**Project Documentation: Online Order Management System**

**Developed by:** Dijanira  
**GitHub Repository:**

**Table of Contents**

1. Introduction
2. Project Requirements
3. General Project Structure
   * Technologies Used
   * Folder Structure
4. Database
   * Table Structures
   * Table Relationships
5. Implemented Features
   * Home Page
   * Registration and Login
   * User Roles
   * Data Display
   * Contact Form
   * Stored Messages
6. RESTful API
   * Available Endpoints
   * Testing with cURL and Postman
7. GitHub Usage
   * Commit Structure
   * Repository Organization
8. Screenshots
9. Conclusion
10. Future Improvements

# 1. Introduction

This project was developed to showcase skills in modern web application development, including user authentication, database integration, and RESTful API implementation. The system efficiently manages orders with defined roles for administrators, users, and visitors.

# Project Objectives

* Build a functional and interactive order management system.
* Demonstrate concepts like authentication, authorization, and data manipulation.
* Use best practices for version control and documentation.

# **2. Project Requirements**

**Necessary Features**

1. An attractive home page introducing the company.
2. User registration and login with role-based access control.
3. Display data stored in three different database tables.
4. A functional contact form with server-side validation.
5. Implement a RESTful API with full CRUD operations.
6. Version control on GitHub with at least five meaningful commits.
7. Detailed documentation with screenshots explaining each feature.

# 3. General Project Structure

## 3.1 Technologies Used

* **Frontend:** HTML5, CSS3, Bootstrap
* **Backend:** Java (Spring Boot), RESTful API
* **Database:** MySQL
* **Version Control:** GitHub

## 3.2 Folder Structure

Uma imagem com texto, captura de ecrã, software

Descrição gerada automaticamente

Uma imagem com texto, captura de ecrã, Tipo de letra, design

Descrição gerada automaticamente

# 4. Database

## 4.1 Table Structures

The project uses four main tables:

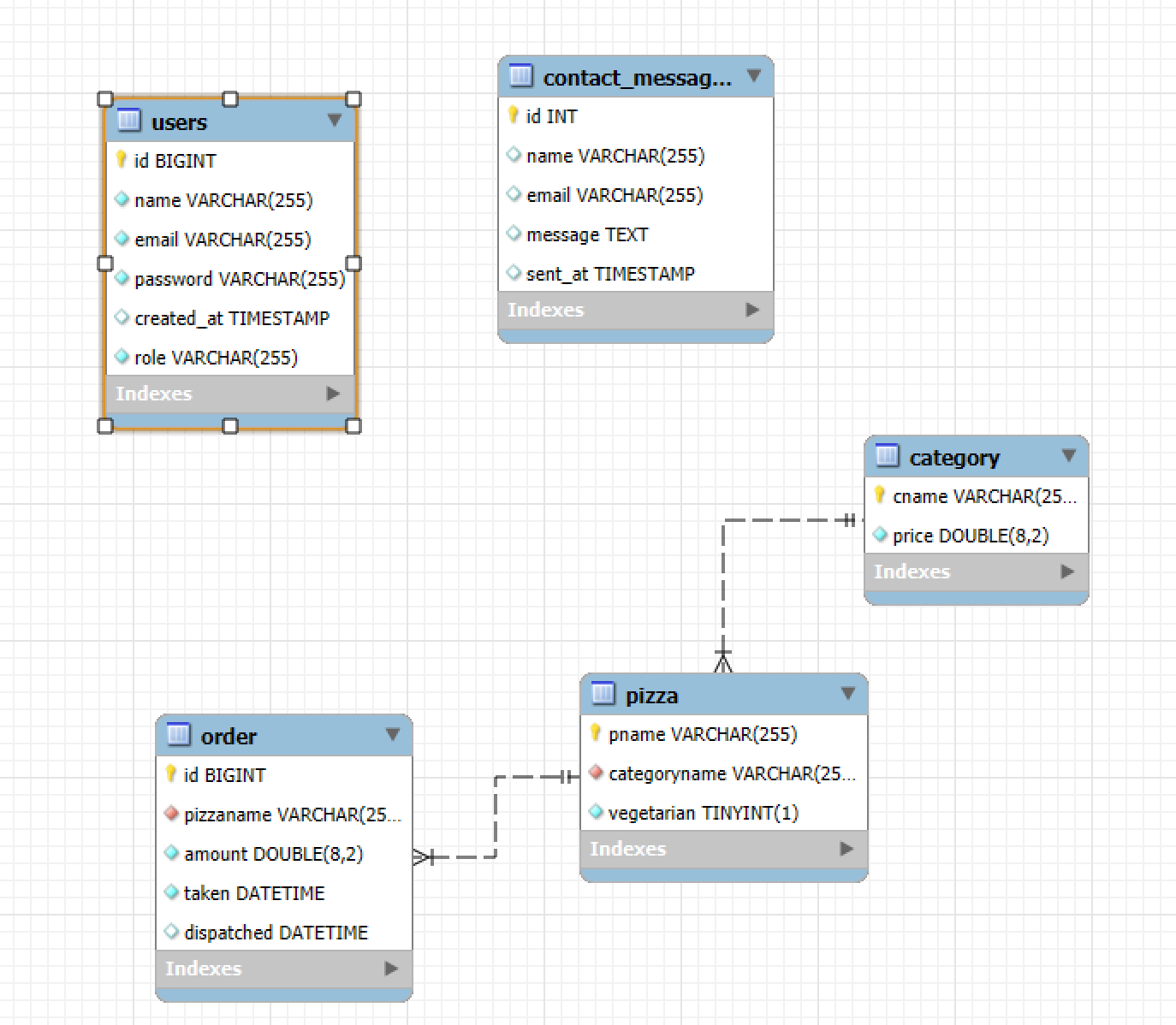
1. **Users:**
   * **Description:** Stores user information.
   * **Fields:**
     + id (PK, auto\_increment): Unique identifier.
     + name: User's name.
     + email (UNIQUE): User's email address.
     + password: Encrypted password.
     + role: User role (Admin, User, Visitor).
     + created\_at: Account creation timestamp.
2. **Order:**
   * **Description:** Stores user order details.
   * **Fields:**
     + id (PK): Unique identifier.
     + pizzaname: Name of the pizza.
     + amount: Ordered quantity.
     + taken: Status (if prepared).
     + dispatched: Dispatch status.
3. **Pizza:**
   * **Description:** List of available pizzas.
   * **Fields:**
     + id (PK):
     + pname: Name of the pizza( Unique identifier).
     + categoryname: Pizza category.
     + vegetarian: Indicates if the pizza is vegetarian (boolean).
4. **Category:**
   * **Description:** Classification of pizzas.
   * **Fields:**
     + cname (PK): Category name.
     + price: Average price of the category.

## 4.2 Table Relationships

* **Orders → Users:** A user can place multiple orders (1

).

* **Pizzas → Categories:** Each pizza belongs to one category (N:1).

4.1 Diagram   
  


# 5. Implemented Features

## 5.1 Home Page

The home page introduces the purpose of the system and includes navigation links for registration, login, and other key pages.

## 5.2 Registration and Login

* **Registration:** Allows new users to create an account.
* **Login:** Grants access to the system with credential validation.

## 5.3 User Roles

User roles determine available features and menu visibility:

* **Admin:** Access to all features, including user management.
* **User:** Can place orders and view statuses.
* **Visitor:** Limited access to public pages.

# 6. RESTful API

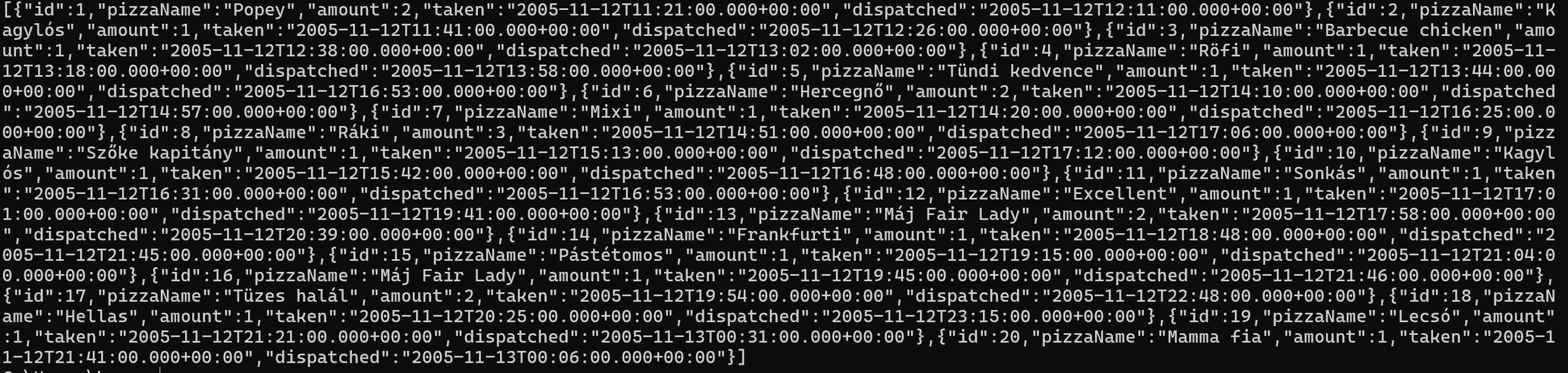
**Available Endpoints**

* **Users:**
  + GET /api/users - Retrieves all users.
  + POST /api/users - Adds a new user.
* **Orders:**
  + GET /api/orders - Retrieves all orders.
  + POST /api/orders - Creates a new order.

# 7. Testing with cURL and Postman

## cURL Test Example

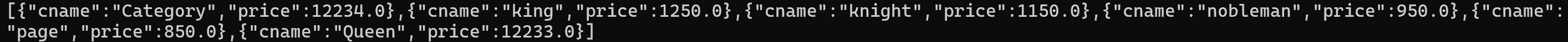
curl -X GET <http://localhost:8080/api/orders>



curl -X GET <http://localhost:8080/api/pizzas>

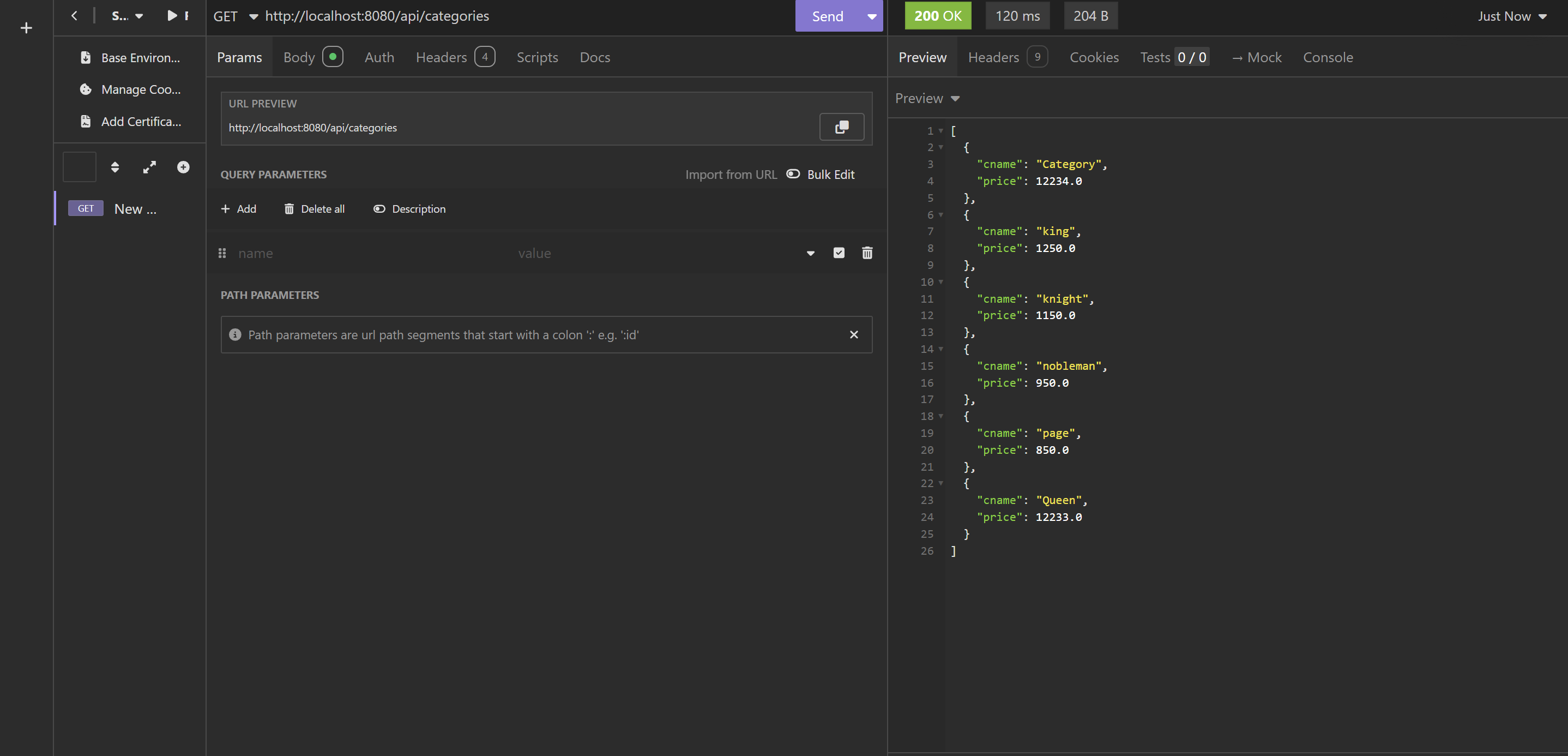


curl -X GET <http://localhost:8080/api/categories>



## Postman Test Example

List Category

  
List Pizza  
Uma imagem com texto, captura de ecrã, software, Software de multimédia

Descrição gerada automaticamente

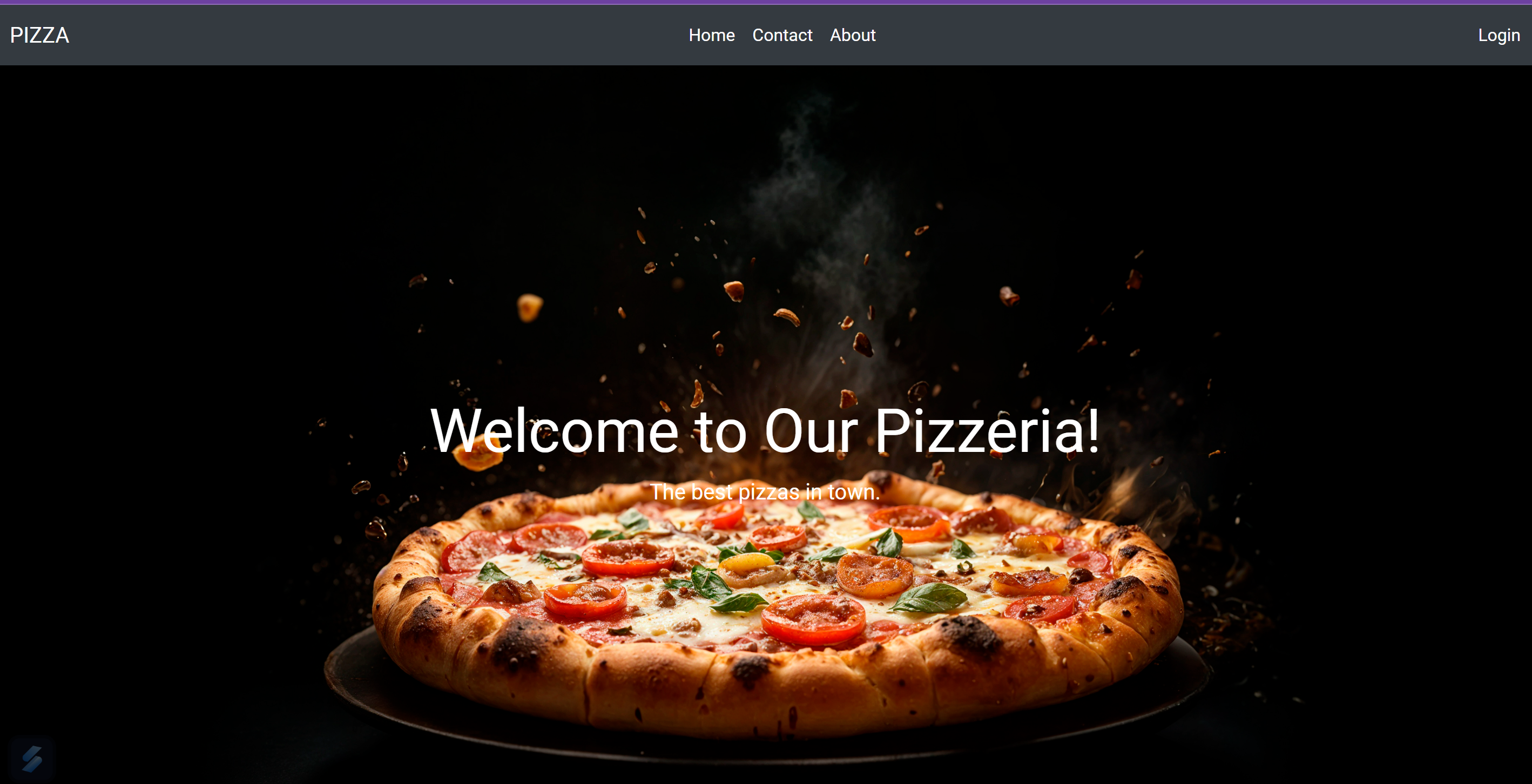
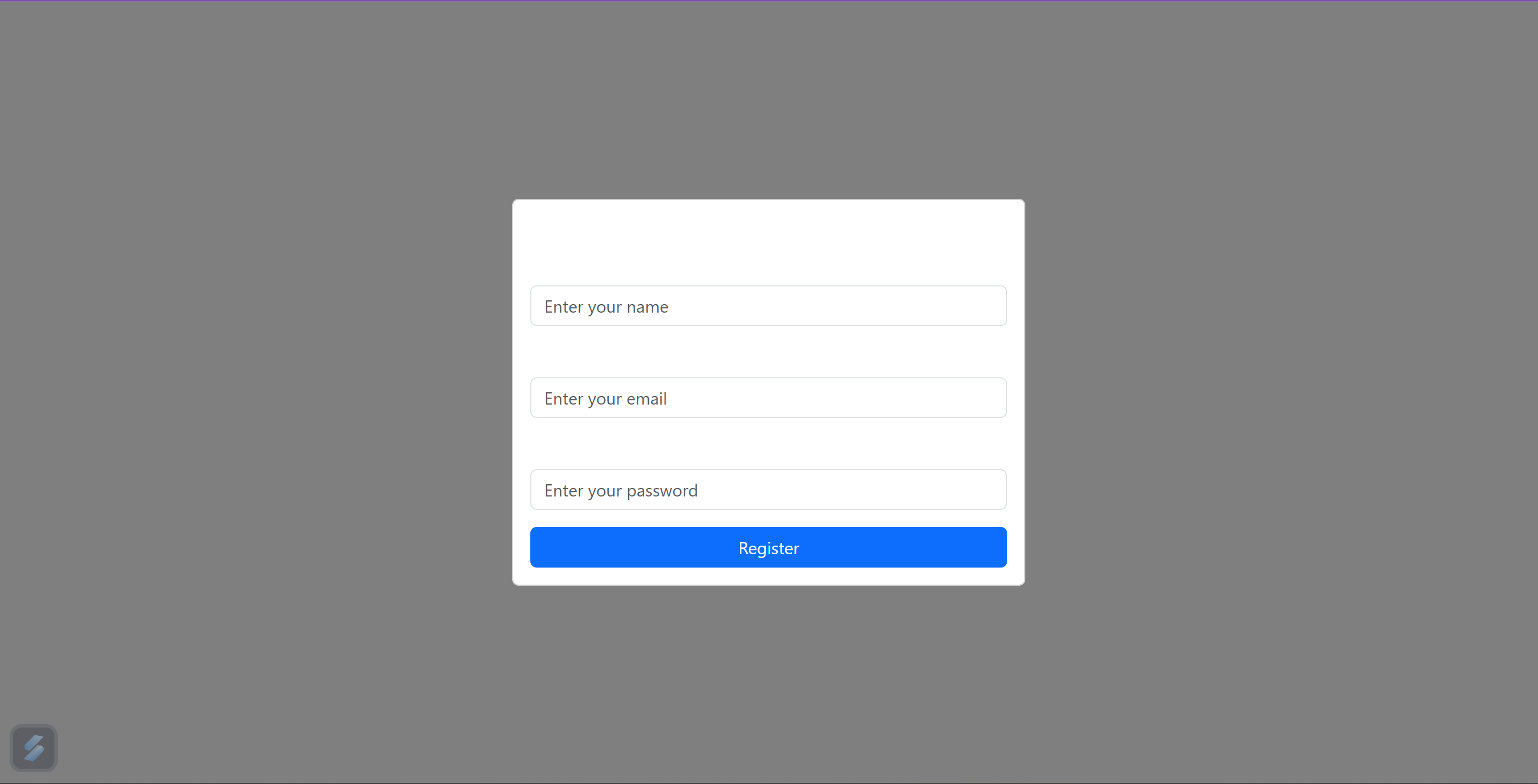
List Orders  
Uma imagem com texto, captura de ecrã, software, Software de multimédia

Descrição gerada automaticamente

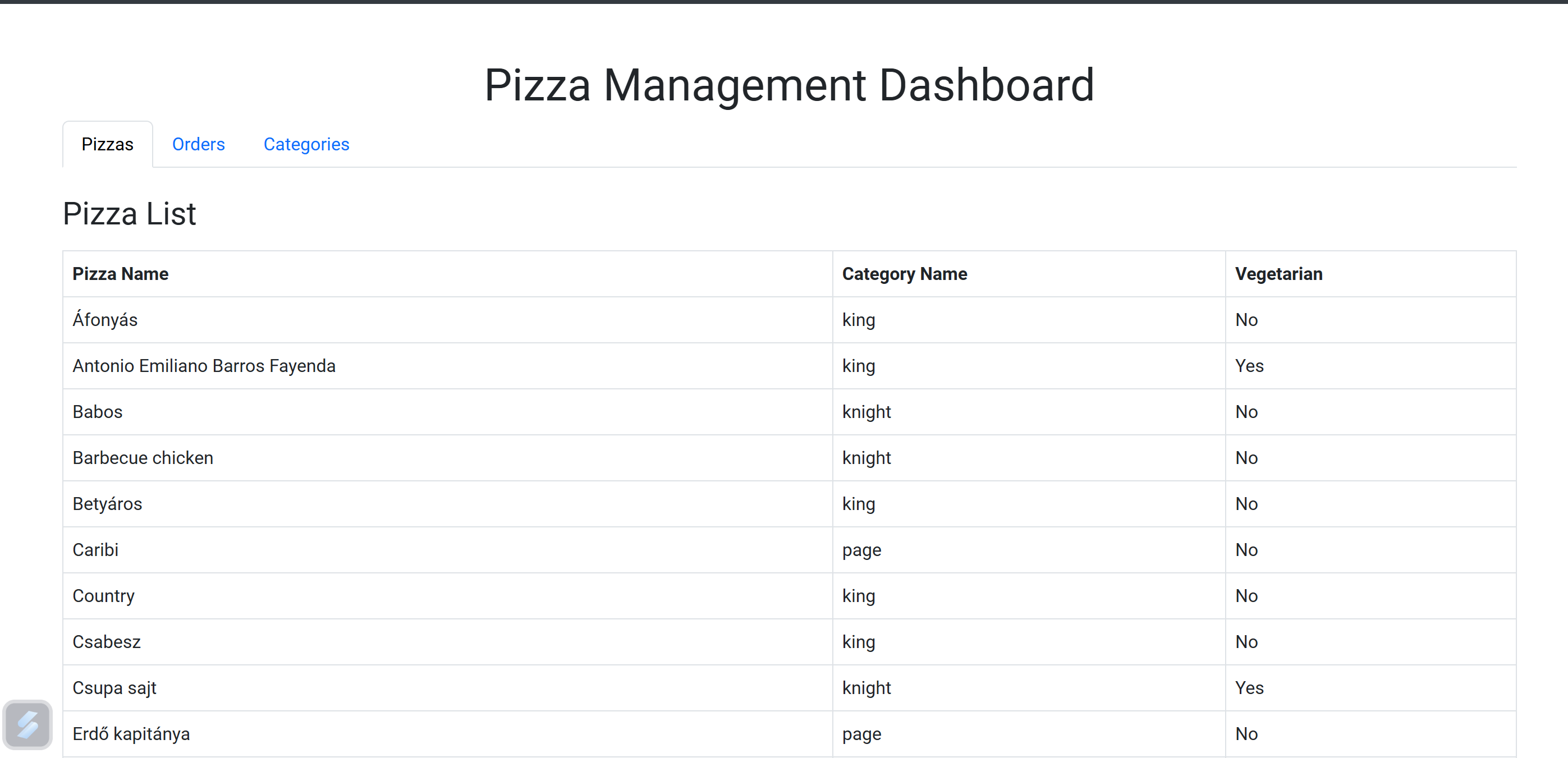
New Category  
Uma imagem com captura de ecrã, software, Software de multimédia, Software gráfico

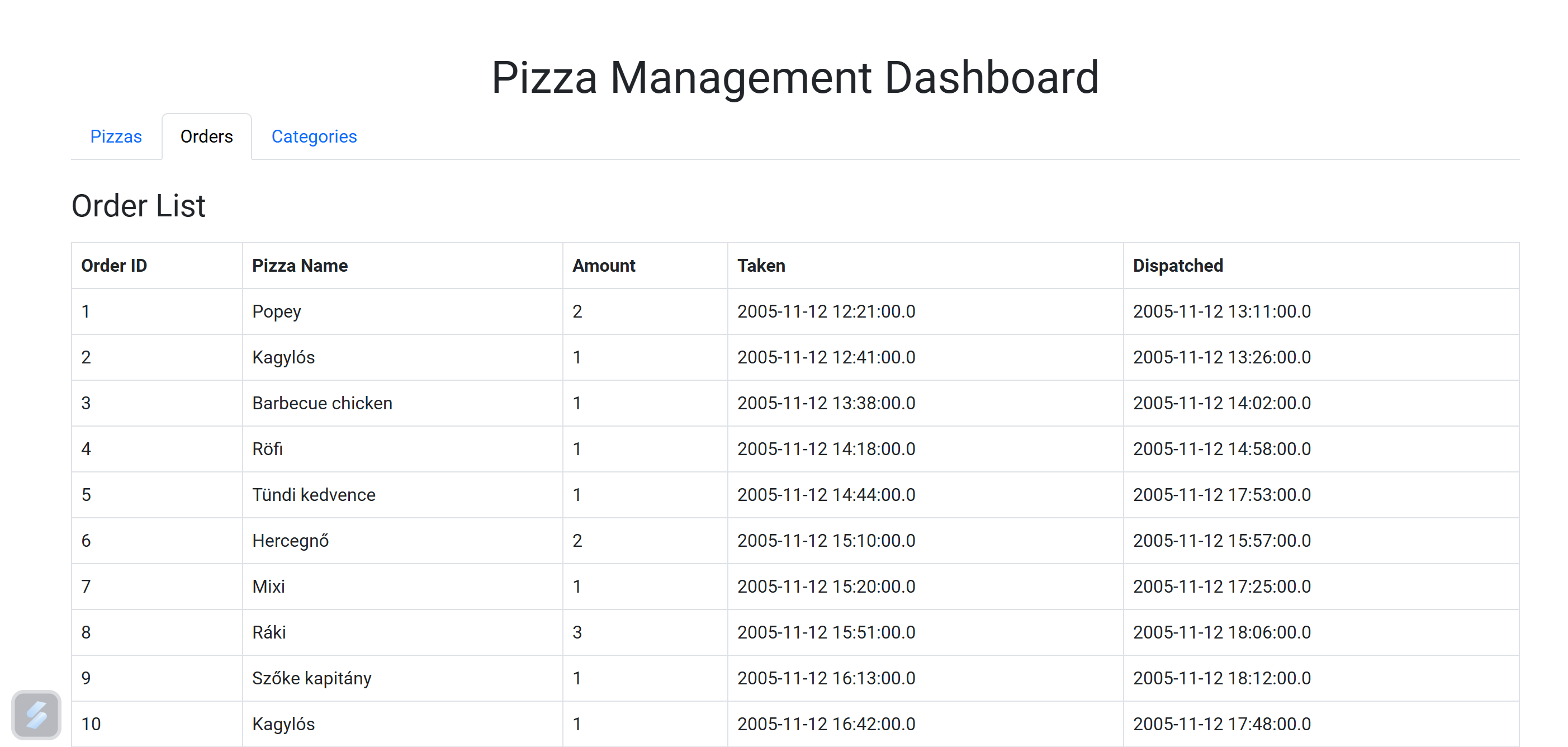
Descrição gerada automaticamente

# 8. Screenshots

1. Home Page  
     
   
2. Registration Form  
   

## Display





**9. GitHub Usage**

* Each feature was implemented with meaningful commits and descriptive messages.

**10. Conclusion and Future Improvements**

This project demonstrates advanced skills in web development, but enhancements such as detailed reporting and integration with external APIs can further improve functionality.