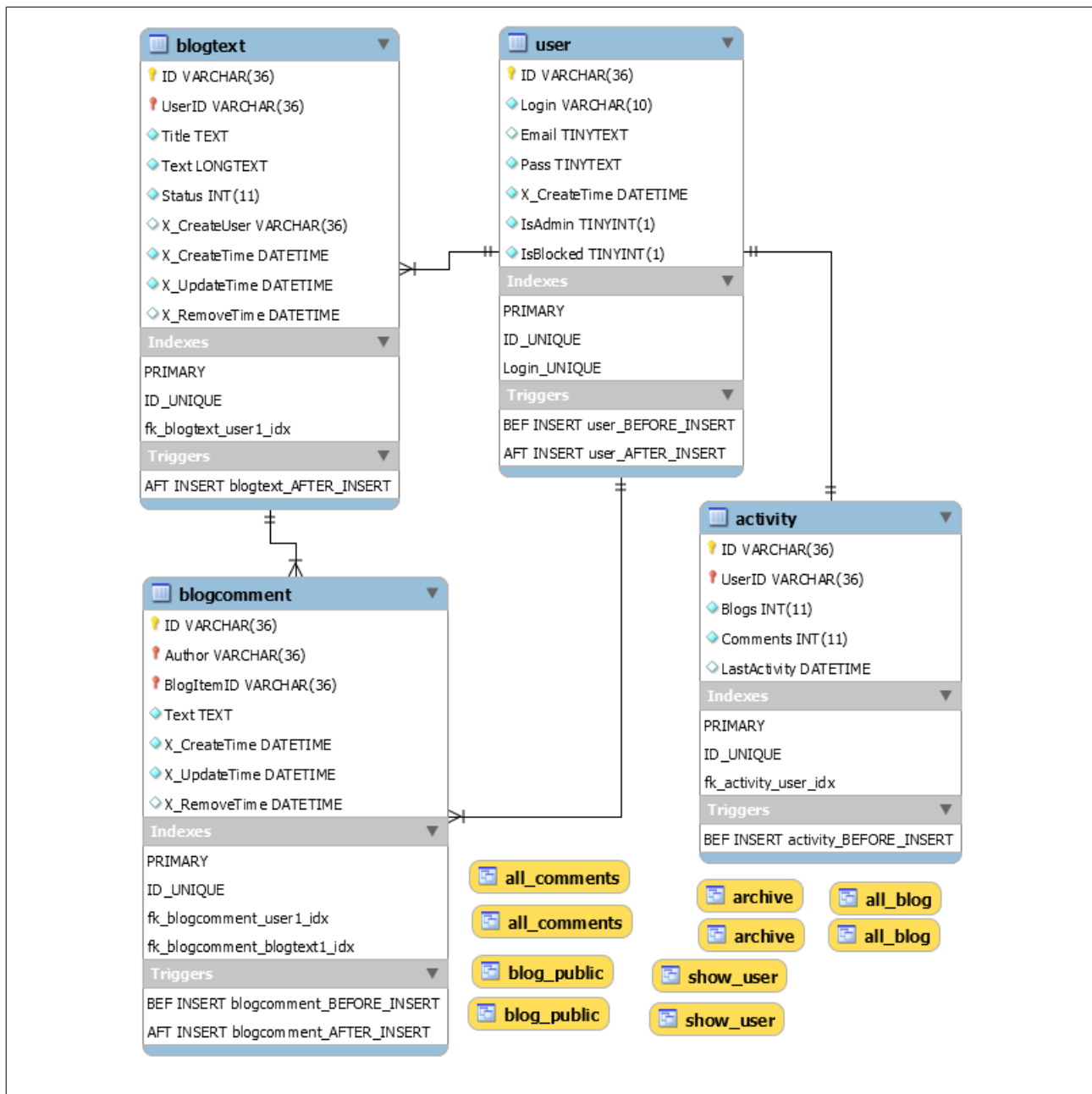
Relacyjna baza danych złożona z 4 tabel: *user*, *blogtext*, *blogcomment* i *activity*.

Relacje:

- jeden-do-jeden: *user.UserId* – *activity.UserID*
- jeden-do-wielu: *user.UserId* – *blogtext.UserID*
- jeden-do-wielu: *user.UserId* – *blogcomment.Author*
- jeden-do-wielu: *blogtext.ID* – *blogcomment.BlogItemD*

Dla bazy danych zdefiniowano 5 widoków.

### 8.2. DIAGRAM ERR



### 8.3. TABELE

#### 8.3.1. tabela *user*

Tabela z danymi o użytkownikach, uprawnieniach i blokadzie konta.

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
ID	VARCHAR(36)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
Login	VARCHAR(10)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
Email	TINYTEXT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
Pass	TINYTEXT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
X_CreateTime	DATETIME	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	CURRENT_TIMESTAMP
IsAdmin	TINYINT(1)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	'0'
IsBlocked	TINYINT(1)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	'0'
		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

### 8.3.2. tabela *blogtext*

Tabela z danymi o wpisach, statusie, i przeniesieniu do archiwum (kolumna X\_RemoveTime o wartość NOT NULL).

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
ID	VARCHAR(36)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
Title	TEXT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
Text	LONGTEXT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
Status	INT(11)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	'0'
X_CreateUser	VARCHAR(36)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
X_CreateTime	DATETIME	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	CURRENT_TIMESTAMP
X_UpdateTime	DATETIME	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	CURRENT_TIMESTAMP
X_RemoveTime	DATETIME	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

### 8.3.3. tabela *blogcommnet*

Tabela z danymi o komentarzach.

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
ID	VARCHAR(36)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
BlogItemID	VARCHAR(36)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
Author	VARCHAR(36)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
Text	TEXT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
X_CreateTime	DATETIME	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	CURRENT_TIMESTAMP
X_UpdateTime	DATETIME	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	CURRENT_TIMESTAMP
X_RemoveTime	DATETIME	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

### 8.3.4. tabela *activity*

Tabela z danymi o aktywności użytkowników. Wartości w kolumnach tabeli dodają i zmieniają się w wyniku operacji na tabelach *user*, *blogtext* i *blogcommnet*:

- kolumny *ID* i *UserId* po operacji *insert* dla tabeli *user* (wyzwalacz, pkt. 8.4.2)
- kolumna *Blogs* po operacji *insert* dla tabeli *blogtext* (wyzwalacz, pkt. 8.4.4)
- kolumna *Commnets* po operacji *insert* dla tabeli *blogcommnet* (wyzwalacz, pkt. 8.4.6)

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
ID	VARCHAR(36)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	"
UserID	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	'UUID'
Blogs	INT(11)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	'0'
Comments	INT(11)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	'0'
LastActivity	DATETIME	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

## 8.4. WYZWALACZE

### 8.4.1. tabela user [ before insert ]



Wyzwalacz generuje wartości UUID() dla kolumny ID (klucz główny) nowego wiersza tabeli *user*.

```
CREATE DEFINER=`root`@`localhost` TRIGGER `sdbm`.`user_BEFORE_INSERT` BEFORE
INSERT ON `user` FOR EACH ROW
BEGIN
SET new.ID = uuid();
END
```

#### 8.4.2. tabela user [ after insert ]

Wyzwalacz generuje wartości UUID() dla nowego wiersza tabeli *activity* w przypadku operacji *insert* (dodanie nowego użytkownika).

```
CREATE DEFINER=`root`@`localhost` TRIGGER `sdbm`.`user_AFTER_INSERT` AFTER
INSERT ON `user` FOR EACH ROW
BEGIN

INSERT INTO `sdbm`.`activity`
(`UserID`)
VALUES
(NEW.ID);

END
```

#### 8.4.3. tabela blogtext [ before insert ]

Wyzwalacz generuje wartości UUID() dla kolumny ID (klucz główny) nowego wiersza tabeli *blogtext*.

```
CREATE DEFINER=`root`@`localhost` TRIGGER `sdbm`.`blogcomment_BEFORE_INSERT`
BEFORE INSERT ON `blogcomment` FOR EACH ROW

BEGIN

SET new.ID = uuid();

END
```

#### 8.4.4. tabela blogtext [ after insert ]

Wyzwalacz inkrementuje aktualną wartość kolumny *Blogs* z tabeli *activity* w przypadku operacji *insert* w tabeli *blogtext*. Oznacza to że dodanie wpisu zwiększa aktualną liczbę wpisów użytkownika o 1.

```
CREATE DEFINER=`root`@`localhost` TRIGGER `sdbm`.`blogtext_AFTER_INSERT`
AFTER INSERT ON `blogtext` FOR EACH ROW
BEGIN

DECLARE x INT;

SET x = (SELECT Blogs FROM Activity WHERE UserID = NEW.X_CreateUser);

UPDATE Activity SET Activity.Blogs = x+1, LastActivity = now()
WHERE UserID = NEW.X_CreateUser;

END
```

#### 8.4.5. tabela blogcommnet [ before insert ]

Wyzwalacz generuje wartości UUID() dla kolumny *ID* (klucz główny) nowego wiersza tabeli *blogcomment*.

```
CREATE DEFINER=`root`@`localhost` TRIGGER `sdbm`.`blogcomment_BEFORE_INSERT`
BEFORE INSERT ON `blogcomment` FOR EACH ROW
BEGIN
SET new.ID = uuid();
END
```

#### 8.4.6. tabela *blogcomment* [ after insert ]

Wyzwalacz inkrementuje aktualną wartość kolumny *Comments* z tabeli *activity* w przypadku operacji *insert* w tabeli *blogcomment*. Oznacza to że dodanie komentarzy zwiększa aktualną liczbę komentarzy użytkownika o 1.

```
CREATE DEFINER=`root`@`localhost` TRIGGER `sdbm`.`blogcomment_AFTER_INSERT`
AFTER INSERT ON `blogcomment` FOR EACH ROW
BEGIN

DECLARE x INT;

SET x = (SELECT Comments FROM Activity WHERE UserID = NEW.X_CreateUser);

UPDATE Activity SET Activity.Comments = x+1, LastActivity = now()
WHERE UserID = NEW.Author;

END
```

#### 8.4.7. tabela *activity* [ before insert ]

Wyzwalacz generuje wartości UUID() dla kolumny *ID* (klucz główny) dla nowego wiersza tabeli *activity*.

```
CREATE DEFINER=`root`@`localhost` TRIGGER `sdbm`.`activity_BEFORE_INSERT`
BEFORE INSERT ON `activity` FOR EACH ROW
BEGIN
SET new.ID = uuid();
END
```

## 8.5. WIDOKI

**8.5.1. widok: all\_blog**

Widok wyświetla wybrane kolumny z dwóch połączonych tabel: *blogtext* i *user* uporządkowane według daty dodania malejąco. Jest wykorzystywany do wyświetlania wpisów, które nie zostały przeniesione do archiwum wpisów.

```
CREATE
  ALGORITHM = UNDEFINED
  DEFINER = `root`@`localhost`
  SQL SECURITY DEFINER
VIEW `all_blog` AS
  SELECT
    `blogtext`.`Title` AS `Title`,
    `blogtext`.`Status` AS `Status`,
    `user`.`Login` AS `Login`,
    `blogtext`.`Text` AS `Text`,
    `blogtext`.`X_UpdateTime` AS `X_UpdateTime`,
    `blogtext`.`X_RemoveTime` AS `X_RemoveTime`,
    `blogtext`.`X_CreateUser` AS `X_CreateUser`,
    `blogtext`.`ID` AS `blogtext_id`
  FROM
    (`blogtext`
    JOIN `user` ON ((`blogtext`.`X_CreateUser` = `user`.`ID`)))
  WHERE
    ISNULL(`blogtext`.`X_RemoveTime`)
  ORDER BY `blogtext`.`X_UpdateTime` DESC
```

**8.5.2. widok: blog\_public**

Widok wyświetla rekordy z widoku all\_blog dla których kolumna status = 1. Jest wykorzystywany do wyświetlania wpisów ze statusem „aktywny” dla niezalogowanego użytkownika.

```
CREATE
  ALGORITHM = UNDEFINED
  DEFINER = `root`@`localhost`
  SQL SECURITY DEFINER
VIEW `blog_public` AS
  SELECT
    `all_blog`.`Title` AS `Title`,
    `all_blog`.`Status` AS `Status`,
    `all_blog`.`Login` AS `Login`,
    `all_blog`.`Text` AS `Text`,
    `all_blog`.`X_UpdateTime` AS `X_UpdateTime`,
    `all_blog`.`X_RemoveTime` AS `X_RemoveTime`,
    `all_blog`.`X_CreateUser` AS `X_CreateUser`,
    `all_blog`.`blogtext_id` AS `blogtext_id`
  FROM
    `all_blog`
  WHERE
    ((`all_blog`.`Status` = 1))
```

**8.5.3. widok: all\_comments**

Widok wyświetla wybrane kolumny z dwóch połączonych tabel: *blogcomment* i *user* uporządkowane według daty dodania malejąco. Jest wykorzystywany do wyświetlania wpisów ze statusem „aktywny” dla niezalogowanego użytkownika.

```
CREATE
  ALGORITHM = UNDEFINED
  DEFINER = `root`@`localhost`
  SQL SECURITY DEFINER
VIEW `all_comments` AS
  SELECT
    `blogcomment`.`BlogItemID` AS `BlogItemID`,
    `blogcomment`.`Author` AS `Author`,
    `blogcomment`.`Text` AS `Text`,
    `blogcomment`.`X_CreateTime` AS `Time`,
    `user`.`Login` AS `Login`
  FROM
    (`blogcomment`
    JOIN `user` ON ((`blogcomment`.`Author` = `user`.`ID`)))
  ORDER BY `blogcomment`.`X_CreateTime` DESC
```

#### 8.5.4. widok: archive

Widok wyświetla wybrane kolumny z dwóch połączonych tabel: *blogtext* i *user*, dla których kolumna *X\_RemoveTime* jest różna od NULL, uporządkowane według daty dodania malejąco. Jest wykorzystywany do wyświetlania archiwum wpisów.

```
CREATE
  ALGORITHM = UNDEFINED
  DEFINER = `root`@`localhost`
  SQL SECURITY DEFINER
VIEW `archive` AS
  SELECT
    `blogtext`.`Title` AS `Title`,
    `blogtext`.`Status` AS `Status`,
    `user`.`Login` AS `Login`,
    `blogtext`.`Text` AS `Text`,
    `blogtext`.`X_UpdateTime` AS `X_UpdateTime`,
    `blogtext`.`X_RemoveTime` AS `X_RemoveTime`,
    `blogtext`.`X_CreateUser` AS `X_CreateUser`,
    `blogtext`.`ID` AS `blogtext_id`
  FROM
    (`blogtext`
    JOIN `user` ON ((`blogtext`.`X_CreateUser` = `user`.`ID`)))
  WHERE
    (`blogtext`.`X_RemoveTime` IS NOT NULL)
  ORDER BY `blogtext`.`Status` DESC
```

#### 8.5.5. widok: show\_user

Widok wyświetla wybrane kolumny z tabeli *user*. Jest wykorzystywany w trakcie logowania użytkownika.

```
CREATE
  ALGORITHM = UNDEFINED
  DEFINER = `root`@`localhost`
  SQL SECURITY DEFINER
VIEW `show_user` AS
  SELECT
    `user`.`ID` AS `ID`,
    `user`.`Login` AS `Login`,
    `user`.`Pass` AS `Pass`,
    `user`.`IsAdmin` AS `IsAdmin`,
    `user`.`IsBlocked` AS `IsBlocked`
  FROM
    `user`
```

1. *Learn Object Oriented PHP By Building a Complete Website*, <https://www.udemy.com/learn-object-oriented-php-by-building-a-complete-website/learn/v4/overview>, Eduonix Learning Solutions, data dostępu: 2016-03-11 – 2016-05-11
2. Nixon R.: *PHP, MYSQL I JavaScript Wprowadzenie Przewodnik twórcy stron i aplikacji sieciowych*, O'REILY, Helion 2015, Gliwice, Polska.
3. Forta B.: *SQL w mgnieniu oka wyd. IV*, Helion