1. **LMS login**
2. **Upali XAMP - Apache Start**
3. **MYSQL Workbench Create Schema**
4. **C:\xampp\htdocs extract final zip ovdje**

Konekcija na bazu (kredencijale za bazu nece biti ove, nego ce biti date na ispitu)

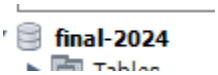
dbname = schema

host=servername

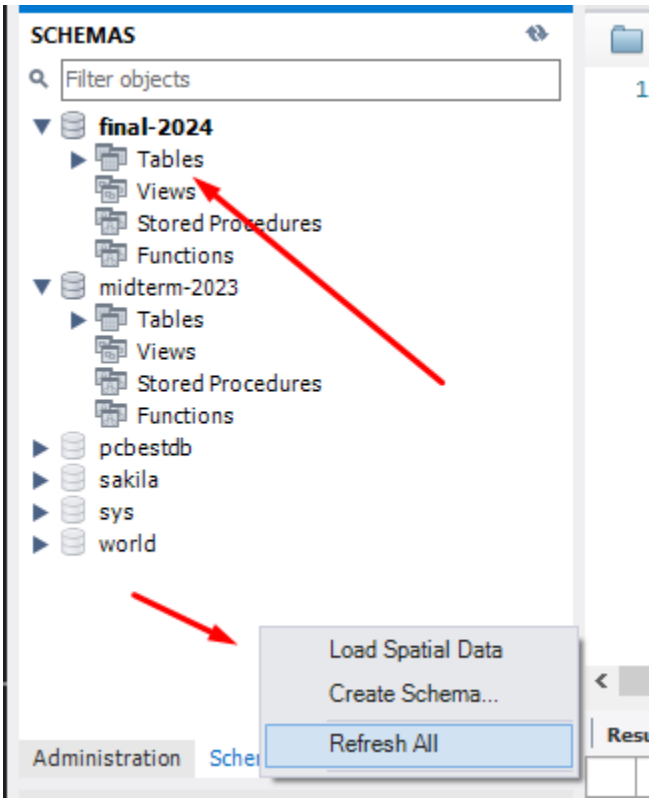
```
        try {
          /** TODO
           * List parameters such as servername, username, password, schema. Make sure to use
appropriate port
           */
          $host = '127.0.0.1';
          $username = 'root';
          $password = '12345';
          $dbname = 'final-2024';
          $port = 3306;

          /** TODO
           * Create new connetion
           */
          $this->conn = new PDO(
            "mysql:host=" . $host . ";dbname=" . $dbname . ";port=" . $port,
            $username,
            $password, [
                PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION,
                PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC
            ]
        );
          echo "Connected successfully";
        }
```

Copy .sql file iz final.zip i runaj ga u MYSQL workbenchu ali pazi da selectaš prethodno napravljenu schemu (mora ime biti boldirano, dupli klik na ime)



Nakon runanja tog .sql fajla desni klik na prazno i refresh all da dobijes tabele



SETUP GOTOV

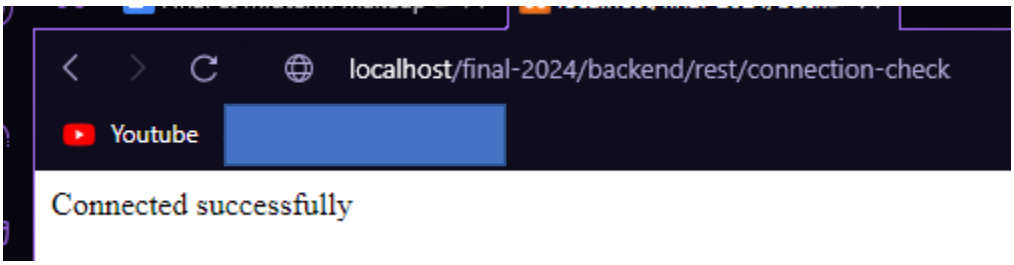
5. **Radimo rute - u rute pasteaj sljedeće**

/connection-check (pazi na ime klase mozda ne bude ExamDao nego nešto drugo npr FinalDao nešto tako)

```
new ExamDao();
```

probaj rutu ali pazi na to da ciljas na index.php

kada ti izbaci



Mozes izbrisati `echo "Connected successfully";`

Pazi na lokaciju index.php fajla pri trazenju ruta

iz base foldera (final-2024) trazis index.php

Ovo pises na vrh route fajla odmah ispod „<?php”

```
require_once __DIR__ . '/../services/ExamService.php';

Flight::set('exam_service', new ExamService);
```

/get-all

Ruta

```
$data = Flight::get('exam_service')->get_customers();
    Flight::json($data);
```

Servis

```
return $this->dao->get_customers();
```

Dao

```
      $query = "";

      $stmt = $this->conn->prepare($query);
      $stmt->execute();
      return $stmt->fetchAll(PDO::FETCH_ASSOC);
```

/get-by-id

Ruta

```
$data = Flight::get('exam_service')->get_customer_meals($customer_id);
    Flight::json($data);
```

Servis

```
return $this->dao->get_customer_meals($customer_id);
```

Dao

```
      $query = "SELECT f.name AS food_name, f.brand AS food_brand, m.created_at AS meal_date
                FROM foods f
                JOIN meals m ON m.food_id=f.id
                WHERE m.customer_id = " . $customer_id;

      $stmt = $this->conn->prepare($query);
      $stmt->execute();
      return $stmt->fetchAll(PDO::FETCH_ASSOC);
```

/add ili /update

Ruta

```
    $payload = Flight::request()->data->getData();

    $data = Flight::get('exam_service')->add_customer($payload);
    Flight::json($data);
```

Servis

```
return $this->dao->add_customer($customer);
```

Dao

```
      $query = "INSERT INTO customers (kolona1, kolona2, kolona3) VALUES (:vrijednost1,
:vrijednost2, :vrijednost3)";

      $stmt = $this->conn->prepare($query);

      $stmt->bindParam(':vrijednost1', $data['first_name']);
      $stmt->bindParam(':vrijednost2', $data['last_name']);
      $stmt->bindParam(':vrijednost3, $data['birth_date']);

      $stmt->execute();
      return $stmt->fetchAll(PDO::FETCH_ASSOC);
```

/paginated

Ovo se dodaje u query kakav bilo da je (ako query vec ima WHERE onda poslije toga ide AND LOWER ...)

```
        WHERE LOWER(kolona1) LIKE CONCAT('%', :search, '%') OR
        LOWER(kolona1) LIKE CONCAT('%', :search, '%')

        ORDER BY {$order_column} {$order_direction}
        LIMIT {$offset}, {$limit}";
```

Ruta

```
    $params = [
        'start' => 1,
        'search' => '',
        'limit' => 1,
        'order_column' => 'imeKolone',
        'order_direction' => 'ASC ili DESC',
    ];

    $data = Flight::get('exam_service')->foods_report($params['start'], $params['limit'],
$params['search'], $params['order_column'], $params['order_direction']);
    Flight::json($data);
```

Servis

```
    public function foods_report($offset, $limit, $search, $order_column, $order_direction){
        return $this->dao->get_foods_report($offset, $limit, $search, $order_column,
$order_direction);
    }
```

Dao sve isto kao ovdje osim querya koji će vjerovatno biti drukčiji samo dodaješ iste WHERE-ove, ORDER BY i LIMIT da bi se smatralo paginateanim

```
    public function get_foods_report($offset, $limit, $search, $order_column,
$order_direction){
      $query = "SELECT f.name, f.brand, f.image_url AS image
        FROM foods f
        WHERE LOWER(f.name) LIKE CONCAT('%', :search, '%') OR
        LOWER(f.brand) LIKE CONCAT('%', :search, '%')
        GROUP BY f.id
        ORDER BY {$order_column} {$order_direction}
        LIMIT {$offset}, {$limit}";

      $stmt = $this->conn->prepare($query);

      $stmt->bindParam(':search', $search);

      $stmt->execute();
      return $stmt->fetchAll(PDO::FETCH_ASSOC);
    }
```

6. **Idemo na frontend (ajax calls i ubacivanje podataka u html)**

U html fajl poslije svega paste ovo (mozda lokacije ovih fajlova ne budu iste ali imena će biti ista)

```
<script src="../assets/js/jquery-3.7.1-latest.js"></script>
<script src="../assets/js/jquery.min.js"></script>
```

U novi script za svaki zadatak praviš zasebnu funkciju i u nju stavljaš AJAX call gdje ćeš od sljedećeg mijenjati samo

url: 'ruta'

method: 'GET/POST/PUT/DELETE'

success funkciju

```
    $.ajax({
      url: '',
      method: 'GET',
      success: function(data) {

      },
      error: function(errorThrown) {
          console.log(errorThrown);
          $("#portofioItems").html("<p>There was an error loading the data. Please try again
later.</p>");
      }
    })
```

U success pravimo output string varijablu specifično po našim potrebama

```
output = '';
```

Onda sa forEach metodom prolazimo kroz **data** koju dobijamo u function(data linij gore^) i concatenate-amo na output

```
        data.forEach(meal => {
          output += '<tr><td>' + meal.food_name + '</td><td>' + meal.food_brand + '</td><td>'
+ meal.meal_date + '</td></tr>'
        })
```

Nakon toga kreiranu output varijablu moramo ubaciti na innerHTML atribut ***željenog elementa***

željeni element **možemo tražiti putem:**

document.getElementById

```
        const customerSelect = document.getElementById("customers-list");
        customerSelect.innerHTML = output;
```

ili document.querySelectora

```
        const mealTable = document.querySelector("#customer-meals tbody");
        mealTable.innerHTML = output;
```

nakon ovoga na dugme ili select ubacimo onchange ili onclick atribut kojim pokrećemo napravljenu funkciju:

```
    <button onclick="addCustomer()">Add Customer</button>
```

```
<select class="form-select" id="customers-list" onchange="getMeals(this.value)">
```

Ako funkcija treba da se pokrene sama čim udjemo na stranicu pozivamo funkciju odmah ispod nje.

Npr:

```
function getCustomers() {
  //funkcija … …
}

getCustomers();
```

Ako su koraci uradjeni pravilno ne bi trebalo biti problema

Kod POST ili PUT ajax poziva dodajemo sljedeće:

```
      data: customer,
      dataType: "json",
```

a customer objekat prethodno kreiramo tako što pokupimo iz odgovarajućih input polja sve relevantne informacija za import ili update:

```
    var customer = {};
    customer.first_name = document.getElementById("first_name").value;
    customer.last_name = document.getElementById("last_name").value;
    customer.birth_date = document.getElementById("birth_date").value;
```

na kraju funkcija cijela izgleda slično ovome:

```
   function addCustomer(){

    var customer = {};
    customer.first_name = document.getElementById("first_name").value;
    customer.last_name = document.getElementById("last_name").value;
    customer.birth_date = document.getElementById("birth_date").value;

    $.ajax({
      url: '../../backend/rest/customers/add',
      method: 'POST',
      data: customer,
      dataType: "json",
      success: function(data) {
        console.log("success");

        getCustomers();
      },
      error: function(jqXHR, textStatus, errorThrown) {
          console.log(jqXHR);
          console.log(textStatus);
          console.log(errorThrown);
          $("#portofioItems").html("<p>There was an error loading the data. Please try again
later.</p>");
      }
    })
  }
```

U slučaju da treba refresh poslije dodavanja ili updateanja podataka pozivamo opet u successu funkciju koju smo prvu napravili odnosno onu koja fetcha sve iz baze

```
getCustomers();
```

uz sve ove korake finalni ispit bez swaggera i JWTa bit trebao biti ispravan

u slučaju da pri pozivanju ruta u url-u one vrate podatke ispravno a na frontendu ne rade onda je greška u script elementima ondosno u javascript funkcijama i ajax callovima