

**Bazy danych
2IS P4
Państwowa Wyższa Szkoła Zawodowa
w Nowym Sączu**

Tytuł projektu: Projekt wspomagający system działalności firmy świadczącej usługi gastronomiczne dla klientów indywidualnych oraz firm.

Temat roboczy: Usługi gastronomiczne.

Skład grupy: Adam Szczepański, Patryk Matusik, Adrian Świąs

Prowadzący: mgr inż. Nikodem Bulanda

Spis treści

1. Cel.....	3
2. Wymagania.....	3
a) Wymagania funkcjonalne.....	3
b) Wymagania нефункционалне.....	3
3. Opis stosowanych technologii	4
4. Diagram UML.....	5
5. Scenariusze.....	6
6. Lista zadań oraz estymacja czasowa.....	8
7. Nazwy metod służących do zbudowania zapytania RAW oraz zapytania z wykorzystaniem ORM / QB.....	9
8. Link do repozytorium.....	17
9. Bilans.....	17
10. Wnioski.....	17
11. Bibliografia.....	18

1. Cel.

Projekt dotyczy systemu wspomagania działalności firmy świadczącej usługi gastronomiczne dla klientów indywidualnych oraz firm. System winien być możliwy do równoległego użytkowania przez kilka tego typu firm. Manager restauracji będzie mógł w prosty i szybki sposób koordynować zarządzaniem restauracji oraz dopilnować, aby żadna rzecz nie została pominięta, co mogłoby prowadzić do zaniżenia renomy restauracji.

2. Wymagania.

a) Wymagania funkcjonalne

- System zamawiania jedzenia
- Prosty system naliczania rabatów
- System wystawiania raportów
- System wyliczania dochodów w danym miesiącu
- Możliwość wcześniejszej rezerwacji stolika

b) Wymagania нефunkcjonalne

- możliwość ustalania menu
- system umożliwiać będzie generowanie raportów miesięcznych,
- system umożliwiać będzie realizację programów rabatowych dla klientów indywidualnych oraz firm.
- system umożliwiać będzie wprowadzenie kategorii danego produktu (np. dania, bądź napoje).

3. Opis stosowanych technologii

MySQL- wolnodostępny, otwartoźródłowy system zarządzania relacyjnymi bazami danych.

PHP- interpretowany, skryptowy język programowania zaprojektowany do generowania stron internetowych i budowania aplikacji webowych w czasie rzeczywistym.

Codeigniter- framework napisany w języku PHP przez Ricka Ellisa, implementujący wzorec Model-View-Controller. Celem projektu jest przygotowanie zestawu narzędzi dla osób, które budują aplikacje internetowe za pomocą PHP, aby umożliwić rozwój projektów znacznie szybciej niż pisanie kodu od podstaw, poprzez bogaty zestaw bibliotek dla najczęściej potrzebnych zadań, jak również prosty interfejs i logiczną strukturę dostępu do tych bibliotek. CodeIgniter pozwala twórczo skupić się na projekcie, minimalizując ilość kodu potrzebnego dla danego zadania.

HTML-hipertekstowy język znaczników, wykorzystywany do tworzenia dokumentów hipertekstowych. HTML pozwala opisać strukturę informacji zawartych wewnątrz strony internetowej, nadając odpowiednie znaczenie semantyczne poszczególnym fragmentom tekstu – formując hiperłącza, akapity, nagłówki, listy – oraz osadza w tekście dokumentu obiekty plikowe np. multimedia bądź elementy baz danych np. interaktywne formularze danych. HTML umożliwia określenie wyglądu dokumentu w przeglądarce internetowej. Do szczegółowego opisu formatowania akapitów, nagłówków, użytych czcionek i kolorów, zalecane jest wykorzystywanie kaskadowych arkuszy stylów.

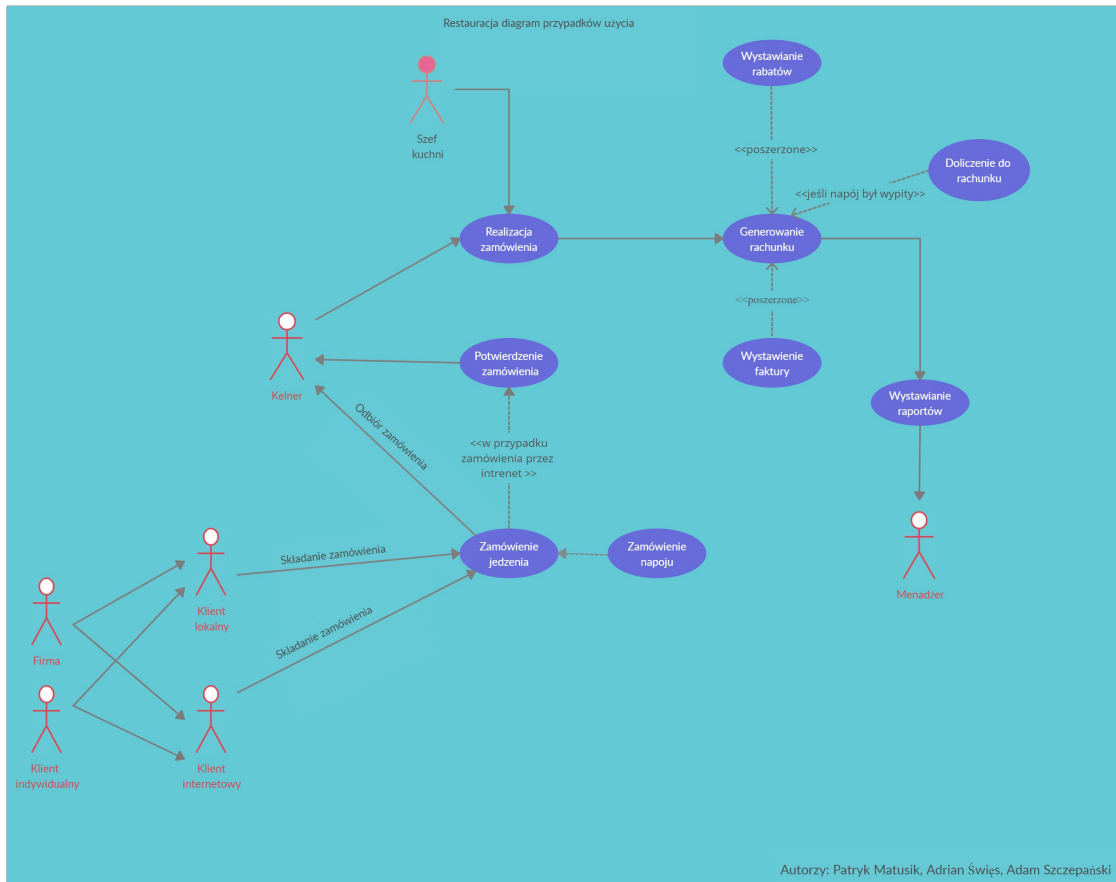
Modal Box-jest nałożona na podformularzu formularza nadrzędnego. Ogólnie, celem jest pokazanie treści z jednego źródła i może mieć pewne oddziaływanie bez wychodzenia z formularza nadrzędnego. Podformularz może dostarczyć informacji, interakcji i tak dalej.

Ajax- technika tworzenia aplikacji internetowych, w których interakcja użytkownika z serwerem odbywa się bez przeładowywania całego dokumentu, w sposób asynchroniczny.

Bootstrap- biblioteka CSS, zawiera zestaw przydatnych narzędzi ułatwiających tworzenie interfejsu graficznego stron oraz aplikacji internetowych. Bazuje głównie na gotowych rozwiązaniach HTML oraz CSS (kompilowanych z plików Less) i może być stosowany m.in. do stylizacji takich elementów jak teksty, formularze, przyciski, wykresy, nawigacje i innych komponentów wyświetlanych na stronie. Biblioteka korzysta także z języka JavaScript.

Jquery - lekka biblioteka programistyczna dla języka JavaScript, ułatwiająca korzystanie z JavaScriptu (w tym manipulację drzewem DOM). Kosztem niewielkiego spadku wydajności w stosunku do profesjonalnie napisanego kodu w niewspomagany JavaScript pozwala osiągnąć interesujące efekty animacji, dodać dynamiczne zmiany strony, wykonać zapytania AJAX. Większość wtyczek i skryptów opartych na jQuery działa na stronach nie wymagając zmian w kodzie HTML (np. zamienia klasyczne galerie złożone z miniatur linkujących do obrazków w dynamiczną galerię).

4. Diagram UML



5. Scenariusze

Scenariusz nr:	1
Tytuł:	Zamówienie jedzenia
Aktor:	Klient indywidualny-lokalny, Kelner, Szef kuchni, Menadżer
Warunek:	Złożenie zamówienia w restauracji
Opis/Przebieg:	1.Kelner prosi o podanie zamówienia 2.Złożenie zamówienia przez klienta 3.Przekazanie zamówienia do realizacji 4.Realizacja zamówienia 5. Wystawianie rabatów 6.Generowanie rachunku 7.Wystawienie faktury na prośbę klienta 8.Wystawienie raportu
Zakończenie:	Zapłata za zamówienie, wyrażenie opinii o jakości zamówienia
Zakończenie alternatywne	Klient anuluje zamówienie i opuszcza restaurację

Scenariusz nr:	2
Tytuł:	Zamówienie jedzenia oraz napoju

Aktor:	Klient indywidualny-lokalny, Kelner, Szef kuchni, Menadżer
Warunek:	Złożenie zamówienia w restauracji
Opis/Przebieg:	<ol style="list-style-type: none"> 1.Kelner prosi o podanie zamówienia 2.Złożenie zamówienia przez klienta 3.Przekazanie zamówienia do realizacji 4.Podanie napoju założonego w zamówieniu 5.Realizacja zamówienia 6. Wystawianie rabatów 7.Generowanie rachunku 8. Wystawienie faktury na prośbę klienta 9.Wystawienie raportu
Zakończenie:	Zapłata za zamówienie, wyrażenie opinii o jakości zamówienia
Zakończenie alternatywne	Klient anuluje zamówienie i opuszcza restaurację lub klient płaci za napój i anuluje zamówienie jedzenia

Scenariusz nr:	3
Tytuł:	Zamówienie jedzenia
Aktor:	Klient lokalny, Kelner, Szef kuchni, Menadżer
Warunek:	Złożenie zamówienia w restauracji
Opis/Przebieg:	<ol style="list-style-type: none"> 1.Kelner prosi o podanie zamówienia 2.Podanie zamówienia przez klienta 3.Przekazanie zamówienia do realizacji 4.Realizacja zamówienia 5. Wystawianie rabatów 6.Generowanie rachunku 7.Wystawienie faktury na prośbę klienta 8.Wystawienie raportu
Zakończenie:	Zapłata za zamówienie, wyrażenie opinii o jakości zamówienia
Zakończenie alternatywne	Klient anuluje zamówienie i opuszcza restaurację

Scenariusz nr:	4
Tytuł:	Zamówienie jedzenia
Aktor:	Klient lokalny, Kelner, Szef kuchni, Menadżer
Warunek:	Złożenie zamówienia w restauracji
Opis/Przebieg:	<ol style="list-style-type: none"> 1.Kelner prosi o podanie zamówienia 2.Podanie zamówienia przez klienta 3.Przekazanie zamówienia do realizacji 4.Klient prosi o zmianę zamówienia. 5.Ponowne przekazanie zamówienia do realizacji 6.Realizacja zamówienia 7. Wystawianie rabatów 8.Generowanie rachunku
Zakończenie:	Zapłata za zamówienie
Zakończenie alternatywne	Klient anuluje zamówienie i opuszcza restaurację

Scenariusz nr:	5
Tytuł:	Zamówienie jedzenia
Aktor:	Klient, Kelner, Szef kuchni, Menadżer
Warunek:	Złożenie zamówienia w restauracji
Opis/Przebieg:	1.Kelner prosi o podanie zamówienia 2.Podanie zamówienia przez klienta 3.Przekazanie zamówienia do realizacji 4.Klient prosi o zmianę stolika oraz zmianę zamówienia. 5.Kelner przekazuje informację menadżerowi. 6.Realizacja zamówienia 7.Wystawianie rabatów 8.Generowanie rachunku
Zakończenie:	Zapłata za zamówienie
Zakończenie alternatywne	Klient anuluje zamówienie i opuszcza restaurację

6. Lista zadań oraz estymacja czasowa

1. Budowa podstawowej strony internetowej - (1 dzień)
2. Budowa bazy danych oraz przystosowanie jej do założeń projektowych - (3 dni)
3. Utworzenie połączenia między bazą danych a stroną internetową - (4 dni)
4. Utworzenie panelu klienta - (12 dni)
5. Utworzenie panelu administratora – (13dni)
6. Stworzenie zbioru konfiguracji do wyboru oraz opracowanie ich zapisu do zamówienia – (7 dni)
7. Zaprojektowanie graficznego wyglądu dań oraz ich wyboru - (6 dni)
8. Opracowanie generowania pliku PDF zawierającego raport zamówienia - (2 dni)
9. Zadbanie o responsywność strony (3 dni)

Sprint 1 (8 dni)

1. Budowa podstawowej strony internetowej - (1 dzień)
2. Budowa bazy danych oraz przystosowanie jej do założeń projektowych - (3 dni)
3. Utworzenie połączenia między bazą danych a stroną internetową - (4 dni)

Sprint 2 (32 dni)

1. Utworzenie panelu klienta - (12 dni)
2. Utworzenie panelu administratora – (13dni)
3. Stworzenie zbioru konfiguracji do wyboru oraz opracowanie ich zapisu do zamówienia – (7 dni)

Sprint 3 (11 dni)

1. Zaprojektowanie graficznego wyglądu dań oraz ich wyboru - (6 dni)
2. Opracowanie generowania pliku PDF zawierającego raport zamówienia - (2 dni)
3. Zadbanie o responsywność strony (3 dni)

7. Nazwy metod służących do zbudowania zapytania RAW oraz zapytania z wykorzystaniem ORM / QB.

```
<?php
class Model_category extends CI_Model
{
    public function __construct()
    {
        parent::__construct();
    }
    public function getCategoryData($id = null)
    {
        if($id) {
            $sql = "SELECT * FROM category WHERE id = ?";
            $query = $this->db->query($sql, array($id));
            return $query->row_array();
        }

        $sql = "SELECT * FROM category ORDER BY id DESC";
        $query = $this->db->query($sql);
        return $query->result_array();
    }
    public function create($data = array())
    {
        if($data) {
            $create = $this->db->insert('category', $data);
            return ($create == true) ? true : false;
        }
    }
    public function update($id = null, $data = array())
    {
        if($id && $data) {
            $this->db->where('id', $id);
            $update = $this->db->update('category', $data);
            return ($update == true) ? true : false;
        }
    }
    public function remove($id = null)
    {
        if($id) {
            $this->db->where('id', $id);
            $delete = $this->db->delete('category');
            return ($delete == true) ? true : false;
        }
    }
    public function getActiveCategory()
    {
        $sql = "SELECT * FROM category WHERE active = ?";
        $query = $this->db->query($sql, array(1));
        return $query->result_array();
    }
}
```



```

<?php
class Model_auth extends CI_Model
{
    public function __construct()
    {
        parent::__construct();
    }
    public function check_email($email)
    {
        if($email) {
            $sql = 'SELECT * FROM users WHERE email = ?';
            $query = $this->db->query($sql, array($email));
            $result = $query->num_rows();
            return ($result == 1) ? true : false;
        }

        return false;
    }
    public function login($email, $password) {
        if($email && $password) {
            $sql = "SELECT * FROM users WHERE email = ?";
            $query = $this->db->query($sql, array($email));

            if($query->num_rows() == 1) {
                $result = $query->row_array();

                $hash_password = password_verify($password, $result['password']);
                if($hash_password === true) {
                    return $result;
                }
                else {
                    return false;
                }
            }
            else {
                return false;
            }
        }
    }
}
)

```

```

<?php
class Model_company extends CI_Model
{
    public function __construct()
    {
        parent::__construct();
    }
    public function getCompanyData($id = null)
    {
        if($id) {
            $sql = "SELECT * FROM company WHERE id = ?";
            $query = $this->db->query($sql, array($id));
            return $query->row_array();
        }
    }
    public function update($data, $id)
    {
        if($data && $id) {
            $this->db->where('id', $id);
            $update = $this->db->update('company', $data);
            return ($update == true) ? true : false;
        }
    }
}
)

```

```

<?php
class Model_groups extends CI_Model
{
    public function __construct()
    {
        parent::__construct();
    }

    public function getGroupData($groupId = null)
    {
        if($groupId) {
            $sql = "SELECT * FROM groups WHERE id = ?";
            $query = $this->db->query($sql, array($groupId));
            return $query->row_array();
        }
        $sql = "SELECT * FROM groups WHERE id != ? ORDER BY id DESC";
        $query = $this->db->query($sql, array(1));
        return $query->result_array();
    }

    public function create($data = '')
    {
        $create = $this->db->insert('groups', $data);
        return ($create == true) ? true : false;
    }

    public function edit($data, $id)
    {
        $this->db->where('id', $id);
        $update = $this->db->update('groups', $data);
        return ($update == true) ? true : false;
    }

    public function delete($id)
    {
        $this->db->where('id', $id);
        $delete = $this->db->delete('groups');
        return ($delete == true) ? true : false;
    }

    public function existInUserGroup($id)
    {
        $sql = "SELECT * FROM user_group WHERE group_id = ?";
        $query = $this->db->query($sql, array($id));
        return ($query->num_rows() == 1) ? true : false;
    }

    public function getUserGroupByUserId($user_id)
    {
        $sql = "SELECT * FROM user_group
        INNER JOIN groups ON groups.id = user_group.group_id
        WHERE user_group.user_id = ?";
        $query = $this->db->query($sql, array($user_id));
        $result = $query->row_array();
        return $result;
    }
}

```

```

]<?php
class Model_orders extends CI_Model
{
    public function __construct()
    {
        parent::__construct();

        $this->load->model('model_tables');
        $this->load->model('model_users');
    }
    public function getOrdersData($id = null)
    {
        if($id) {
            $sql = "SELECT * FROM orders WHERE id = ?";
            $query = $this->db->query($sql, array($id));
            return $query->row_array();
        }

        $user_id = $this->session->userdata('id');
        if($user_id == 1) {
            $sql = "SELECT * FROM orders ORDER BY id DESC";
            $query = $this->db->query($sql);
            return $query->result_array();
        }
        else {
            $user_data = $this->model_users->getUserData($user_id);
            $sql = "SELECT * FROM orders WHERE store_id = ? ORDER BY id DESC";
            $query = $this->db->query($sql, array($user_data['store_id']));
            return $query->result_array();
        }
    }
    public function getOrdersItemData($order_id = null)
    {
        if(!$order_id) {
            return false;
        }
        $sql = "SELECT * FROM order_items WHERE order_id = ?";
        $query = $this->db->query($sql, array($order_id));
        return $query->result_array();
    }
}

```

```

public function countOrderItem($order_id)
{
    if($order_id) {
        $sql = "SELECT * FROM order_items WHERE order_id = ?";
        $query = $this->db->query($sql, array($order_id));
        return $query->num_rows();
    }
}

```

```

public function countTotalPaidOrders()
{
    $sql = "SELECT * FROM orders WHERE paid_status = ?";
    $query = $this->db->query($sql, array(1));
    return $query->num_rows();
}

```

```

public function getProductData($id = null)
{
    if($id) {
        $sql = "SELECT * FROM products where id = ?";
        $query = $this->db->query($sql, array($id));
        return $query->row_array();
    }

    $user_id = $this->session->userdata('id');
    if($user_id == 1) {
        $sql = "SELECT * FROM products ORDER BY id DESC";
        $query = $this->db->query($sql);
        return $query->result_array();
    }
    else {
        $user_data = $this->model_users->getUserData($user_id);
        $sql = "SELECT * FROM products ORDER BY id DESC";
        $query = $this->db->query($sql);

        $data = array();
        foreach ($query->result_array() as $k => $v) {
            $store_ids = json_decode($v['store_id']);
            if(in_array($user_data['store_id'], $store_ids)) {
                $data[] = $v;
            }
        }
        return $data;
    }
}

```

```

public function getProductDataByCat($cat_id = null)
{
    if($cat_id) {
        $user_id = $this->session->userdata('id');
        if($user_id == 1) {
            $sql = "SELECT * FROM products ORDER BY id DESC";
            $query = $this->db->query($sql);
            $result = array();
            foreach($query->result_array() as $key => $value) {
                $category_ids = json_decode($value['category_id']);
                if(in_array($cat_id, $category_ids)) {
                    $result[] = $value;
                }
            }

            return $result;
        }
        else {
            $user_data = $this->model_users->getUserData($user_id);
            $sql = "SELECT * FROM products ORDER BY id DESC";
            $query = $this->db->query($sql);
            $data = array();
            foreach ($query->result_array() as $k => $v) {
                $store_ids = json_decode($v['store_id']);
                $category_ids = json_decode($v['category_id']);
                if(in_array($cat_id, $category_ids) && in_array($user_data['store_id'], $store_ids)) {
                    $data[] = $v;
                }
            }
            return $data;
        }
    }
}

```

```

public function getActiveProductData()
{
    $user_id = $this->session->userdata('id');
    if($user_id == 1) {
        $sql = "SELECT * FROM products WHERE active = ? ORDER BY id DESC";
        $query = $this->db->query($sql, array(1));
        return $query->result_array();
    }
    else {
        $this->load->model('model_users');
        $user_data = $this->model_users->getUserData($user_id);
        $sql = "SELECT * FROM products WHERE active = ? ORDER BY id DESC";
        $query = $this->db->query($sql, array(1));

        $data = array();
        foreach ($query->result_array() as $k => $v) {
            $store_ids = json_decode($v['store_id']);
            if(in_array($user_data['store_id'], $store_ids)) {
                $data[] = $v;
            }
        }
        return $data;
    }
}

```

```

public function countTotalProducts()
{
    $sql = "SELECT * FROM products";
    $query = $this->db->query($sql);
    return $query->num_rows();
}

```

```

public function getOrderYear()
{
    $sql = "SELECT * FROM orders WHERE paid_status = ?";
    $query = $this->db->query($sql, array(1));
    $result = $query->result_array();
    $return_data = array();
    foreach ($result as $k => $v) {
        $date = date('Y', $v['date_time']);
        $return_data[] = $date;
    }

    $return_data = array_unique($return_data);

    return $return_data;
}

```

```

public function getOrderData($year)
{
    if($year) {
        $months = $this->months();
        $sql = "SELECT * FROM orders WHERE paid_status = ?";
        $query = $this->db->query($sql, array(1));
        $result = $query->result_array();
        $final_data = array();
        foreach ($months as $month_k => $month_y) {
            $get_mon_year = $year.'-'. $month_y;

            $final_data[$get_mon_year][] = '';
            foreach ($result as $k => $v) {
                $month_year = date('Y-m', $v['date_time']);

                if($get_mon_year == $month_year) {
                    $final_data[$get_mon_year][] = $v;
                }
            }
        }

        return $final_data;
    }
}

public function getStoreWiseOrderData($year, $store)
{
    if($year && $store) {
        $months = $this->months();

        $sql = "SELECT * FROM orders WHERE paid_status = ? AND store_id = ?";
        $query = $this->db->query($sql, array(1, $store));
        $result = $query->result_array();

        $final_data = array();
        foreach ($months as $month_k => $month_y) {
            $get_mon_year = $year.'-'. $month_y;

            $final_data[$get_mon_year][] = '';
            foreach ($result as $k => $v) {
                $month_year = date('Y-m', $v['date_time']);

                if($get_mon_year == $month_year) {
                    $final_data[$get_mon_year][] = $v;
                }
            }
        }

        return $final_data;
    }
}

```

```

public function getStoresData($id = null)
{
    if($id) {
        $sql = "SELECT * FROM stores WHERE id = ?";
        $query = $this->db->query($sql, array($id));
        return $query->row_array();
    }

    $sql = "SELECT * FROM stores ORDER BY id DESC";
    $query = $this->db->query($sql);
    return $query->result_array();
}

```

```

public function getActiveStore()
{
    $sql = "SELECT * FROM stores WHERE active = ?";
    $query = $this->db->query($sql, array(1));
    return $query->result_array();
}

public function countTotalStores()
{
    $sql = "SELECT * FROM stores WHERE active = ?";
    $query = $this->db->query($sql, array(1));
    return $query->num_rows();
}

```

```

public function getTableData($id = null)
{
    if($id) {
        $sql = "SELECT * FROM tables WHERE id = ?";
        $query = $this->db->query($sql, array($id));
        return $query->row_array();
    }
    $user_id = $this->session->userdata('id');
    if($user_id == 1) {
        $sql = "SELECT * FROM tables ORDER BY id DESC";
        $query = $this->db->query($sql);
        return $query->result_array();
    }
    else {
        $this->load->model('model_users');
        $user_data = $this->model_users->getUserData($user_id);
        $sql = "SELECT * FROM tables WHERE store_id = ? ORDER BY id DESC";
        $query = $this->db->query($sql, array($user_data['store_id']));
        return $query->result_array();
    }
}

```

```

public function getActiveTable()
{
    $user_id = $this->session->userdata('id');
    if($user_id == 1) {
        $sql = "SELECT * FROM tables WHERE available = ? AND active = ?";
        $query = $this->db->query($sql, array(1, 1));
        return $query->result_array();
    }
    else {
        $this->load->model('model_users');
        $user_data = $this->model_users->getUserData($user_id);
        $sql = "SELECT * FROM tables WHERE store_id = ? AND available = ? AND active = ? ORDER BY id DESC";
        $query = $this->db->query($sql, array($user_data['store_id'], 1, 1));
        return $query->result_array();
    }
}

```

```

public function getUserData($userId = null)
{
    if($userId) {
        $sql = "SELECT * FROM users WHERE id = ?";
        $query = $this->db->query($sql, array($userId));
        return $query->row_array();
    }

    $sql = "SELECT * FROM users WHERE id != ? ORDER BY id DESC";
    $query = $this->db->query($sql, array(1));
    return $query->result_array();
}

public function getUserGroup($userId = null)
{
    if($userId) {
        $sql = "SELECT * FROM user_group WHERE user_id = ?";
        $query = $this->db->query($sql, array($userId));
        $result = $query->row_array();

        $group_id = $result['group_id'];
        $g_sql = "SELECT * FROM groups WHERE id = ?";
        $g_query = $this->db->query($g_sql, array($group_id));
        $q_result = $g_query->row_array();
        return $q_result;
    }
}

```

```

public function countTotalUsers()
{
    $sql = "SELECT * FROM users WHERE id != ?";
    $query = $this->db->query($sql, array(1));
    return $query->num_rows();
}

```

8. Link do repozytorium.

<https://github.com/szchelzarn/BAZY2021PWSZ-P4-G4>

9. Bilans.

Założenia projektu, które zostały jasno nakreślone na początku zajęć projektowych zostały wykonane w stu procentach.

10.Wnioski.

Projekt był bardzo czasochłonny i wymagał od nas dużo pracy oraz zaangażowania. Praca grupowa w naszym projekcie przebiegała bardzo konstruktywnie i bez konfliktów, co pozwoliło na dość sprawne zrealizowanie projektu. Jednakże nie obyło się bez problemów, które pochłonięły wiele czasu na ich rozwiązanie. Jednym z takich problemów była budowa bardzo rozbudowanej bazy danych, z którą były problemy aby połączyć ją z aplikacją. Wyrażamy nadzieję, że zaprezentowany przez nas projekt zawarł wszystkie założenia określone na wstępie.

11. Bibliografia.

1. <https://pl.wikipedia.org/wiki/JQuery>
2. [https://pl.wikipedia.org/wiki/Bootstrap_\(framework\)](https://pl.wikipedia.org/wiki/Bootstrap_(framework))
3. <https://pl.wikipedia.org/wiki/AJAX>
4. <http://www.w3big.com/pl/bootstrap/bootstrap-modal-plugin.html>
5. <https://pl.wikipedia.org/wiki/HTML>
6. <https://pl.wikipedia.org/wiki/CodeIgniter>
7. <https://pl.wikipedia.org/wiki/PHP>
8. <https://pl.wikipedia.org/wiki/MySQL>