Web Technologies

PROG8186

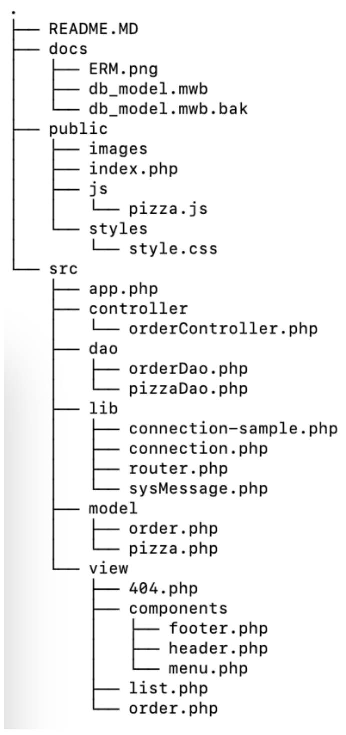
Assignment 3

Rodrigo Bruner (#8993586)

I'll start by explaining the structure of the project. At the root of the system, you will find three directories:

- **doc:** project documentation
- **public:** public and root files of the project in Apache (images, CSS, JS, etc...)
- **src:** system source code.
  - The solution uses the MVC (Model-View-Controller) design pattern. A directory was created to store each layer of this pattern. Within the View layer, I have made another directory called "components" which aims to store reused code fragments within the View layer.
  - Another pattern adopted was DAO (Data Access Object), so another directory was created for these files.
  - Finally, the lib directory is a directory to store the libraries created to support this development.

Below is the project's directory tree to make it easier to understand the structure.



```
.
├── README.MD
├── docs
│   ├── ERM.png
│   ├── db_model.mwb
│   └── db_model.mwb.bak
├── public
│   ├── images
│   ├── index.php
│   ├── js
│   │   └── pizza.js
│   └── styles
│       └── style.css
└── src
    ├── app.php
    ├── controller
    │   └── orderController.php
    ├── dao
    │   ├── orderDao.php
    │   └── pizzaDao.php
    ├── lib
    │   ├── connection-sample.php
    │   ├── connection.php
    │   ├── router.php
    │   └── sysMessage.php
    ├── model
    │   ├── order.php
    │   └── pizza.php
    └── view
        ├── 404.php
        ├── components
        │   ├── footer.php
        │   ├── header.php
        │   └── menu.php
        ├── list.php
        └── order.php
```

Below is a presentation of each of the solution's source code files.

./public/index.php

```php
<?php
    //Just call app.php
    require_once '../src/app.php';
?>
```

./src/app.php

This file is responsible for creating the solution's routes and importing the main files for the system to work.

```php
<?php
include 'lib/router.php';
include 'lib/connection.php';
include 'lib/sysMessage.php';
include 'model/order.php';
include 'dao/orderDao.php';
include 'model/pizza.php';
include 'dao/pizzaDao.php';
include 'controller/orderController.php';

$app = new Router();

$orderController = new OrderController();

/**
 * Routes
 *
 */

 // Index, order page
$app->get(  '/', function() use ($orderController) {
    $orderController->index();
});

// List orders
$app->get('/list', function() use ($orderController) {
    $orderController->listOrders();
});
```

```
// Create order
$app->post( '/', function() use ($orderController) {
    $orderResult = $orderController->createOrder();
    if($orderResult->getType() == SysMessage::ERROR){
        //Return to index page with error message
        header("Location: http://localhost/?error=".urlencode($orderResult-
>getMessage()));
    } else {
        //Return to index page with success message
        header("Location: http://localhost/?success=".urlencode("Order created
successfully"));
    }
});

// Call the callback of the route
$app->start();

?>
```

src/lib/router.php

Another design pattern has been adopted, which in turn handles HTTP requests and directs the application's controllers. Basically, it creates an associative matrix between an HTTP action, a system path and a callback function. The start method basically captures the HTTP method called from the URI to find the callback and execute it.

```
<?php
/**
 * Router class
 * Manage the routes of the application
 *
 * @autor: Rodrigo Bruner
 */


class Router {

    /**
     * Routes
     * @var array
     */
    private $routes = [];
```

```php
    /**
     * Add a route to the HTTP GET method
     * @param string $path
     * @param callable $callback
     */
    public function get($path, $callback) {
        $this->routes['GET'][$path] = $callback;
    }

    /**
     * Add a route to the HTTP POST method
     * @param string $path
     * @param callable $callback
     */
    public function post($path, $callback) {
        $this->routes['POST'][$path] = $callback;
    }

    /**
     * Resolve the route
     * Call the callback of the route
     * If the calback does not exist, return a 404 error
     * @return mixed
     */
    public function start() {

        $method = $_SERVER['REQUEST_METHOD'];
        $uri = $_SERVER['REQUEST_URI'];
        $path = parse_url($uri, PHP_URL_PATH);

        $callback = $this->routes[$method][$path] ?? false;

        // var_dump($path);
        if ($callback === false) {
            http_response_code(404);
            require __DIR__ . '../views/404.php';
            return;
        }
        echo call_user_func($callback);
    }

}

?>
```

src/lib/connection.php

This file connects to the database via a PHP lib called PDO (PHP Data Objects). Another designer partner adopted it, this time Sington.

```php
<?php
/**
 * Connection class
 * Manage the connection to the database
*/

class Connection {

    // Database settings
    private static $host = 'localhost';
    private static $database = 'pizzaria';
    private static $username = 'root';
    private static $password = 'root';

    //Connection
    private static $conn;

    // create or get connection
    public static function getConnection() {
        try {
            if (!isset(self::$conn)) {
                $dsn = 'mysql:host=' . self::$host . ';dbname=' . self::$database;
                self::$conn = new PDO($dsn, self::$username, self::$password);
                self::$conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
            }

            return self::$conn;
        } catch (PDOException $e) {
            die('Caught exception: Database connection failed, please check your
settings or contact the administrator. Error: ' . $e->getMessage());
        }
    }

    // close connection
    public static function closeConnection() {
        if (isset(self::$conn)) {
            self::$conn->close();
        }
    }
    // Prevents the creation of new instances of the class
    public function __clone() {}

    // Prevents the creation of new instances of the class
    public function __wakeup() {}

}

?>
```

src/lib/sysMessage.php

I created a transport object between the application layers called SysMessage. This time, it's not a design pattern but a way of facilitating communication through a clear contract, centralizing the messaging part of the layers in an object with a clear responsibility, separating responsibilities and facilitating the maintenance of the solution. The designer pattern constructor and Setters and Getters are applied to this object.

```
<?php
class SysMessage {
    const ERROR = 'ERROR';
    const SUCCESS = 'SUCCESS';
    // type of message
    private $type;
    // content of the message
    private $message;
    // data to be sent in the message
    private $extraData;

    // constructor
    public function __construct($type, $message, $extraData = null) {
        $this->type = $type;
        $this->message = $message;
        $this->extraData = $extraData;
    }

    // getters and setters

    public function getType() {
        return $this->type;
    }

    public function getMessage() {
        return $this->message;
    }
```

```php
    public function getExtraData() {
        return $this->extraData;
    }

    public function setType($type) {
        $this->type = $type;
    }

    public function setMessage($message) {
        $this->message = $message;
    }

    public function setExtraData($extraData) {
        $this->extraData = $extraData;
    }
}
?>
```

src/controller/orderController.php

As the solution only deals with Orders and not other system modules, I centralized everything in a single Controller to make it easier to present the data.

```php
<?php
class OrderController{

    private $pdo; // Database connection
    private $orderDAO; // Order DAO
    private $pizzaDAO; // Pizza DAO

    // Initialize db and DAOs
    public function __construct(){
        $this->pdo = Connection::getConnection();
        $this->orderDAO = new OrderDAO($this->pdo);
        $this->pizzaDAO = new PizzaDAO($this->pdo);
    }

    //Home page, with form to create an order
    public function index(){
        include '../src/view/order.php';
    }

    public function createOrder(){
        // Get the data from the form
        $firstName = $_POST['firstName'] ?? null;
        $lastName = $_POST['lastName'] ?? null;
        $email = $_POST['email'] ?? null;
        $phone = $_POST['phone'] ?? null;
        $street = $_POST['street'] ?? null;
        $number = $_POST['number'] ?? null;

        // Validate the data
        if(!$firstName || !$lastName || !$email || !$phone || !$street || !$number){
            return new SysMessage(SysMessage::ERROR, 'Please fill in all fields');
        }
        // Validate email
        if(!filter_var($email, FILTER_VALIDATE_EMAIL)){
            return new SysMessage(SysMessage::ERROR, 'Invalid email');
        }
        // Validate phone
        if(!is_numeric($phone)){
            return new SysMessage(SysMessage::ERROR, 'Invalid phone number');
        }
        // Validate number
        if(!is_numeric($number)){
            return new SysMessage(SysMessage::ERROR, 'Invalid number');
        }
        // Create new order
        $order = new Order(
            0,
            $firstName,
            $lastName,
            $email,
            $phone,
            $street,
            $number
        );

        try {
            //Create the order and get the order id
            $orderId = $this->orderDAO->create($order);

            // If the order not created, return an error message
            if(!$orderId){
                return new SysMessage(SysMessage::ERROR, 'An error occurred while
creating the order');
            }

            // Add pizzas to the order
            $addPizzasResult = $this->addPizzas($orderId);

            // If the pizzas not created
            if ($addPizzasResult->getType() === SysMessage::ERROR) {
                //delete the order
                $this->orderDAO->delete($orderId);
                //return an error message
                return new SysMessage(SysMessage::ERROR, 'An error occurred while 
creating the order', $addPizzasResult->getMessage());
            }
            // Set the order id
            $order->setId($orderId);
            // Set the pizzas
            $order->setPizzas($addPizzasResult->getExtraData());
            // Return a success message
            return new SysMessage(SysMessage::SUCCESS, 'Order '.$orderId.' created 
successfully', $order);

        //If some exception occurs
        } catch (Exception $e) {
            // Return an error message
            return new SysMessage(SysMessage::ERROR, 'An error occurred while creating 
the order', $e);
        }
    }


    // Add pizzas to the order
    public function addPizzas(int $orderId){

        // Get the number of pizzas
        $numberOfPizzas = $_POST['qtPizzas'] ?? 0;

        // If the number of pizzas = 0 return an error message
        if ($numberOfPizzas < 1) {
            return new SysMessage(SysMessage::ERROR, 'Please specify the number of 
pizzas');
        }

        // for each pizza
        for ($i = 1; $i <= $numberOfPizzas; $i++) {
            // Get the data from the form
            $size = $_POST["size{$i}"] ?? null;
            $dough = $_POST["dough{$i}"] ?? null;
            $sauce = $_POST["sauce{$i}"] ?? null;
            $cheese = $_POST["cheese{$i}"] ?? [];
            $toppings = $_POST["toppings{$i}"] ?? [];

            // Validate if some data is not set
            if (!$size || !$dough || !$sauce || !$cheese || !$toppings) {
                return new SysMessage(SysMessage::ERROR, 'Please fill in all required 
fields for pizza ' . $i);
```

```php
                return;
            }


            // Create a new pizza and save it in an array
            $pizzas[] = new Pizza(
                $orderId,
                $size,
                $dough,
                $sauce,
                $cheese,
                $toppings
            );
        }
        //If some error occurs return an error and stop the process

        try {
            // For each pizza in the array
            foreach ($pizzas as $pizza) {
                // Save the pizza in the database
                $pizzasId[] = $this->pizzaDAO->create($pizza);
            }
            // Return a success message
            return new SysMessage(SysMessage::SUCCESS, 'Pizzas created successfully',
$pizzas);
        } catch (Exception $e) {
            $this->deletePizzasByOrderID($orderId);
            return new SysMessage(SysMessage::ERROR, 'An error occurred while creating
the pizzas', $e);
        }
    }


    public function listOrders(){
        // Get all orders
        $orders = $this->orderDAO->list();
        // For each order, get the pizzas
        foreach ($orders as $key => $order) {
            // Set the pizzas in the order
            $orders[$key]->setPizzas($this->pizzaDAO->selectByOrderID($order-
>getId()));
        }
        include '../src/view/list.php';
    }
}
?>
```

src/model/order.php

```php
<?php
class Order{
    private int $id = 0;
    private string $firstName = "";
    private string $lastName = "";
    private string $email = "";
    private string $phone = "";
    private string $street = "";
    private string $number = "";
    private array $pizzas = [];

    // Constructor
    public function __construct(int $id, string $firstName, string $lastName, string
$email, string $phone, string $street, string $number){
        $this->id = $id;
        $this->firstName = $firstName;
        $this->lastName = $lastName;
        $this->email = $email;
        $this->phone = $phone;
        $this->street = $street;
        $this->number = $number;
    }

    //Getters and Setters
    public function getId(): int{
        return $this->id;
    }

    public function getFirstName(): string{
        return $this->firstName;
    }

    public function getLastName(): string{
        return $this->lastName;
    }

    public function getEmail(): string{
        return $this->email;
    }

    public function getPhone(): string{
        return $this->phone;
    }

    public function getStreet(): string{
        return $this->street;
    }

    public function getNumber(): string{
        return $this->number;
    }

    public function getPizzas(): array{
        return $this->pizzas;
    }

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

    public function setFirstName(string $firstName){
        $this->firstName = $firstName;
    }

    public function setLastName(string $lastName){
        $this->lastName = $lastName;
    }

    public function setEmail(string $email){
        $this->email = $email;
    }

    public function setPhone(string $phone){
        $this->phone = $phone;
    }

    public function setStreet(string $street){
        $this->street = $street;
    }

    public function setNumber(string $number){
        $this->number = $number;
    }

    public function setPizzas(array $pizzas){
        $this->pizzas = $pizzas;
    }

    public function addPizza(Pizza $pizza){
        $this->pizzas[] = $pizza;
    }
}

?>
```

src/model/pizza.php

```php
<?php

class Pizza{

    private int $orderId = 0;
    private string $size  = "Medium";
    private string $doughType = "";
    private string $sauceType = "";
    private $cheesesType = [];
    private $toppingsType = [];

    // Constructor
    public function __construct(
        int $orderId,
        string $size,
        string $doughType,
        string $sauceType,
        array $cheesesType,
        array $toppingsType
    ) {
        $this->orderId = $orderId;
        $this->size = $size;
        $this->doughType = $doughType;
        $this->sauceType = $sauceType;
        $this->cheesesType = $cheesesType;
        $this->toppingsType = $toppingsType;
    }

    //Getters and Setters
    public function getOrderId(): int{
        return $this->orderId;
    }

    public function getSize(): string{
        return (string) $this->size;
    }

    public function getDoughType(): string{
        return $this->doughType;
    }

    public function getSauceType(): string{
        return $this->sauceType;
    }
    public function getCheesesType(): array{
        return $this->cheesesType;
    }

    public function getCheesesTypeAsString(): string {
        return implode(", ", $this->cheesesType);
    }

    public function getToppingsType(): array{
        return $this->toppingsType;
    }

    public function getToppingsTypeAsString(): string {
        return implode(", ", $this->toppingsType);
    }

    public function setOrderId(int $orderId){
        $this->orderId = $orderId;
    }

    public function setSize(string $size){
        $this->size = $size;
    }

    public function setDoughType(string $doughType){
        $this->doughType = $doughType;
    }

    public function setSauceType(string $sauceType){
        $this->sauceType = $sauceType;
    }

    public function setCheesesType(array $cheesesType){
        $this->cheesesType = $cheesesType;
    }

    public function setToppingsType(array $toppingsType){
        $this->toppingsType = $toppingsType;
    }
}
```

src/dao/orderDao.php

```php
<?php
class OrderDAO {
    // PDO instance
    private $pdo;

    public function __construct($pdo) {
        // Set the connection to the database
        $this->pdo = $pdo;
    }

    public function create(Order $order) {
        try {
            // Create the sql
            $sql = "INSERT INTO orders (first_name, last_name, email, phone, street,
number) VALUES (?, ?, ?, ?, ?, ?)";
            // Prepare the sql
            $stmt = $this->pdo->prepare($sql);
            // Bind the values and execute the sql
            $stmt->execute([
                $order->getFirstName(),
                $order->getLastName(),
                $order->getEmail(),
                $order->getPhone(),
                $order->getStreet(),
                $order->getNumber()
            ]);
            // Return the last inserted id
            return $this->pdo->lastInsertId();
        } catch (Exception $e) {
            throw new Exception("Error create new order", 1);
        }
    }

    public function list(){
        try {
            // Create the sql
            $sql = "SELECT * FROM orders";
            // Prepare the sql
            $stmt = $this->pdo->prepare($sql);
            // Execute the sql
            $stmt->execute();
            $orders = [];
            while ($row = $stmt->fetch()) {
                $orders[] = new Order(
                    $row['id'],
                    $row['first_name'],
                    $row['last_name'],
                    $row['email'],
                    $row['phone'],
                    $row['street'],
                    $row['number'],
                );
            }
            return $orders;
        } catch (Exception $e) {
            return $e;
        }
    }

    public function update(Order $order) {
        try {
            // Create the sql
            $sql = "UPDATE orders SET first_name = ?, last_name = ?, email = ?, phone
= ?, street = ?, number = ? WHERE id = ?";
            // Prepare the sql
            $stmt = $this->pdo->prepare($sql);
            // Bind the values and execute the sql
            $stmt->execute([
                $order->getId(),
                $order->getFirstName(),
                $order->getLastName(),
                $order->getEmail(),
                $order->getPhone(),
                $order->getStreet(),
                $order->getNumber(),
            ]);
            return true;
        } catch (Exception $e) {
          return $e;
        }
    }

    public function delete($orderId) {
        try {
            $sql = "DELETE FROM orders WHERE id = ?";
            $stmt = $this->pdo->prepare($sql);
            $stmt->execute([$orderId]);
            return true;
        } catch (Exception $e) {
            return $e;
        }
    }
}
?>
```

src/dao/pizzaDao.php

```php
<?php

class PizzaDAO {

    // DB Connection
    private $pdo;

    public function __construct($pdo) {
        // Get the connection
        $this->pdo = Connection::getConnection();
    }

    public function create(Pizza $pizza) {
        try{
            // create sql
            $sql = "INSERT INTO pizzas (orders_id, size, dough_type, sauce_type, 
cheeses_type, toppings_type) VALUES (?, ?, ?, ?, ?, ?)";
            // prepare sql
            $stmt = $this->pdo->prepare($sql);
            // bind values and execute the sql
            $stmt->execute([
                $pizza->getOrderId(),
                $pizza->getSize(),
                $pizza->getDoughType(),
                $pizza->getSauceType(),
                $pizza->getCheesesTypeAsString(),
                $pizza->getToppingsTypeAsString()
            ]);
            // return the last inserted id
            return $this->pdo->lastInsertId();
        }catch(Exception $e){
            return $e;
        }
    }


    public function deletePizzasByOrderID($orderId) {
        try{
            // create sql
            $sql = "DELETE FROM pizzas WHERE orders_id = ?";
            // prepare sql
            $stmt = $this->pdo->prepare($sql);
            // bind values and execute the sql
            $stmt->execute([$orderId]);
            // return true if success
            return true;
        }catch(Exception $e){
            return $e;
        }
    }


    public function selectByOrderID($orderId) {
        try {
            // create sql
            $sql = "SELECT * FROM pizzas WHERE orders_id = ?";
            // prepare sql
            $stmt = $this->pdo->prepare($sql);
            // bind values and execute the sql
            $stmt->execute([$orderId]);
            // fetch all results
            $rows = $stmt->fetchAll();
            $pizzas = [];

            // foreach pizza, create a new pizza object in array
            foreach ($rows as $row) {
                $pizzas[] = new Pizza(
                    $row['orders_id'],
                    $row['size'],
                    $row['dough_type'],
                    $row['sauce_type'],
                    explode(',', $row['cheeses_type']),
                    explode(',', $row['toppings_type'])
                );
            }
            // return the array of pizzas
            return $pizzas;
        } catch (Exception $e) {
            return $e;
        }
    }
}
?>
```

src/view/components/header.php

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="styles/style.css">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/4.7.0/css/font-awesome.min.css">
    <script src="js/pizza.js"></script>
    <title>Pizza day!</title>
</head>
<body>
```

src/view/components/menu.php

```
<nav>
    <a class="active" href="/"><i class="fa fa-fw fa-shopping-basket"></i> Order
now</a>
    <a href="/list"><i class="fa fa-fw fa-tasks"></i> Order list</a>
</nav>
```

src/view/components/fooder.php

```
</body>
</html>
```

src/view/404.php

```
<?php
    require_once 'components/header.php';
    require_once 'components/menu.php';
?>
<h1>Page not found</h1>
<?php
require_once 'components/footer.php';
?>
```

src/view/order.php

```
<?php
    require_once 'components/header.php';
    require_once 'components/menu.php';
?>
<main>
    <h1>Pizza day!</h1>
    <?php
        if(isset($_GET['error'])){
            echo '<div class="errorMsg">'.$_GET['error'].'</div>';
        }
        if(isset($_GET['success'])){
            echo '<div class="successMsg">'.$_GET['success'].'</div>';
        }
    ?>
    <!-- I stop sending the form to validate the data -->
    <form submit="/" method="POST" onsubmit=""> <!--
event.preventDefault();validateForm();-->
        <div id="makePizza">
            <div id="pizzas">
                <h3>Enter the number of pizzas you want.</h3>
            </div>
        </div>
        <div class="formGrid">
            <label for="number">Enter the number of pizzas you want</label>
            <!--Every time this value changes, I update the pizza form-->
            <input type="number" name="qtPizzas" id="number"
onchange="addPizzaFields()">

            <label for="firstName">First name</label>
            <input type="text" name="firstName" id="firstName">

            <label for="lastName">Last name</label>
            <input type="text" name="lastName" id="lastName">

            <label for="email">E-mail</label>
            <input type="text" name="email" id="email">

            <label for="phone">Phone number</label>
            <input type="text" name="phone" id="phone">

            <label for="street">Street</label>
            <input type="text" name="street" id="street">

            <label for="stNumber">Number</label>
            <input type="text" name="number" id="stNumber">

            <button type="submit">Place order</button>
        </div>
    </form>
</main>
<script>
    addPizzaFields();
</script>
<?php
    require_once 'components/footer.php';
?>
```

src/view/list.php

```
<?php
require_once 'components/header.php';
require_once 'components/menu.php';
?>
<main>
    <h1>Orders</h1>
    <table>
        <thead>
            <tr>
                <th>Order ID</th>
                <th>First name</th>
                <th>Last name</th>
                <th>E-mail</th>
                <th>Phone</th>
                <th>Street</th>
                <th>Number</th>
                <th>Qt pizzas</th>
                <th>Show detatil</th>
            </tr>
        </thead>
        <tbody>
            <?php foreach($orders as $order){ ?>
                <tr>
                    <td><?php echo $order->getId() ?></td>
                    <td><?php echo $order->getFirstName() ?></td>
                    <td><?php echo $order->getLastName() ?></td>
                    <td><?php echo $order->getEmail() ?></td>
                    <td><?php echo $order->getPhone() ?></td>
                    <td><?php echo $order->getStreet() ?></td>
                    <td><?php echo $order->getNumber() ?></td>
                    <td><?php echo count($order->getPizzas()) ?></td>
                    <td>
                        <button onclick="toggleVisibility('order<?php echo $order-
>getId() ?>', this)">
                            <i class="fa fa-eye"></i> Show Details
                        </button>
                    </td>
                </tr>
                <tr id='order<?php echo $order->getId() ?>' style="display: none;"">
                    <td colspan="8">
                        <table>
                            <thead>
                                <tr>
                                    <th>Size</th>
                                    <th>Dough</th>
                                    <th>Souce</th>
                                    <th>Cheeses</th>
                                    <th>Toppings</th>
                                </tr>
                            </thead>
                            <tbody>
                                <?php foreach($order->getPizzas() as $pizza){ ?>
                                    <tr>
                                        <td><?php echo $pizza->getSize() ?></td>
                                        <td><?php echo $pizza->getDoughType() ?></td>
                                        <td><?php echo $pizza->getSauceType() ?></td>
                                        <td><?php echo $pizza-
>getCheesesTypeAsString() ?></td>

                                        <td><?php echo $pizza-
>getToppingsTypeAsString() ?></td>
                                    </tr>
                                <?php } ?>
                            </tbody>
                        </table>
                    </td>
                </tr>
            <?php } ?>
        </tbody>
    </table>
</main>
<?php
require_once 'components/footer.php';
?>
```

public/js/pizza.js

```
function addPizzaFields() {
    /* Define the options for each field */
    const sizes = ['Small', 'Medium', 'Large', 'X-Large'];

    const doughTypes = [
        'whole grain crust',
        'whole grain thin crust',
        'whole grain thick crust',
        'regular',
        'regular thin crust',
        'regular thick crust'];

    const sauceTypes = [
        'home-style Italian tomato',
        'buffalo blue cheese',
        'creamy garlic',
        'chipotle',
        'pesto',
        'spicy',
        'sweet chilli Thai',
        'tandoori',
        'Texas',
        'no sauce'];

    const cheeseTypes = [
        'mozzarella',
        'dairy-free',
        'four cheese blend'];

    const toppingsTypes = [
        'anchovies',
        'artichokes',
        'bacon strips',
        'broccoli',
        'bruschetta',
        'buffalo chicken',
        'caramelized onions',
        'cilantro',
        'chipotle chicken',
        'chipotle steak',
        'chorizo sausage',
        'fire-roasted red peppers',
        'green olives',
        'green peppers',
        'grilled chicken',
        'grilled zucchini',
        'ground beef',
        'hot banana peppers',
        'Italian ham',
        'jalapeno peppers',
        'kalamata olives',
        'mushrooms',
        'New York style pepperoni',
        'pepperoni',
        'pineapple',
        'plant-based chorizo crumble',
        'plant-based pepperoni',
        'red onions',
        'roasted garlic',
        'Roma tomatoes',
        'salami',
        'spicy Italian sausage',
        'steak strips',
        'spinach',
        'sun-dried tomatoes'
    ];

    // Preparing the div to receive the fields
    const pizzasDiv = document.getElementById('pizzas');
    pizzasDiv.innerHTML = '';

    // Taking the number of pizzas and converting to integer
    var numberOfPizzas = (document.getElementById('number').value*1);

    if(numberOfPizzas < 1) {
        numberOfPizzas = 1;
        document.getElementById('number').value = 1;
    }

    // check if the number of pizzas is greater than 10
    if(numberOfPizzas > 10) {
        alert('For orders of more than 10 pizzas, please call (222) 222-2222.');
```

```
        numberOfPizzas = 10;
        document.getElementById('number').value = 10;
    }

    // If the number of pizzas is less than 1, a message is displayed
    if (numberOfPizzas < 1) {
        pizzasDiv.innerHTML = '<h3>Enter the number of pizzas you want.<h3>';
    }

    // For each pizza, the fields are created
    for (let i = 1; i < numberOfPizzas+1; i++) {

        console.log('Creating pizza fields for pizza number', i);

        // For each pizza, a div is created that will receive the fields
        const pizzaContainer = document.createElement('div');
        pizzaContainer.className = 'pizzaContainer';

        // Creating a element h3 to show the title of the pizza

        const title = document.createElement('h3');
        // adding the title to the pizza
        title.textContent = `Pizza ${i}`;

        // adding the title to the pizza container
        pizzaContainer.appendChild(title);

        // Creating the label for size and a select field
        const sizeLabel = document.createElement('label');
        sizeLabel.textContent = '* Size for Pizza';
        pizzaContainer.appendChild(sizeLabel);

        // Creating the selecte field
        const sizeSelect = document.createElement('select');
        sizeSelect.name = `size${i}`;

        // for each size, in array sizes, a option is created
        sizes.forEach(size => {
            // create the element
            const option = document.createElement('option');
            // set attributes
            option.value = size;
            option.textContent = size;
            if (size === 'large') { //set default value
                option.selected = true;
            }
            // adding the option to the select field
            sizeSelect.appendChild(option);
```

```javascript
});
pizzaContainer.appendChild(sizeSelect);

//Creating the label for dough and the options
const doughLabel = document.createElement('label');
doughLabel.textContent = '* Dough type for Pizza';
pizzaContainer.appendChild(doughLabel);

// to apply css I created a div to receive the radio buttons
const doughContainer = document.createElement('div');
doughContainer.className = 'doughContainer';
// foreach dough type, in array doughTypes, add a option
doughTypes.forEach(dough => {
    // create a label
    const doughOptionLabel = document.createElement('label');
    //set the text
    doughOptionLabel.textContent = dough;
    //add the label to the div
    doughContainer.appendChild(doughOptionLabel);

    // create a element
    const doughOptionInput = document.createElement('input');
    // set the attributes
    doughOptionInput.type = 'radio';
    doughOptionInput.name = `dough${i}`;
    doughOptionInput.value = dough;
    // add the input to the label
    doughOptionLabel.appendChild(doughOptionInput);
});
pizzaContainer.appendChild(doughContainer);

// The sauce follows the same logic as the dough
const sauceLabel = document.createElement('label');
sauceLabel.textContent = '* Sauce for Pizza';
pizzaContainer.appendChild(sauceLabel);

const sauceContainer = document.createElement('div');
sauceContainer.className = 'sauceContainer';

sauceTypes.forEach(sauce => {
    const sauceOptionLabel = document.createElement('label');
    sauceOptionLabel.textContent = sauce;
    sauceContainer.appendChild(sauceOptionLabel);

    const sauceOptionInput = document.createElement('input');
    sauceOptionInput.type = 'radio';
    sauceOptionInput.name = `sauce${i}`;
    sauceOptionInput.value = sauce;
```

```
            sauceOptionLabel.appendChild(sauceOptionInput);
        });
        pizzaContainer.appendChild(sauceContainer);

        // Exactly the same logic as the sauce and dough but with checkboxes
        const cheeseLabel = document.createElement('label');
        cheeseLabel.textContent = 'Base cheese for Pizz';
        pizzaContainer.appendChild(cheeseLabel);
        const cheeseContainer = document.createElement('div');
        cheeseContainer.className = 'cheeseContainer';

        cheeseTypes.forEach(cheese => {
            const cheeseOptionLabel = document.createElement('label');
            cheeseOptionLabel.textContent = cheese;
            cheeseContainer.appendChild(cheeseOptionLabel);

            const cheeseOptionInput = document.createElement('input');
            cheeseOptionInput.type = 'checkbox';
            cheeseOptionInput.name = `cheese${i}[]`;
            cheeseOptionInput.value = cheese;
            cheeseOptionLabel.appendChild(cheeseOptionInput);
        });

        pizzaContainer.appendChild(cheeseContainer);

        // Same logic as the cheese
        const toppingsLabel = document.createElement('label');
        toppingsLabel.textContent = '* Toppings for Pizza';
        pizzaContainer.appendChild(toppingsLabel);
        const toppingsContainer = document.createElement('div');
        toppingsContainer.className = 'toppingsContainer';

        toppingsTypes.forEach(topping => {
            const toppingOptionLabel = document.createElement('label');
            toppingOptionLabel.textContent = topping;
            toppingsContainer.appendChild(toppingOptionLabel);

            const toppingOptionInput = document.createElement('input');
            toppingOptionInput.type = 'checkbox';
            toppingOptionInput.value = topping;
            toppingOptionInput.name = `toppings${i}[]`;
            toppingOptionLabel.appendChild(toppingOptionInput);
        });
        pizzaContainer.appendChild(toppingsContainer);

        pizzasDiv.appendChild(pizzaContainer);
    }
}
```

```
function validateForm() {

    errorMessages = "";


    // Check if the number of pizzas is filled out
    const numberOfPizzas = document.getElementById('number').value;
    if (!numberOfPizzas || isNaN(numberOfPizzas)) {
        errorMessages += '<li>Please fill out the number of pizzas field.</li>';
    }

    if (numberOfPizzas > 10) {
        errorMessages += '<li>For orders of more than 10 pizzas, please call (222)
222-2222.</li>';
    }

    const firstName = document.getElementById('firstName').value;
    if (firstName.length < 2) {
        errorMessages += '<li>First name must be at least 2 characters long.</li>';
    }

    const lastName = document.getElementById('lastName').value;
    if (lastName.length < 2) {
        errorMessages += '<li>Last name must be at least 2 characters long.</li>';
    }

    const email = document.getElementById('email').value;
    const emailPattern = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
    if (!emailPattern.test(email)) {
        errorMessages += '<li>Please enter a valid email address.</li>';
    }

    const phone = document.getElementById('phone').value;
    const phonePattern = /^\d{3}-\d{3}-\d{4}$/;
    if (!phonePattern.test(phone)) {
        errorMessages += '<li>Please enter a valid phone number.</li>';
    }

    const street = document.getElementById('street').value;
    if (street.length < 2) {
        errorMessages += '<li>Street must be at least 2 characters long.</li>';
    }

    const stNumber =  document.getElementById('stNumber').value;
    if (isNaN(stNumber) || stNumber < 1 || stNumber === "") {
        errorMessages += '<li>Number must be a number.</li>';
    }
    console.log(stNumber);

    for (let i = 0; i < numberOfPizzas; i++) {
        const size = document.querySelector(`select[name="size${i + 1}"]`).value;
        const dough = document.querySelector(`input[name="dough${i + 1}"]:checked`);
        const sauce = document.querySelector(`input[name="sauce${i + 1}"]:checked`);
        const toppings = document.querySelectorAll(`input[name="toppings${i +
1}"]:checked`);

        if (!size || !dough || !sauce || toppings.length === 0) {
            errorMessages += `<li>Please fill out all fields for Pizza ${i +
1}.</li>`;
        }
    }

    const messageDiv = document.getElementById('messages');
    if(errorMessages) {
        messageDiv.classList.add('errorMsg');
        messageDiv.innerHTML = '<h4>Error(s)</h4><ul>' + errorMessages + '</ul>';
        return false;
    } else {
        messageDiv.classList.add('successMsg');
        messageDiv.innerHTML = '<img src="https://external-
content.duckduckgo.com/iu/?u=https%3A%2F%2Fvectorified.com%2Fimages%2Fdelivery-icon-
1.png&f=1&nofb=1&ipt=0b8b146fc9e0d44d2ecbee5dd2ef09840d9201c3ad97ecf97801a396c206597b&
ipo=images"><h4>Your order was successful! Thank you very much.</h4>';
        return true;
    }


    return true;
}

function toggleVisibility(id, element) {
    var row = document.getElementById(id);
    if (row.style.display === "none") {
        row.style.display = "table-row";
        element.innerHTML = '<i class="fa fa-eye-slash"></i> Hide Details';
    } else {
        row.style.display = "none";
        element.innerHTML = '<i class="fa fa-eye"></i> Show Details';
    }
}
```

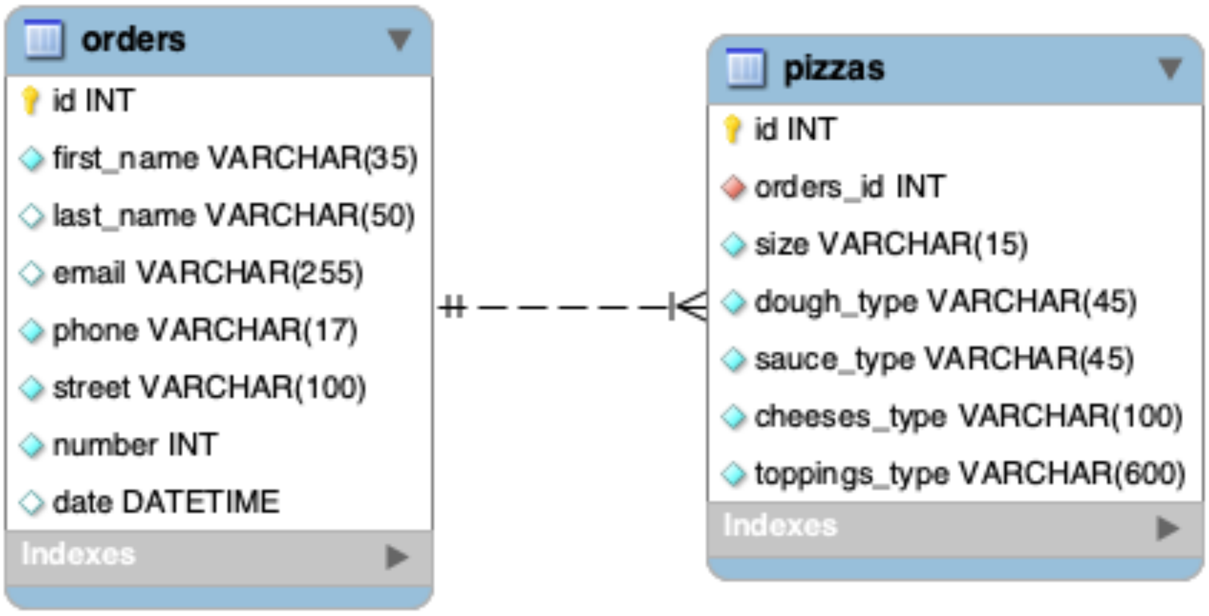
public/css/style.css

```css
:root{
    --default-bg-color: #FFFFFF;
    --primary-font-color: #00000;
    --primary-color :#B61600;
    --secondary-color:#FBAD89;
    --tertiary-color:#FFAC0E;
    --page-max-width: 1024px;
}

body{
    padding-left: 1vw;
    padding-right: 1vw;
    padding-top: 0px;
    margin: 0;
    background: var(--default-bg-color);
}

h1{
    color: var(--primary-color);
}

hr{
    border-color: var(--tertiary-color);
    margin: 20px;
}

#makePizza{
    width: 75%;
    float:left
}

label {
    margin-top: 15px;
    padding: 10px;
    color: var(--primary-color);
    display: block;
    font-weight: bold;

}

input[type=number]{
    width: 100px;
    padding: 10px;
    border: 1px solid var(--primary-color);
    border-radius: 4px;
    box-sizing: border-box;
    resize: vertical;
}

input[type=text]{
    width: 100%;
    padding: 10px;
    border: 1px solid var(--primary-color);
    border-radius: 4px;
    box-sizing: border-box;
    resize: vertical;
}

button {
    width: 100%;
    padding: 10px;
    border-radius: 5px;
    font: inherit;
    margin-top: 20px;
    font-weight: bold;
    outline: 2px solid var(--primary-color);
    background-color: var(--tertiary-color);
    font-weight: bold;
    color: var(--primary-color);
}

button:hover {
    outline: 2px solid var(--primary-color);
    background-color: var(--secondary-color);
    font-weight: bold;
}

.formGrid {
    margin-left: 20px;
    width: 20%;
    float:left
}

.pizzaContainer{
    margin: 15px;
    padding: 15px;
    border-style: solid;
    border-radius: 5px;
    border-color: var(--tertiary-color);
}
```

```css
.doughContainer label,
.sauceContainer label,
.cheeseContainer label,
.toppingsContainer label{
    display: inline;
    margin-left: 10px;
    font-weight: normal;
}

main{
    margin-top: 5px;
    width: 80vw;
    padding: 20px;
    border-radius: 5px;
}

.errorMsg{
    border-color: red;
    background-color: lightpink;
    color: red;
    padding: 10px;
    margin: 15px;
    border-radius: 5px;
}

.successMsg{
    border-color: green;
    background-color: lightgreen;
    color: green;
    padding: 10px;
    margin: 15px;
    border-radius: 5px;
}

.successMsg img{
    width: 50px;
    float: left;
}

/* Style the navigation bar */
nav {
    width: 100%;
    background-color: var(--primary-color);
    overflow: auto;
}

/* Navbar links */
nav a {
    float: left;
    text-align: center;
    padding: 12px;
    color: white;
    text-decoration: none;
    font-size: 17px;
}

/* Navbar links on mouse-over */
nav a:hover {
    background-color: var(--secondary-color)
}

/* Current/active navbar link */
.active {
    background-color: var(--tertiary-color);
}


table {
    width: 100%;
    border-collapse: collapse;
}

table, th, td {
    border: 1px solid black;
}
```

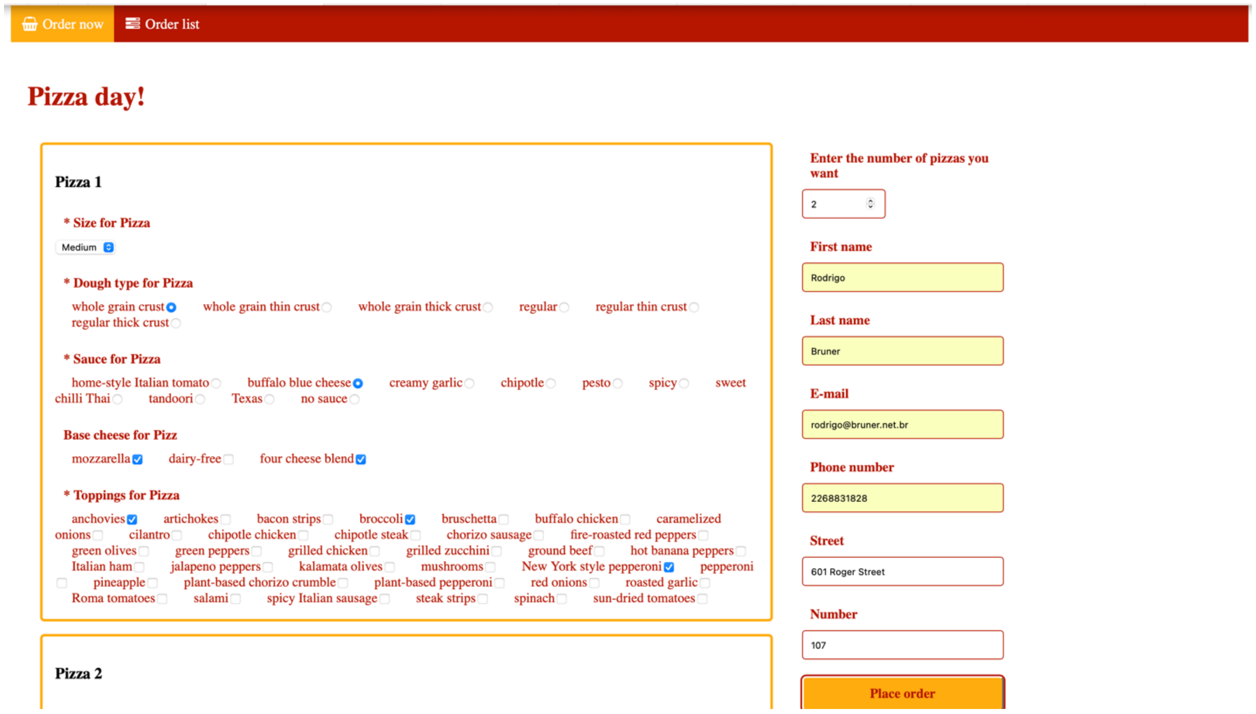
Database

```
-- -----------------------------------------------------
-- Schema pizzaria
-- -----------------------------------------------------
CREATE SCHEMA IF NOT EXISTS `pizzaria` DEFAULT CHARACTER SET utf8 ;
USE `pizzaria` ;

-- -----------------------------------------------------
-- Table `pizzaria`.`orders`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `pizzaria`.`orders` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `first_name` VARCHAR(35) NOT NULL,
  `last_name` VARCHAR(50) NULL,
  `email` VARCHAR(255) NULL,
  `phone` VARCHAR(17) NOT NULL,
  `street` VARCHAR(100) NOT NULL,
  `number` INT NOT NULL,
  `date` DATETIME NULL DEFAULT NOW(),
  PRIMARY KEY (`id`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `pizzaria`.`pizzas`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `pizzaria`.`pizzas` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `orders_id` INT NOT NULL,
  `size` VARCHAR(15) NOT NULL,
  `dough_type` VARCHAR(45) NOT NULL,
  `sauce_type` VARCHAR(45) NOT NULL,
  `cheeses_type` VARCHAR(100) NOT NULL,
  `toppings_type` VARCHAR(600) NOT NULL,
  PRIMARY KEY (`id`),
  INDEX `fk_pizzas_orders_idx` (`orders_id` ASC) VISIBLE,
  CONSTRAINT `fk_pizzas_orders`
    FOREIGN KEY (`orders_id`)
    REFERENCES `pizzaria`.`orders` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```



Screenshots

Bellow some screenshots



Pizza day!

Pizza 1

* Size for Pizza

Medium

* Dough type for Pizza

[✓ whole grain crust] whole grain thin crust whole grain thick crust regular regular thin crust regular thick crust

* Sauce for Pizza

home-style Italian tomato [✓ buffalo blue cheese] creamy garlic chipotle pesto spicy sweet chilli Thai tandoori Texas no sauce

Base cheese for Pizz

[✓ mozzarella] dairy-free [✓ four cheese blend]

* Toppings for Pizza

[✓ anchovies] artichokes bacon strips [✓ broccoli] bruschetta buffalo chicken caramelized onions cilantro chipotle chicken chipotle steak chorizo sausage fire-roasted red peppers green olives green peppers grilled chicken grilled zucchini ground beef hot banana peppers Italian ham jalapeno peppers kalamata olives mushrooms [✓ New York style pepperoni] pepperoni pineapple plant-based chorizo crumble plant-based pepperoni red onions roasted garlic Roma tomatoes salami spicy Italian sausage steak strips spinach sun-dried tomatoes

Pizza 2

Enter the number of pizzas you want

2

First name

Rodrigo

Last name

Bruner

E-mail

rodrigo@bruner.net.br

Phone number

2268831828

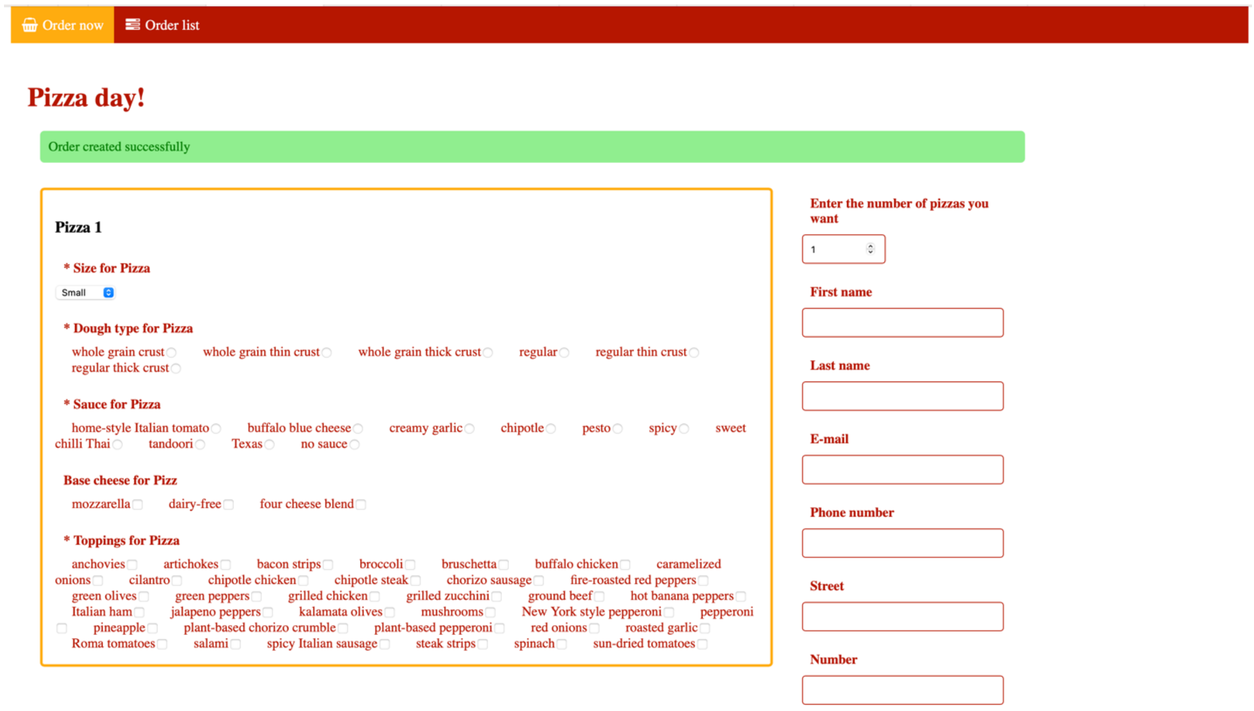
Street

601 Roger Street

Number

107

Note. Form to order



Order now Order list

Pizza day!

Order created successfully

Pizza 1

* Size for Pizza

Small

* Dough type for Pizza

whole grain crust whole grain thin crust whole grain thick crust regular regular thin crust regular thick crust

* Sauce for Pizza

home-style Italian tomato buffalo blue cheese creamy garlic chipotle pesto spicy sweet chilli Thai tandoori Texas no sauce

Base cheese for Pizz

mozzarella dairy-free four cheese blend

* Toppings for Pizza

anchovies artichokes bacon strips broccoli bruschetta buffalo chicken caramelized onions cilantro chipotle chicken chipotle steak chorizo sausage fire-roasted red peppers green olives green peppers grilled chicken grilled zucchini ground beef hot banana peppers Italian ham jalapeno peppers kalamata olives mushrooms New York style pepperoni pepperoni pineapple plant-based chorizo crumble plant-based pepperoni red onions roasted garlic Roma tomatoes salami spicy Italian sausage steak strips spinach sun-dried tomatoes

Enter the number of pizzas you want

1

First name

Last name

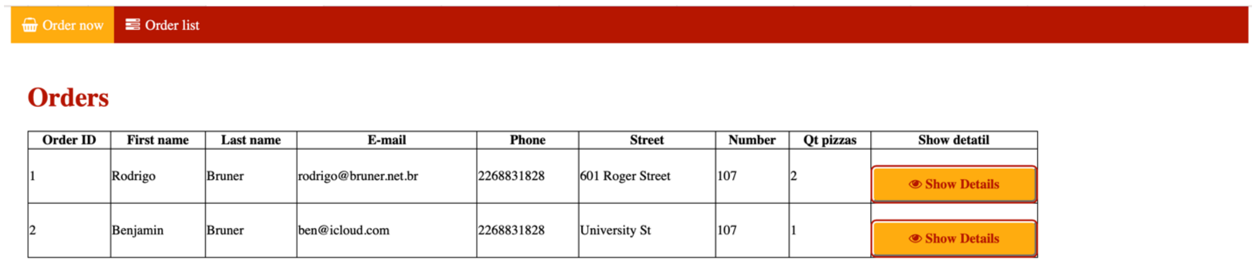
E-mail

Phone number

Street

Number

Note. Order saved

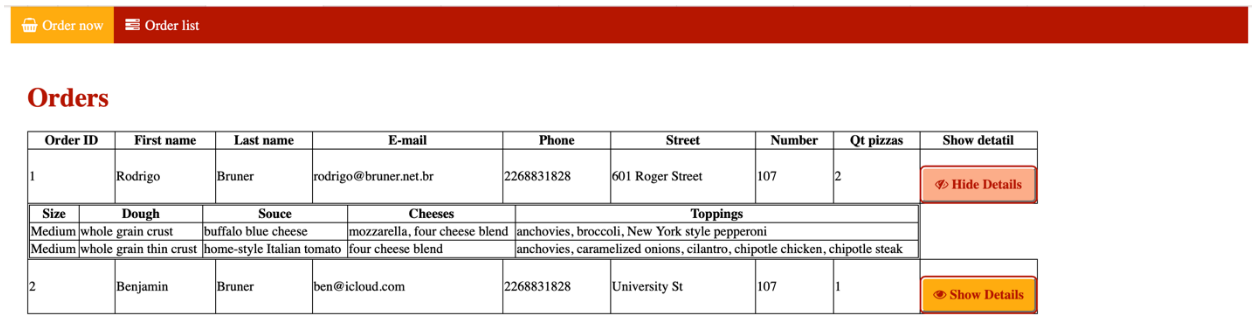


Order now Order list

Orders

Order ID	First name	Last name	E-mail	Phone	Street	Number	Qt pizzas	Show detatil
1	Rodrigo	Bruner	rodrigo@bruner.net.br	2268831828	601 Roger Street	107	2	Show Details
2	Benjamin	Bruner	ben@icloud.com	2268831828	University St	107	1	Show Details

Note. List of the saved orders

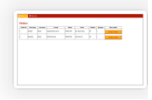


Orders

Order ID	First name	Last name	E-mail	Phone	Street	Number	Qt pizzas	Show detatil
1	Rodrigo	Bruner	rodrigo@bruner.net.br	2268831828	601 Roger Street	107	2	Hide Details

Size	Dough	Souce	Cheeses	Toppings
Medium	whole grain crust	buffalo blue cheese	mozzarella, four cheese blend	anchovies, broccoli, New York style pepperoni
Medium	whole grain thin crust	home-style Italian tomato	four cheese blend	anchovies, caramelized onions, cilantro, chipotle chicken, chipotle steak

Order ID	First name	Last name	E-mail	Phone	Street	Number	Qt pizzas	Show detatil
2	Benjamin	Bruner	ben@icloud.com	2268831828	University St	107	1	Show Details



Note. Detail of the order.

Install and run

1. Clone the repository:

   ```
   git clone git@github.com:rodrigobruner/pizzaria.git
   ```

2. Navigate to the project directory:

   ```
   cd pizzaria
   ```

3. Configure the Apacehe web server

   3.1 Create a vhost on your web server pointing to the /public directory

   3.2 Enable the mod_rewrite

   3.3 Restart Apache

4. Configuring the database

   4.1 Create a database on your MySQL server;

   4.2 Select your database and run the /docs/database.sql file to create the tables;

   4.3 Rename the file

   src/lib/connection-sample.php

   to

   src/lib/connection.php

   and set the following parameters:

   ```
   // Database settings
   private static $host = '[SERVER]';
   private static $database = '[DATABASE]';
   private static $username = '[USERNAME]';
   private static $password = '[PASSWORD]';
   ```