Specification of software requirements

Project: Industrial dining room

Document file

Date	Reviewn	Author	Datafied quality dep.
		Aguirre Gonzalez Marcos Brito Canales Genesis Jacqueline Llamas Zamudio Jabes Kimberly Itzel Soto Garcia	

Document validated by the parties on date:

By the customer	By the supplying company
Fdo. D./ Dña	Fdo. D./Dña

INDEX

1 Introduction

This document will present the Software Requirements Specifications (ERS) for the Industrial Dining room project. These specifications have been structured with the instructions of the IEEE 830 document

1.1 Purpose

The project seeks to implement a solution for the efficient management of operations in industrial dining rooms. This includes the creation of profiles for staff, as well as the coordination of resources in the management of the dining room. The proposal is designed to optimize all services, from menu planning and inventory management to cost control.

1.2 Scope

This document is aimed at factories seeking to improve their performance and management of industrial dining room services through a comprehensive digital system. The objective is to implement a solution that optimizes the management of menus, reservations, and the administration of costs and discounts, ensuring greater efficiency in the process of feeding employees.

1.3 Personnel involved

Name	Marcos Aguirre Gonzalez
Role	Database, Documentation
Professional category	TSU-Software Development
Responsibilities	Database creation, Software documentation
Contact information	0323106322@ut-tijuana.edu.mx

Name	Brito Canales Genesis Jacqueline
Role	Programmer, Documentation
Professional category	TSU-Software Development
Responsibilities	Software Programming creation, Software documentation
Contact information	0323105958@ut-tijuana.edu.mx

Name	Jabes Llamas Zamudio
Role	Leader, Operating Systems, Documentation
Professional category	TSU-Software Development
Responsibilities	,Task assignment, Software documentation
Contact information	0323105909@ut-tijuana.edu.mx

Name	Kimberly Itzel Soto Garcia
Role	Programmer, Documentation
Professional category	TSU-Software Development
Responsibilities	Software Programming, Software Documentation
Contact information	0323105982@ut-tijuana.edu.mx

1.4 Definitions, acronyms and abbreviations

Name	Description
ERS	Software Requirements Specification
RF	Functional requirement
RNF	Non-functional requirement

1.5 References

Reference	Title
Standard IEEE 830	IEEE
Fontela, C. (2012). ISBN: 978-987-1609-22-2	UML Software Model for Professionals.

1.6 Summary

In this first section, an introduction to the document is provided, the project objectives are established, and an overview of the system is presented. The purpose of the system and its scope are defined, providing a clear context for the development of the project.

The second section details the specific actions required, the constraints that must be taken into account and the dependencies that affect the development of the system. This describes aspects that could influence the development process, including technical limitations, external requirements, and conditions that must be met.

The third section lists all the program requirements. This section specifies the required functionality, performance criteria, and characteristics that the system must meet to meet user needs.

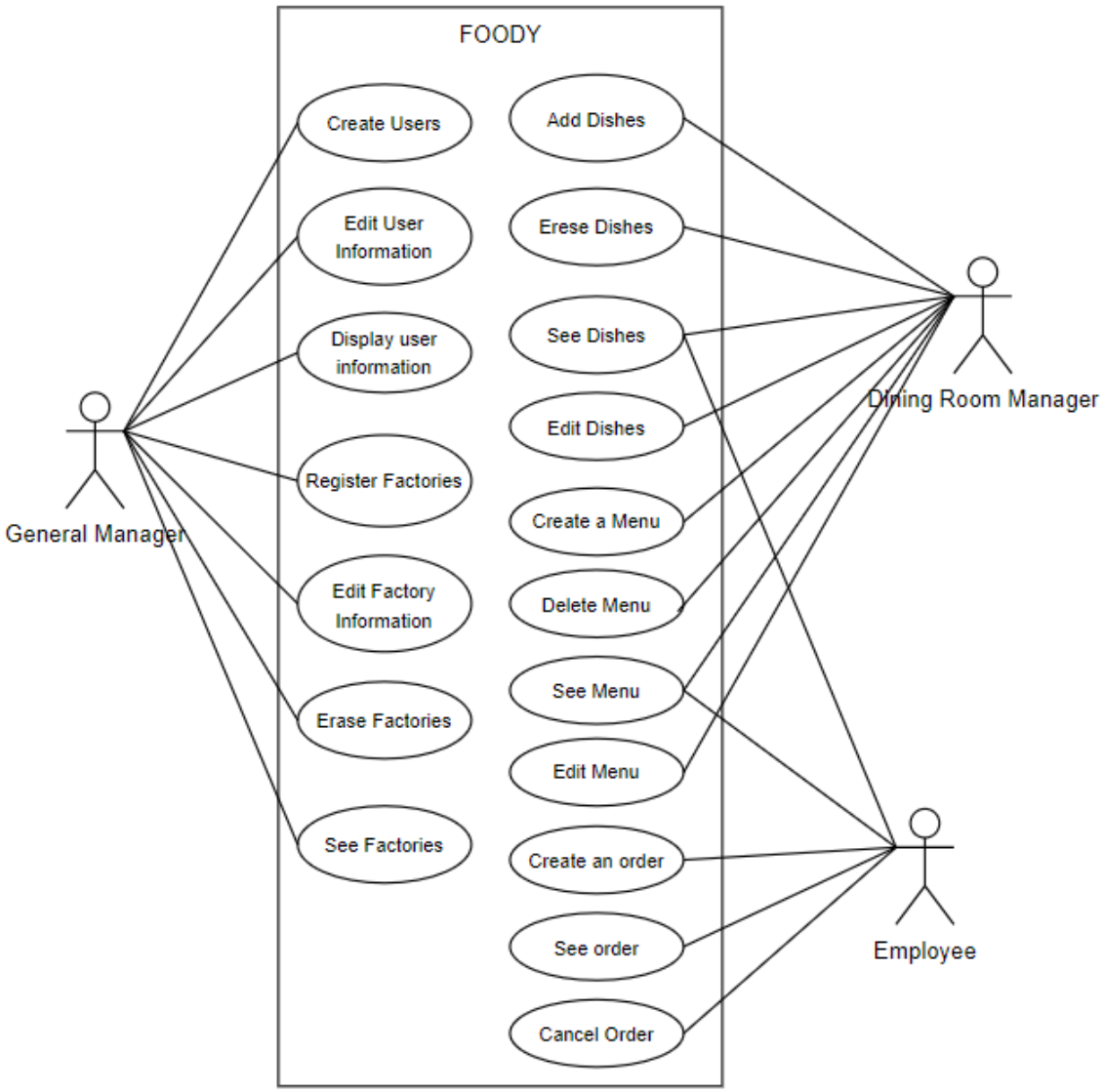
2 Description general

2.1 Perspective of product

The industrial dining room management system is integrated into the factory's ecosystem of management and human resources tools, with the goal of improving operational efficiency and employee experience. This product not only facilitates the day-to-day operation of the Dining room service, but is also designed to accommodate future growth.

The system provides a platform for menu management, reservations, and inventory monitoring, enabling seamless interaction between employees, managers, and suppliers. Through an accessible web interface, employees can easily manage their meals, while managers gain visibility and control over internal processes.

2.2 Functionality of the product



2.3 Personal Characteristics

User type	Employee
Training	N/A
Activities	Order dishes, book, view menus and categories

User type	Dining room manager
Training	Chef
Activities	Generate orders to suppliers, add dishes to menus, and manage ingredients.

User type	Program Administrator
Training	Systems engineer or software development technician
Activities	Registering, editing and modifying users and factories

2.4 Restrictions

- interface to be used on a website
- Language and technologies in use HTML, CSS, JAVASCRIPT, PHP and MySQL.
- Portable on Windows and Linux operating system
- The system must be intuitive for the user

2.5 Assumptions and dependencies

- The requirements described here are assumed to be stable.
- On the equipment that is going to be executed, the system must that it meets the requirements for a correct execution.

2.6 Foreseeable evolution of the system

- Advanced security.
- Performance optimization
- Mobile device compatibility

3 Specific requirements

Requirement number	FR01
Requirement name	Developed in Javascript, PHP, CSS and HTML for the web
Description of the requirement	The Software must be developed in JavaScript, php and designed in Css, HTML to be able to be executed on a web page.
Non-functional requirement	
Requirement Priority	☐ High/Essential ☐ Average/Desired ☐ Low/ Optional

Requirement number	FR02
Requirement name	Save information in a database
Description of the requirement	A database must be created that is capable of storing all the necessary data in an orderly manner.
Non-functional requirement	
Requirement Priority	☐ High/Essential ☐ Average/Desired ☐ Low/ Optional

Requirement number	FR03
Requirement name	Enter data
Description of the requirement	The Software must be developed so that the user can enter or select data
Non-functional requirement	
Requirement Priority	☐ High/Essential ☐ Average/Desired ☐ Low/ Optional

Requirement number	FR04
Requirement name	Dish menu management
Description of the requirement	The Software must allow the administrator to add, edit and delete dishes from the menu in addition to displaying the dishes together with their respective data.
Non-functional requirement	
Requirement Priority	☒ High/Essential ☐ Average/Desired ☐ Low/ Optional

Requirement number	FR05
Requirement name	Reservation of dishes
Description of the requirement	Allow employees to view the menu and make reservations for specific days
Non-functional requirement	
Requirement Priority	☒ High/Essential ☐ Average/Desired ☐ Low/ Optional

Requirement number	FR06
Requirement name	Receipt generation
Description of the requirement	A receipt will be generated automatically after an employee makes a reservation and in turn with this same receipt you can pick up your food.
Non-functional requirement	
Requirement Priority	☒ High/Essential ☐ Average/Desired ☐ Low/ Optional

Requirement number	FR07
Requirement name	Login
Description of the requirement	The Software must have a login system for employees and administrators.
Non-functional requirement	
Requirement Priority	☒ High/Essential ☐ Average/Desired ☐ Low/ Optional

Requirement number	FR08
Requirement name	Viewing reservations
Description of the requirement	It must allow the employee to consult and manage their reservations to modify or cancel reservations
Non-functional requirement	
Requirement Priority	☐ High/Essential ☒ Average/Desired ☐ Low/ Optional

Requirement number	NFR01
Requirement name	Usability
Description of the requirement	The interface must be intuitive and easy to use for employees

Requirement Priority	☒ High/Essential ☐ Average/Desired ☐ Low/ Optional

Requirement number	NFR02
Requirement name	Security
Description of the requirement	Personal information and employee data must be protected
Requirement Priority	☒ High/Essential ☐ Average/Desired ☐ Low/ Optional

Requirement number	NFR03
Requirement name	Scalability
Description of the requirement	The system must be able to adapt to a number of employees and dishes without the need for restructuring
Requirement Priority	☒ High/Essential ☐ Average/Desired ☐ Low/ Optional

Requirement number	NFR04
Requirement name	Performance
Description of the requirement	The system will be optimized to carry out the processes efficiently
Requirement Priority	☒ High/Essential ☐ Average/Desired ☐ Low/ Optional

3.1 Common interfaces

3.1.1 User interfaces

The user interface will be designed with a focus on usability, using drop-down menus and clearly labeled buttons. Every time the user navigates between sections, the aim will be to have an interface that is not too overloaded with information.

3.1.2 Hardware interfaces

The industrial dining room management system interacts with various hardware components that will ensure its correct operation. The main hardware interfaces and their characteristics are described below.

- Web server: The system will be hosted on a web server that will run the backend.
- Network devices: These are required since they are used to establish the connection with the server through the Internet.
- Mobile devices: They are the means by which the employee is able to interact with the system.

3.1.3 Software Interfaces

- Windows or Linux operating systems
- Database MySQL

3.1.4 Communication interfaces

The dining room management system requires communication between different modules and services for its correct operation.

- Communication between web server and clients: The web interface of the system will be accessible from web browsers allowing employees and administrators to interact with the system.
- Communication between Web Server and database: The web server will connect to a MySQL database to manage all system information, including reservations, menus, inventories, users, etc.
- Communication between dining room and web service: The Dining room manager will be connected to the web server to process employee reservations and generate electronic receipts.

3.2 Functional requirements

3.2.1 Functional requirement 1

Developed in Javascript, PHP, CSS and HTML for the web: The system is implemented using web technologies mainly JavaScript and Php for the entire area of document functionality and HTML, Css to be able to create interfaces with a good design

3.2.2 Functional requirement 2

Save the information in a database: All information is stored in a database and a secure connection will be established between the software and the database. MySQL syntax will be used

3.2.3 Functional requirement 3

Enter data: Users will be able to enter data into the system through intuitive interfaces. This will include being able to easily interact with the functions as well as implementing data validations to ensure that the information is correct.

3.2.4 Functional requirement 4

Dish menu management: Administrators will be able to manage the dish menu through an interface. This will include functionality to add, edit and delete dishes, as well as categorize dishes and assign prices.

3.2.5 Functional requirement 5

Dish reservations: Employees will be able to make dish reservations through an easy-to-use interface. When selecting a dish, a calendar will be displayed to choose the desired date.

3.2.6 Functional requirement 6

Receipt generation: The system automatically generates a receipt for each reservation made, which will include details such as the selected dish, price, reservation date and employee information.

3.2.7 Functional requirement 7

Login: Users will be able to have their own account where they can place their orders and will have an easy-to-use interface

3.2.8 Functional requirement 8

Reservation View: Employees will be able to view all their past and future reservations in a dedicated section of their profile. The view will include details such as date, reserved dish and reservation status.

3.3 Non-functional requirements

3.3.1 Performance requirements

- The system must be well structured so that it has good performance within the software functions and is optimized.

3.3.2 Security

To ensure that all employee data and information related to orders and reservations is handled securely, measures are implemented to prevent unauthorized access and protect the integrity of the information by ensuring that only authorized users can make changes to the menu.

3.3.3 Reliability

The program will be available 24 hours a day, providing continuous access to workers who require services at any time. To ensure optimal performance and system stability, performance testing and maintenance are performed during non-business days, ensuring that they do not affect regular operation or user experience.

3.3.4 Maintainability

The program will have constant maintenance, ensuring its optimal functioning at all times. In addition, it will be designed to be easily upgradeable, which will allow the addition of new functionalities in an agile manner, ensuring that the system always operates efficiently and remains aligned with user needs.

3.3.5 Portability

The system will be implemented in the LINUX operating system, but it can be used in Windows

3.4 Other requirements

The system will be intuitive and easy to understand for any user.

3.5 Software architecture

3.5.1 Multilayer Architecture

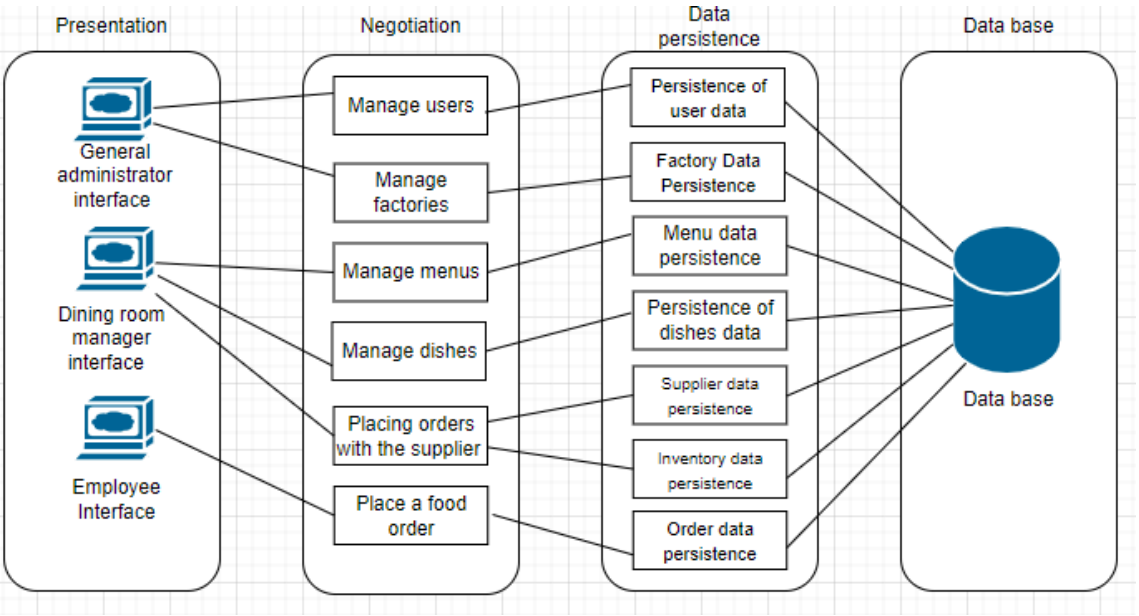
We chose this architecture because it is a design structure that divides an application into ordered subsystems, each with different responsibilities.

A MultiLayer database allows us to have a more robust and easy-to-maintain system, while adapting to changes with greater flexibility.

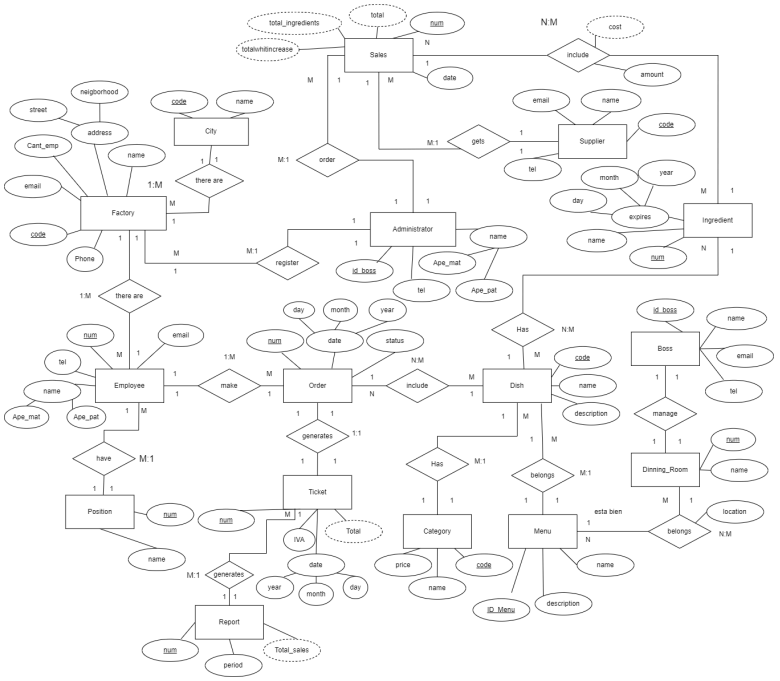
Each layer can be optimized individually, contributing to better load distribution and overall system performance.

With responsibilities divided, the system is easier to maintain. Developers can work on a specific layer without interfering with the others, making it easier to fix bugs or implement improvements.

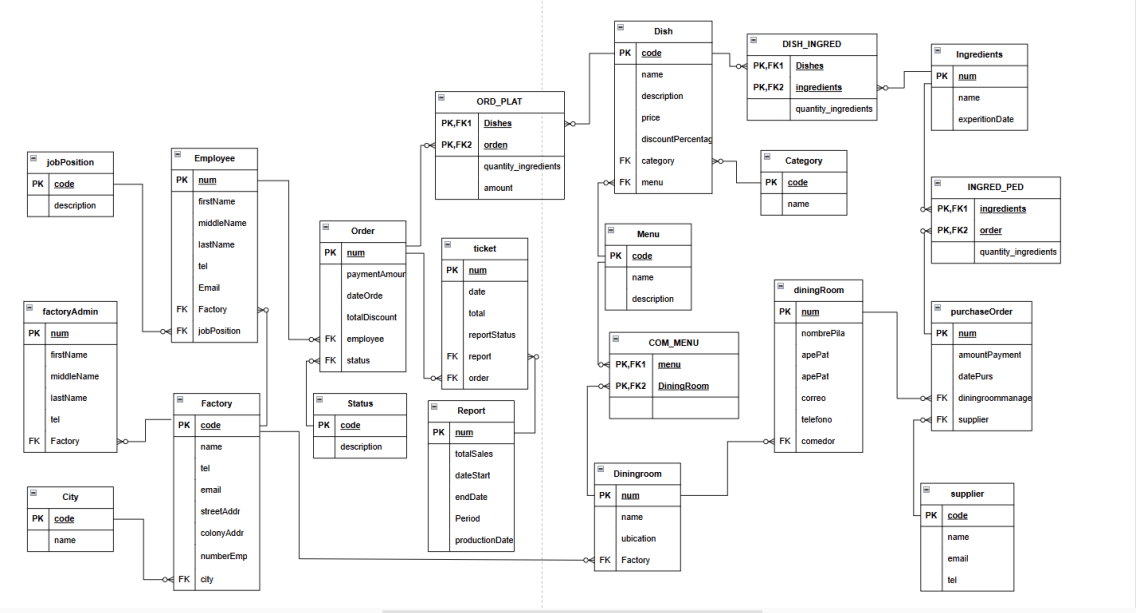
3.5.2 Multilayer Architecture Diagram



3.6 Diagram relationship



3.7 Diagram Entity-relationship diagram



3.8 Use cases

General administrator:

-Register factories
-Register employees
-Modify employee data
-Delete employees

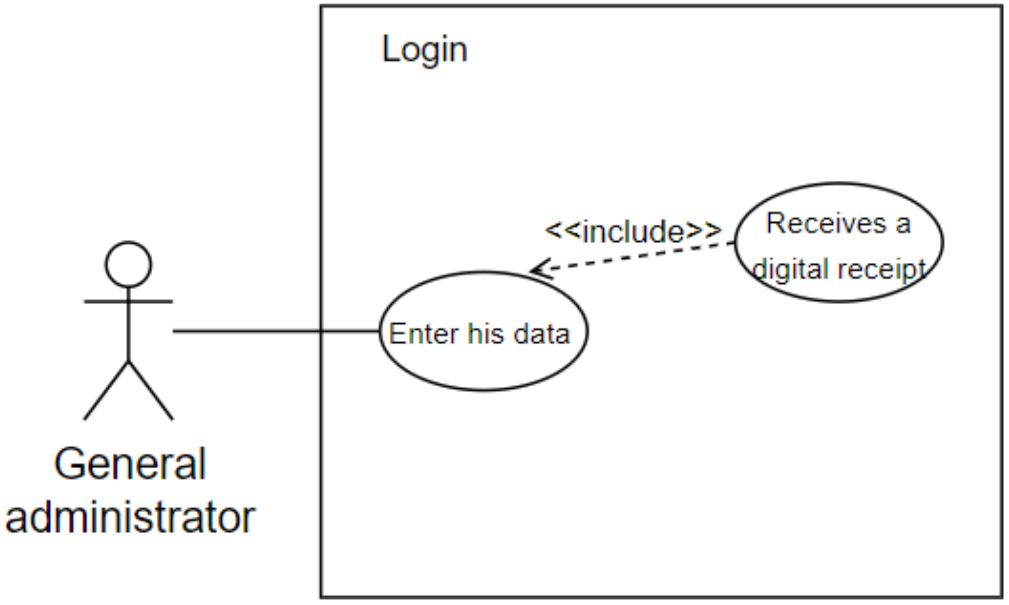
Dining room manager:

- program options
  -Ingredients
    -Add ingredients
    -Update ingredients
    -remove ingredients
  generate an order to the supplier
- add a weekly menu
- modify the menu
-add dishes
-add ingredients
-add description to dishes

Employee :

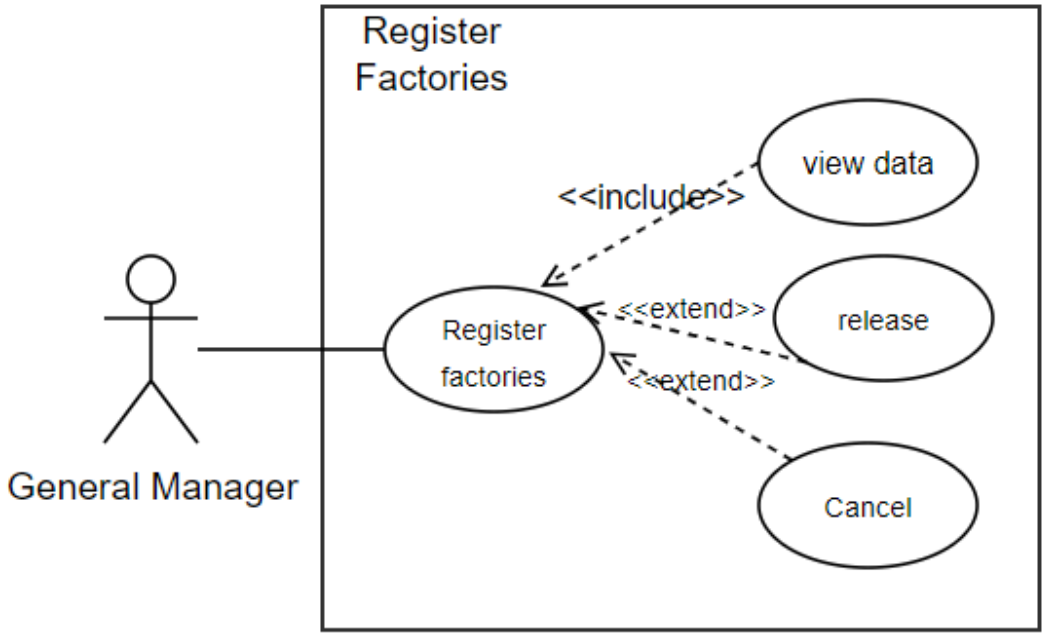
-Order
-Consult an order
-Order history
-cancel an order
-display the weekly menu
  -categories of dishes
    -dishes of each category

Name	Login
Authors	Llamas Zamudio Jabes
Date	21/09/24
Descriptión	Allows General administrator to login to their account
Actors	General administrator
Preconditions	The General administrator manager must be authenticated in the system.
Normal flow	• The employee must enter his or her data • The employee must select the login option
Alternative flow	If the data entered is not correct, the system displays an error message.
Postconditions	The General administrator manager will log in



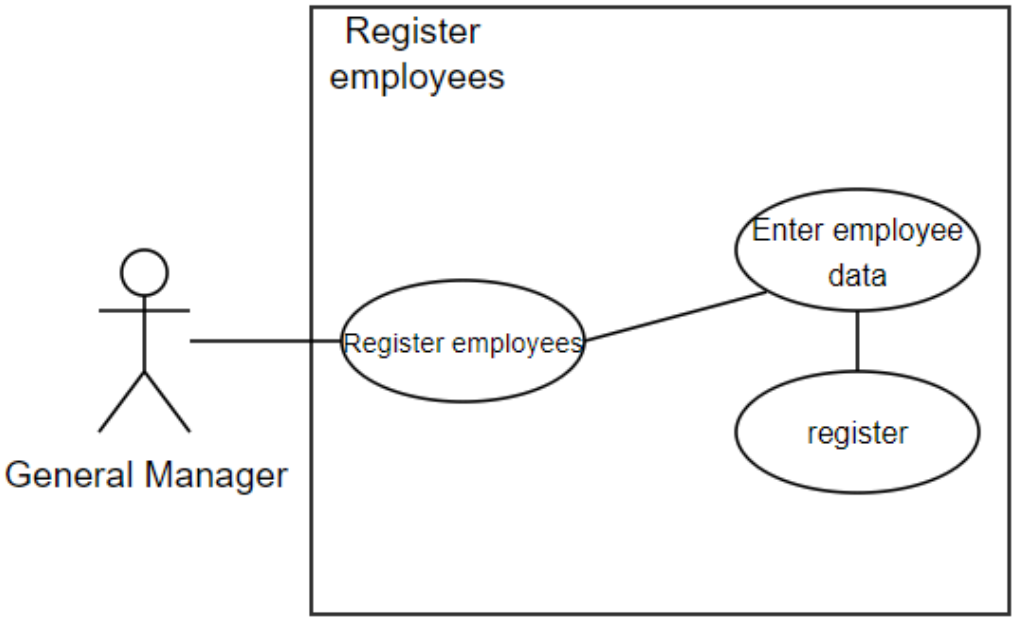
Name	Register Factories
Authors	Llamas Zamudio Jabes
Date	21/09/24
Description	Allows the general administrator to register a new factory in the system.

Actors	General administrator
Preconditions	The administrator must be authenticated in the system.
Normal flow	• The administrator selects “Register Factory”. • He enters the required data (name, address, etc.). • The system validates the information. • The system saves the new factory and displays a confirmation message.
Alternative flow	If the validation fails, the system displays an error message and requests corrections.
Postconditions	The new factory is registered in the system.



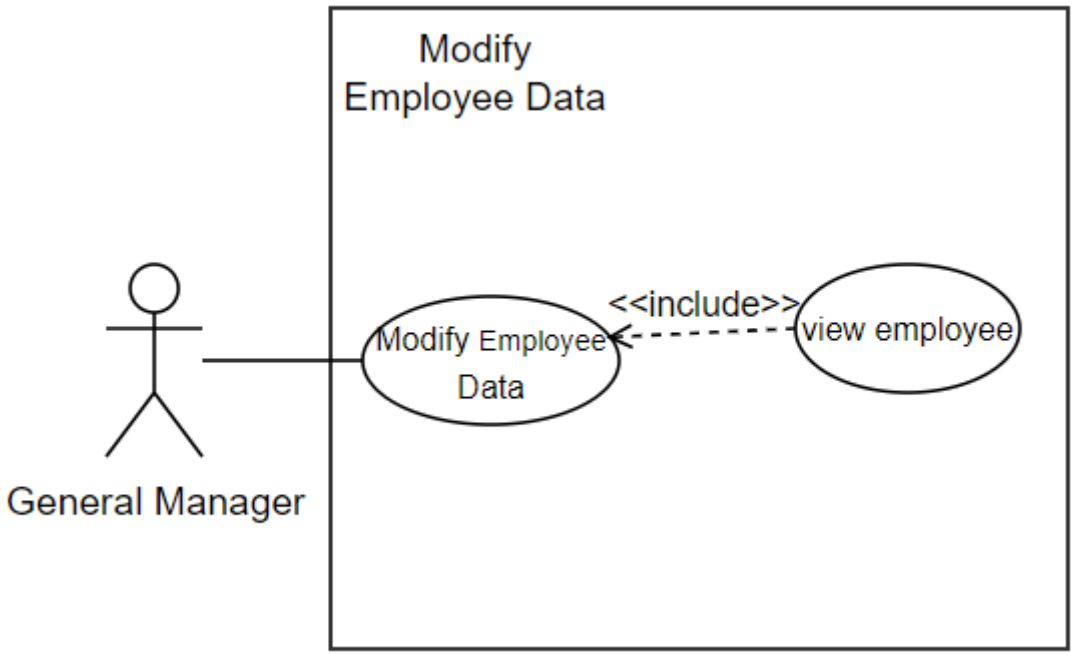
Name	Register employees
Authors	Llamas Zamudio Jabes
Date	21/09/24
Descriptión	Allows the general administrator to add a new employee to the system.
Actors	General Manager
Preconditions	The administrator must be authenticated in the system.

Normal flow	• The administrator selects “Register Employees”. • He/she enters the employee's data (name, email, position). • The system validates the information. • The system saves the new employee and displays a confirmation message.
Alternative flow	If validation fails, the system displays an error message and requests corrections.
Postconditions	The new employee is registered in the system.

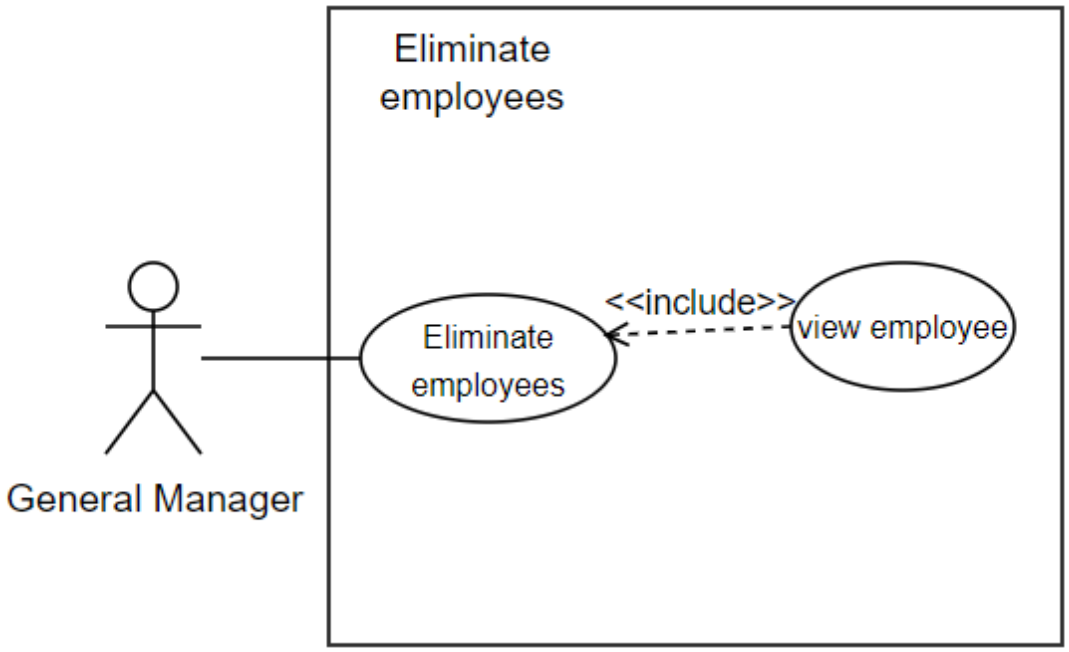


Name	Modify Employee Data
Authors	Llamas Zamudio Jabes
Date	21/09/24
Description	Allows the administrator to modify the information of an existing employee.
Actors	General Manager
Preconditions	The administrator must be authenticated in the system.
Normal flow	• The administrator selects “Modify Employee Data”. • Enter the Id of the employee to be modified. • Make the necessary changes. • The system saves the changes and displays a confirmation message.

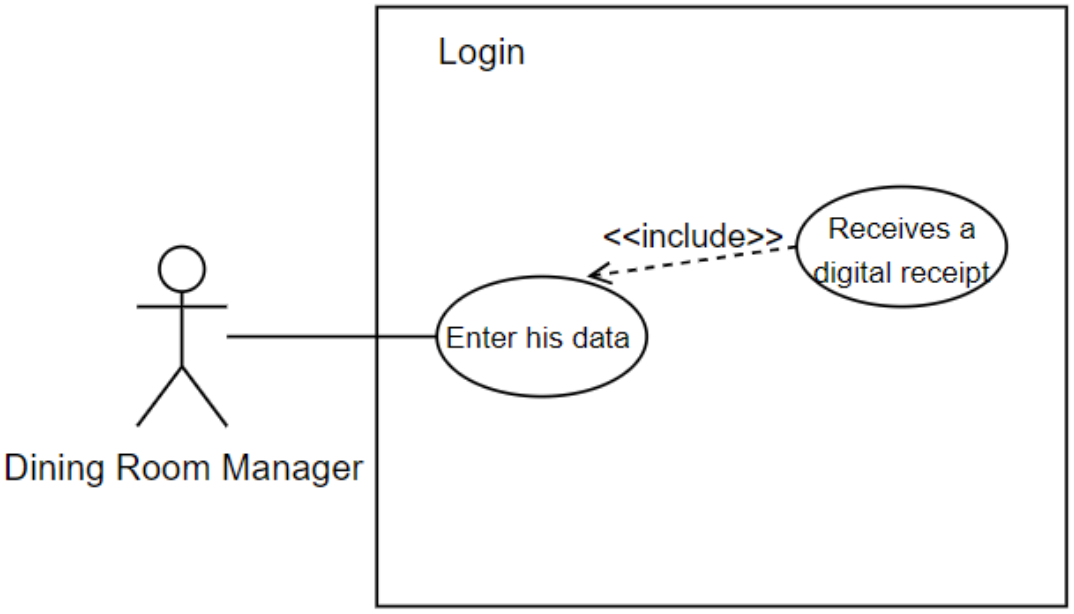
Alternative flow	If the employee is not found, the system displays an error message.
Postconditions	The employee's data is maintained in the system.



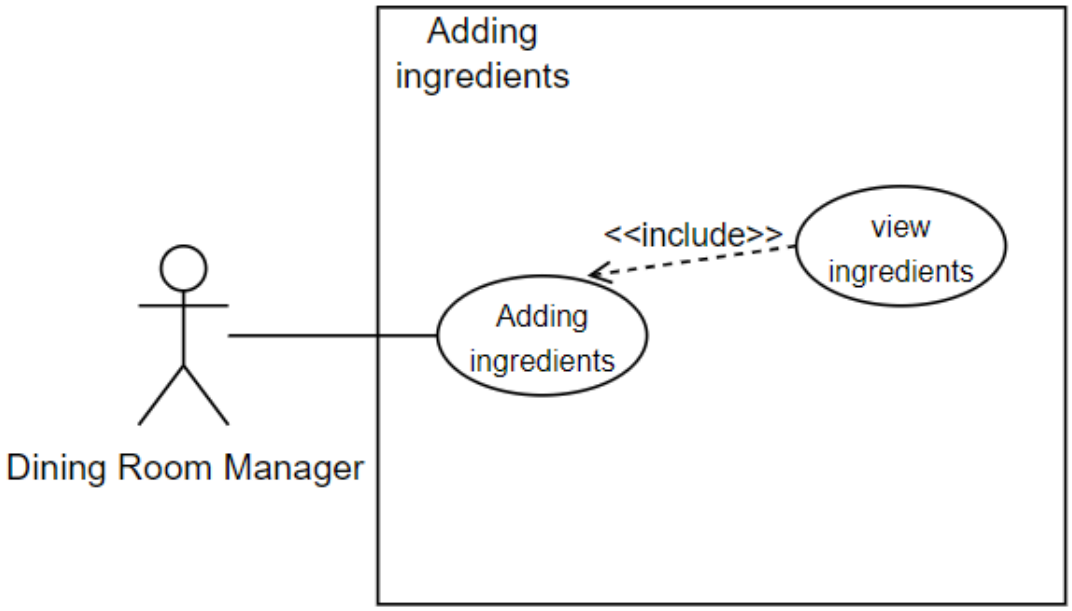
Name	Delete Employees
Authors	Llamas Zamudio Jabes
Date	21/09/24
Description	Allows the administrator to remove an employee from the system
Actors	General Manager
Preconditions	The administrator must be authenticated in the system.
Normal flow	• The administrator selects “Delete Employees”. • Enter the id of the employee to be deleted. • Confirms the deletion. • The system deletes the employee and displays a confirmation message.
Alternative flow	If the employee is not found, the system displays an error message.
Postconditions	The employee is removed from the system



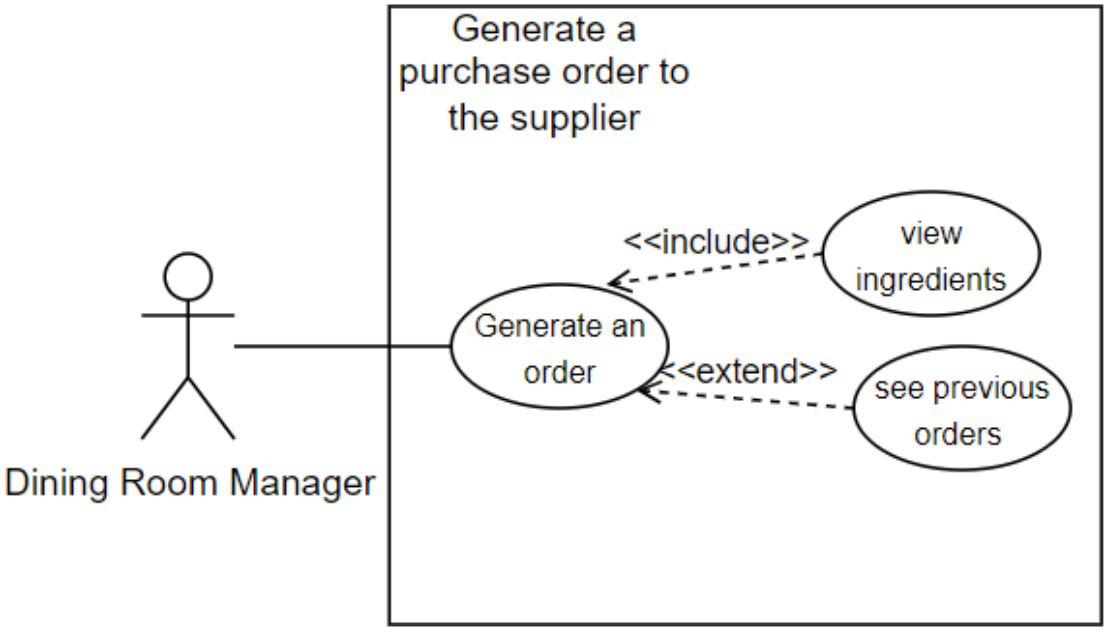
Name	Login
Authors	Llamas Zamudio Jabes
Date	21/09/24
Descriptión	Allows employees to login to their account
Actors	Dining room manager
Preconditions	The Dining room manager must be authenticated in the system.
Normal flow	• The employee must enter his or her data • The employee must select the login option
Alternative flow	If the data entered is not correct, the system displays an error message.
Postconditions	The Dining room manager will log in



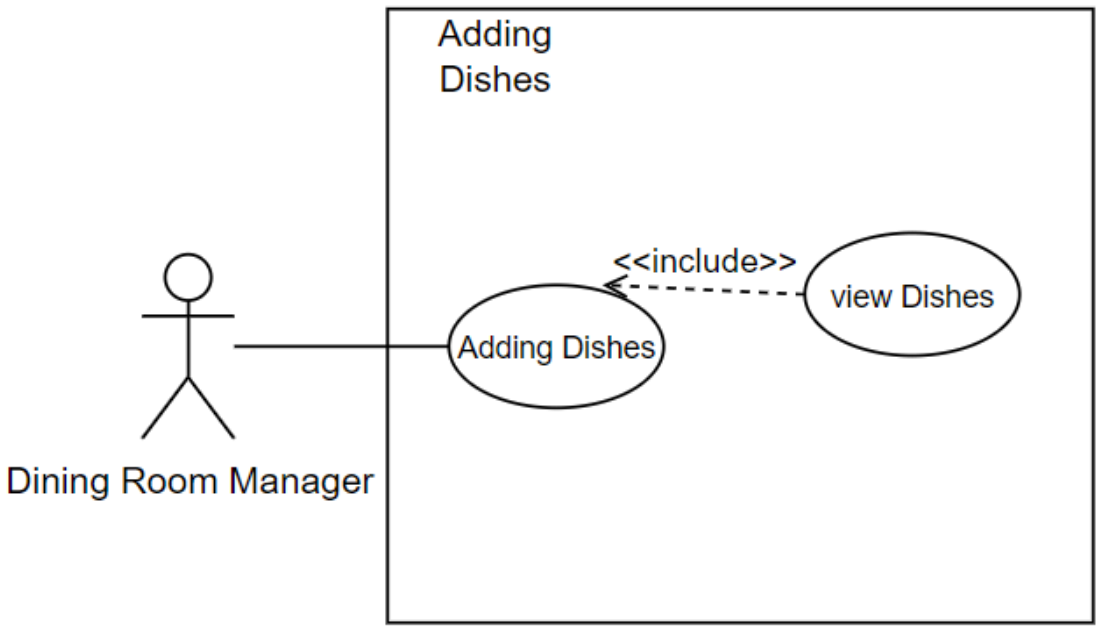
Name	Add ingredients
Authors	Llamas Zamudio Jabes
Date	21/09/24
Descriptión	Allows the Dining room administrator to add new ingredients to the system.
Actors	Dining room manager
Preconditions	The administrator must be authenticated in the system.
Normal flow	• The administrator selects “Add Ingredients”. • He/she enters the ingredient data. • The system saves the ingredient and displays a confirmation message.
Alternative flow	If the data is invalid, the system displays an error message.
Postconditions	The new ingredient is registered in the system.



Name	Generate a purchase order to the supplier
Authors	Llamas Zamudio Jabes
Date	21/09/24
Descriptión	Allows the Dining room manager to generate an order to supply ingredients.
Actors	Dining room manager
Preconditions	The administrator must be authenticated in the system.
Normal flow	• The administrator selects “Generate Order”. • He selects the ingredients and the quantity to order. • The system generates the order and displays a confirmation message.
Alternative flow	If there are not enough ingredients to place the order, the system displays a warning message.
Postconditions	The order is sent to the supplier

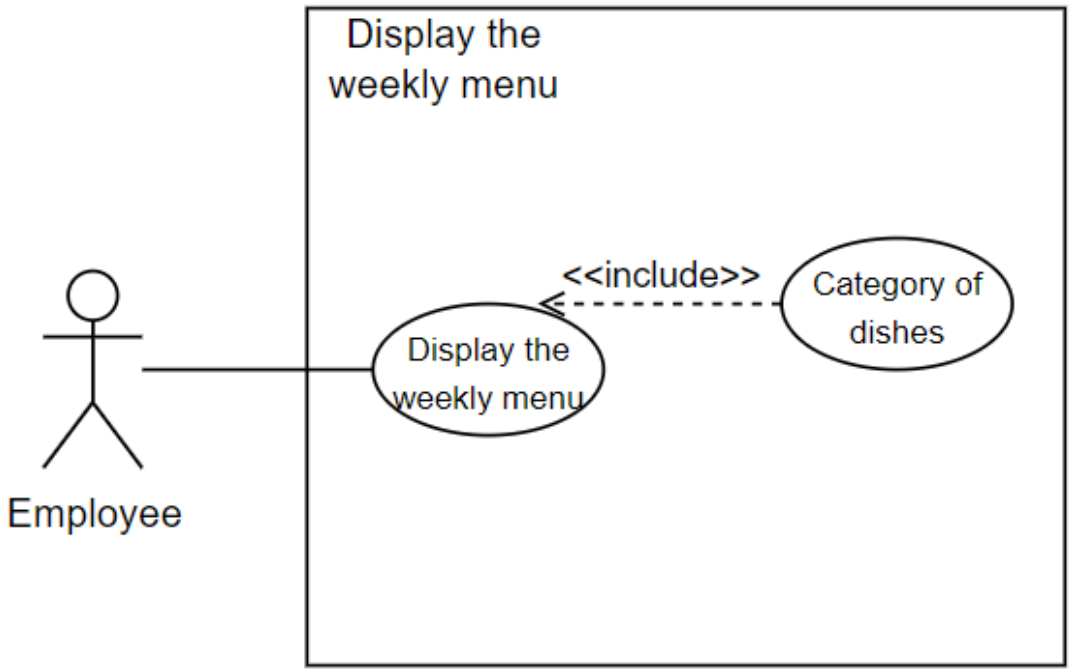


Name	Adding Dishes
Authors	Llamas Zamudio Jabes
Date	21/09/24
Descriptión	Allows the administrator to add a dish
Actors	Dining room manager
Preconditions	The administrator must be authenticated in the system.
Normal flow	• The administrator selects “Generate Order”. • He selects the ingredients and the quantity to order. • The system generates the order and displays a confirmation message.
Alternative flow	If there are not enough ingredients to place the order, the system displays a warning message.
Postconditions	The order is sent to the supplier

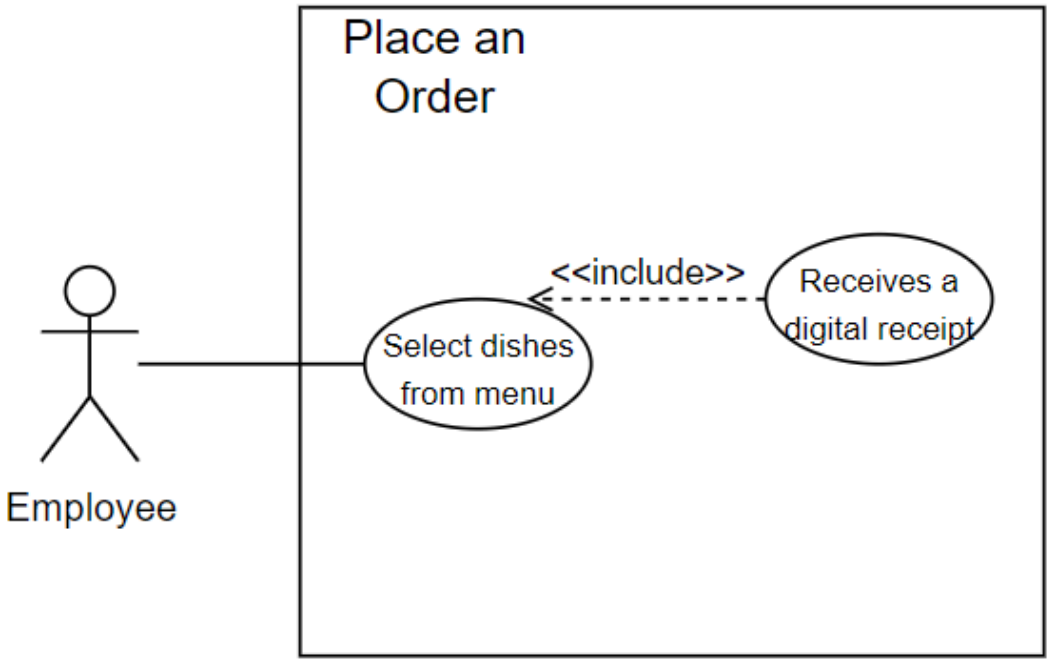


Empleado

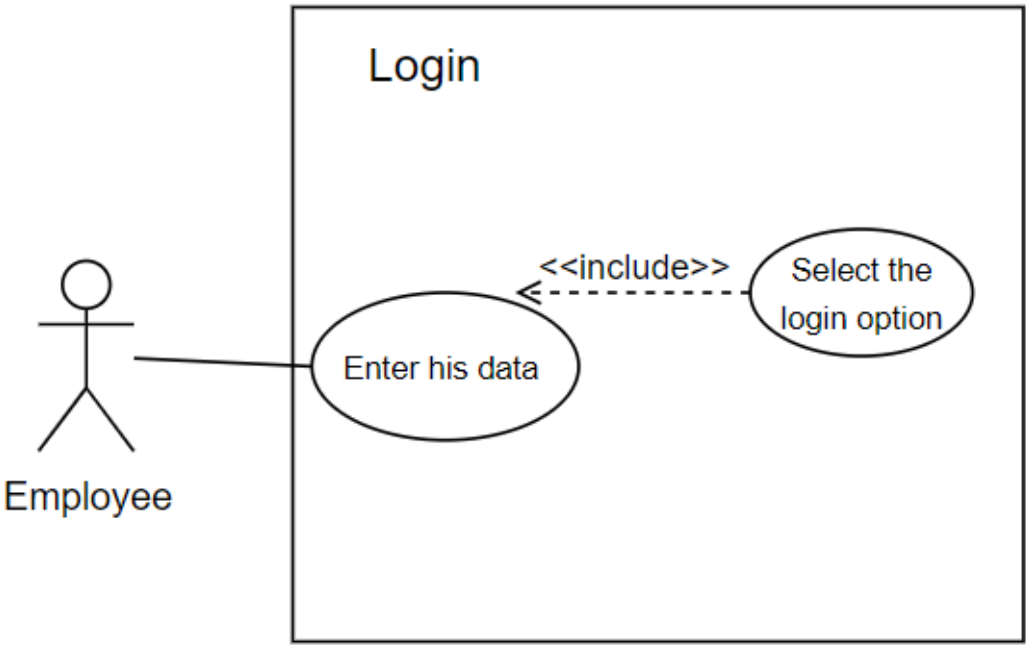
Name	Display the weekly menu
Authors	Llamas Zamudio Jabes
Date	21/09/24
Description	Allows employees to see the weekly menu available
Actors	Employees
Preconditions	The employee must be authenticated in the system.
Normal flow	• The employee selects “Display Weekly Menu”. • The system displays the menu with the categories of dishes.
Alternative flow	If no menu is available, the system displays an error message.
Postconditions	The employee displays the weekly menu.



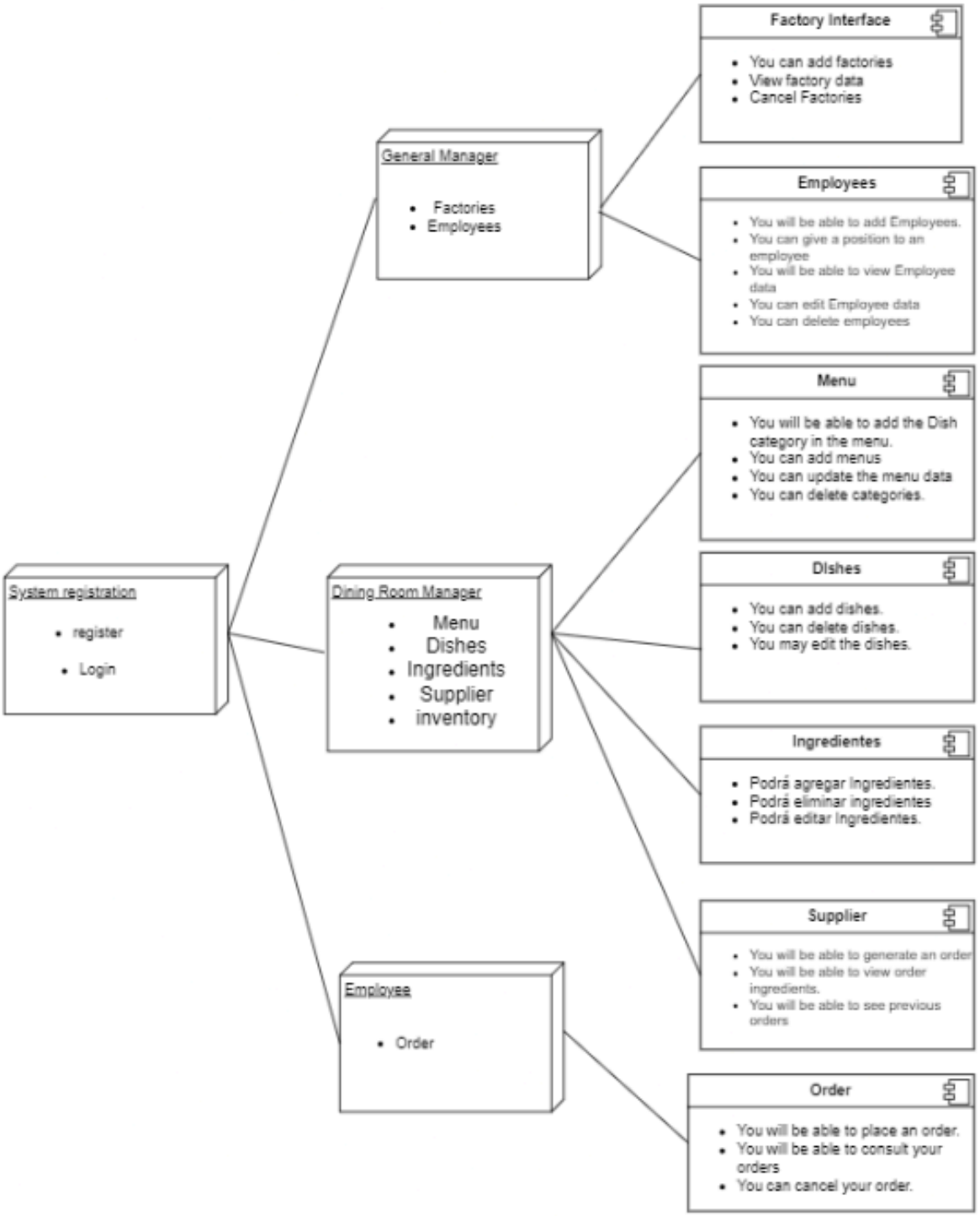
Name	Place an Order
Authors	Llamas Zamudio Jabes
Date	21/09/24
Descriptión	Allows employees to place a food order.
Actors	Employees
Preconditions	The employee must be authenticated in the system and a menu must be available.
Normal flow	• The employee selects a dish from the menu. • The system generates the order. • The employee receives a digital receipt.
Alternative flow	If the pan is not available, the system displays an error message.
Postconditions	The order is registered in the system.



Name	Login
Authors	Llamas Zamudio Jabes
Date	21/09/24
Descriptión	Allows employees to login to their account
Actors	Employees
Preconditions	The employee must be authenticated in the system.
Normal flow	• The employee must enter his or her data • The employee must select the login option
Alternative flow	If the data entered is not correct, the system displays an error message.
Postconditions	The employee will log in

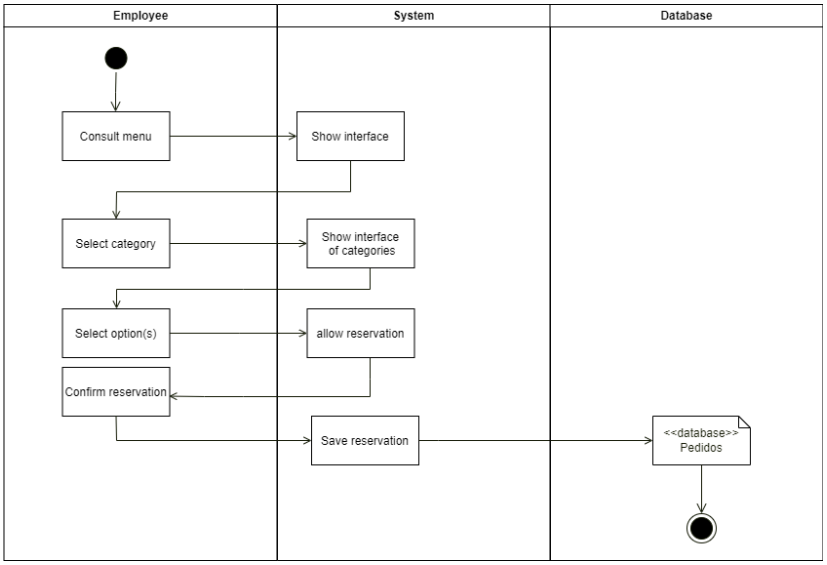

Login
<<include>>
Select the login option
Enter his data
Employee

3.9 DEPLOYMENT DIAGRAM

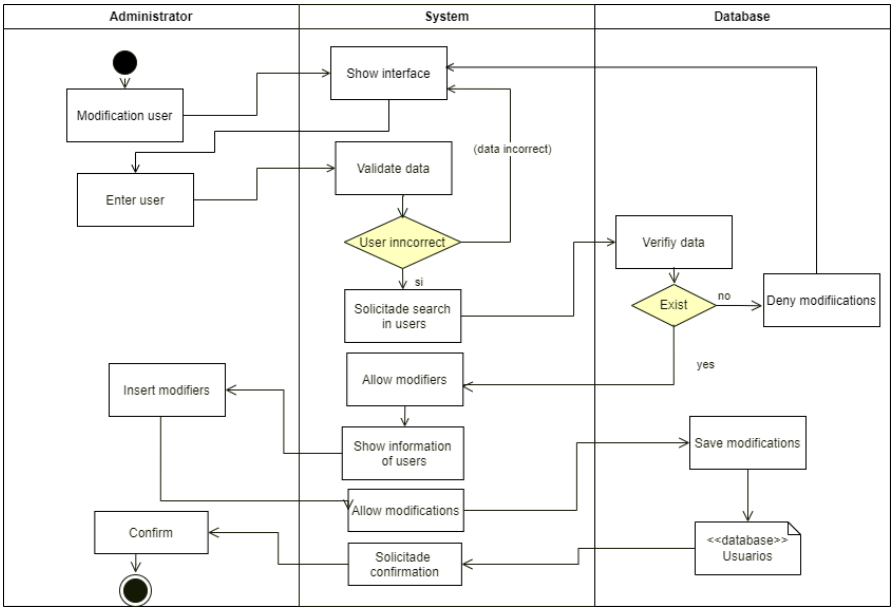


Diagrams of activities

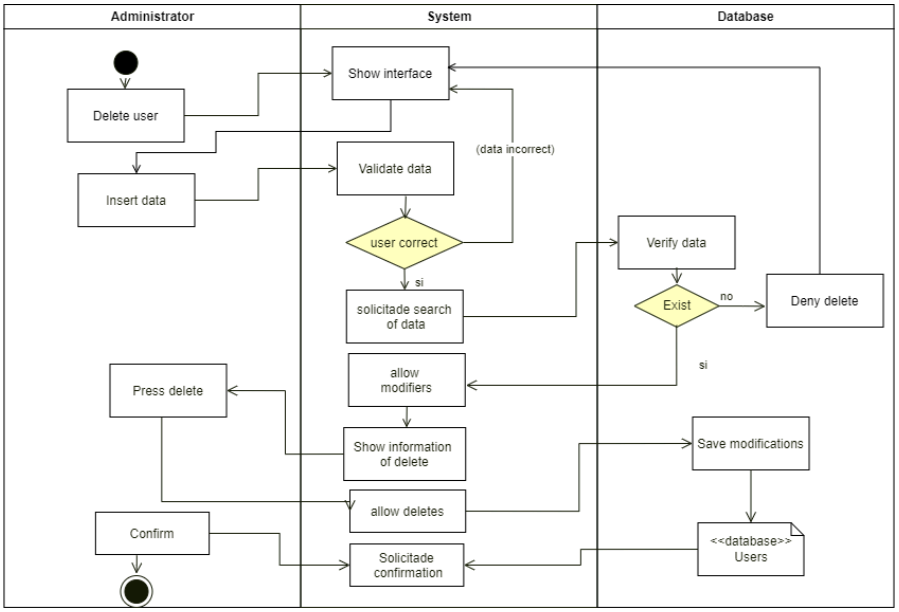
Dish Reservation



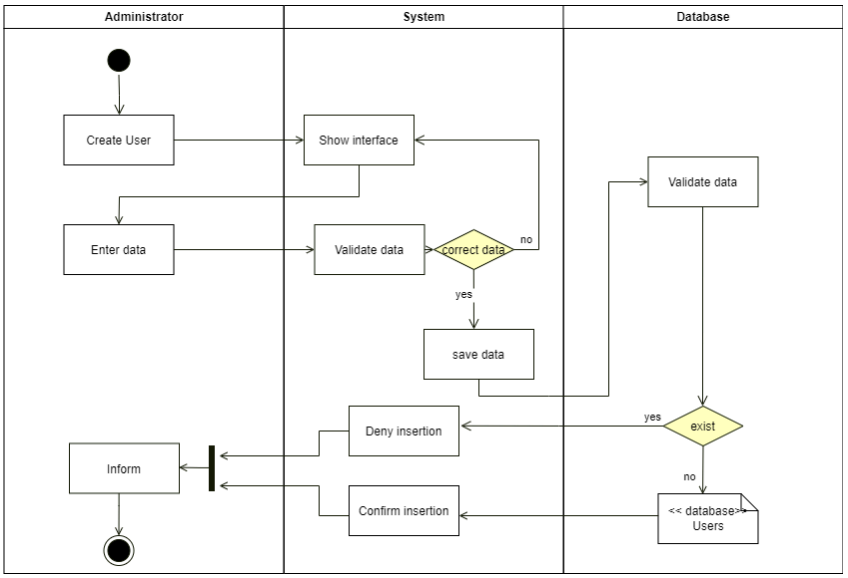
User Modification



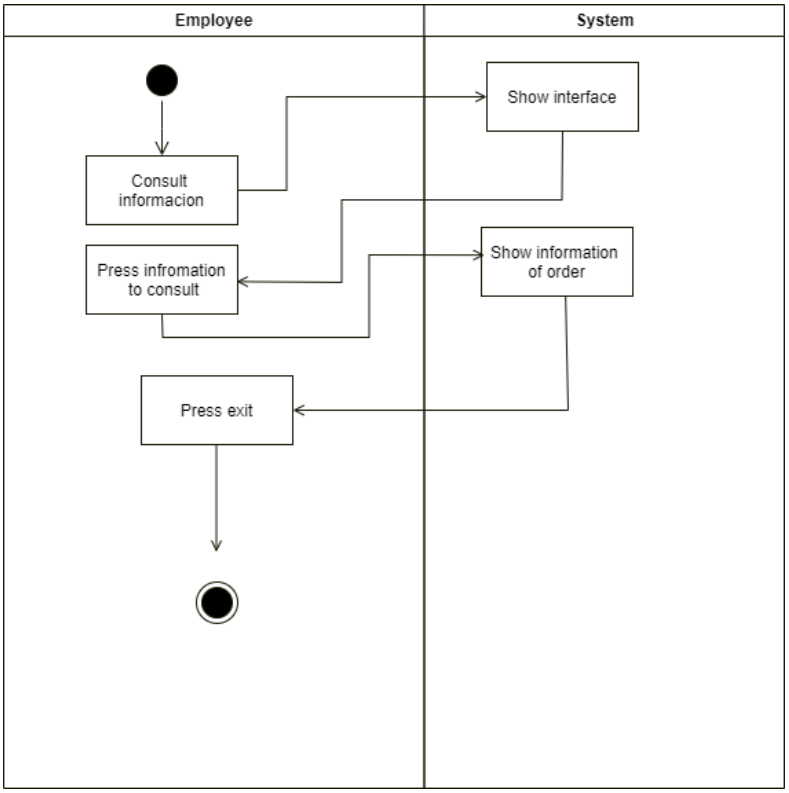
Delete User



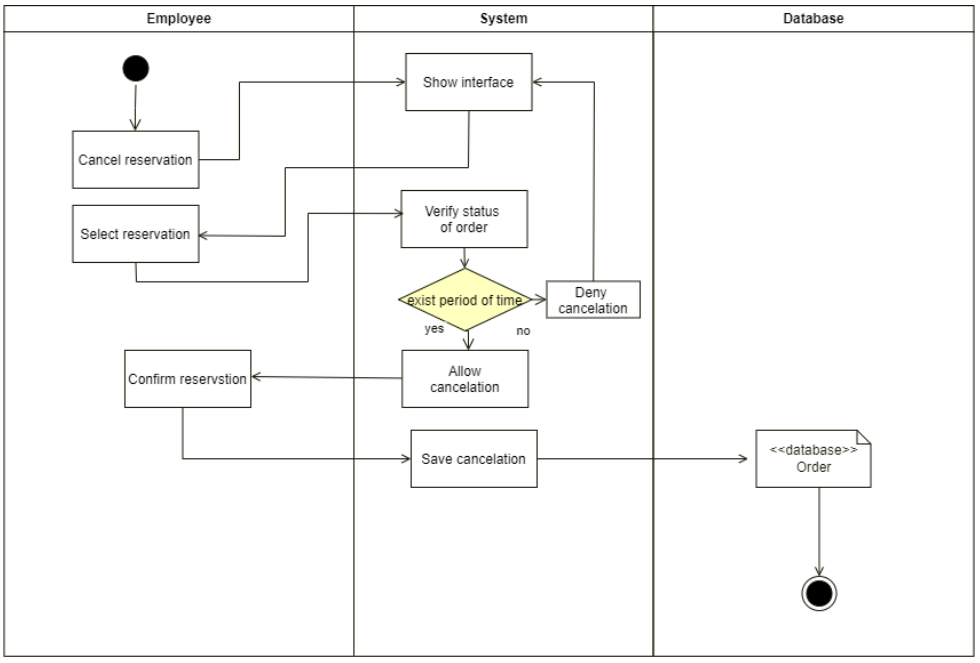
Create User



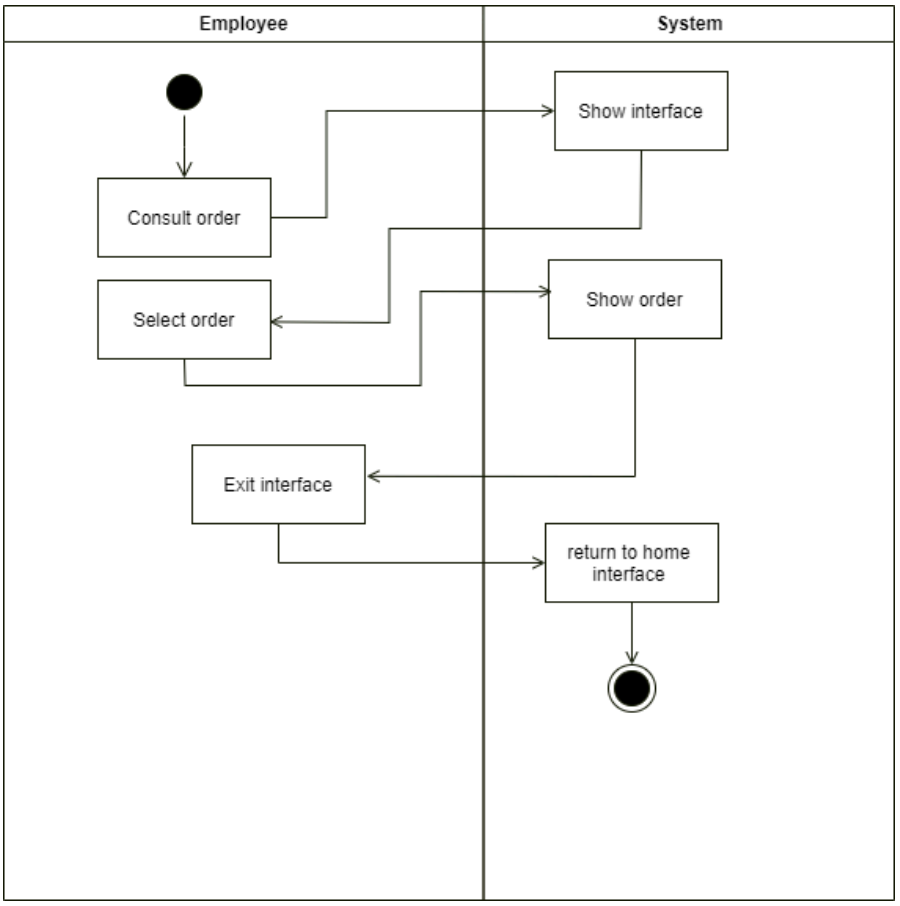
View Menu



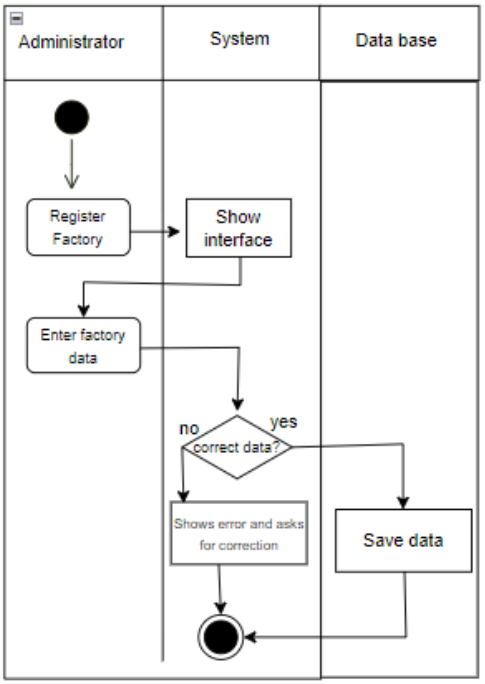
Cancelation of Reservation



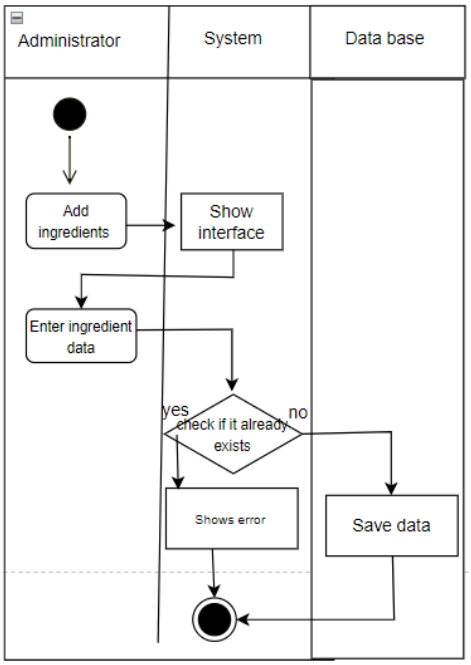
Consult Order



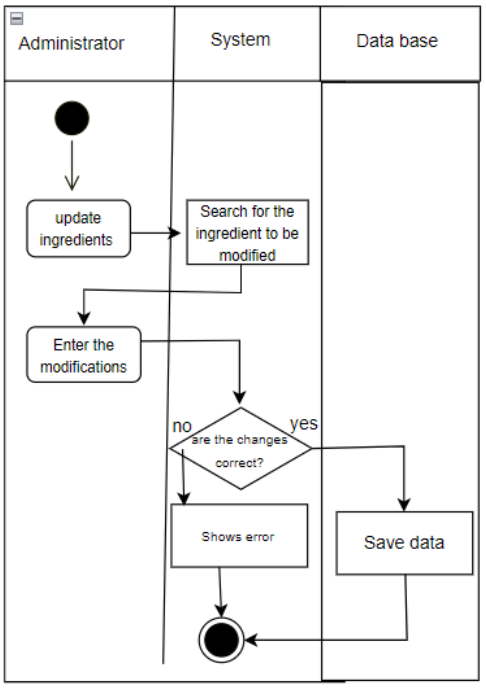
Register Factory



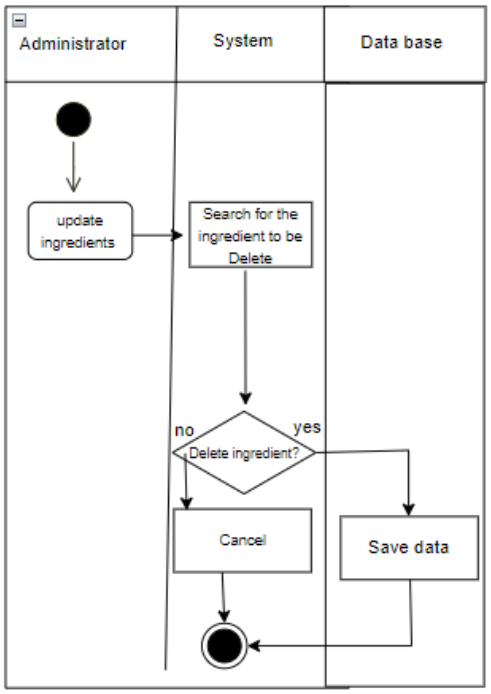
Add ingredients



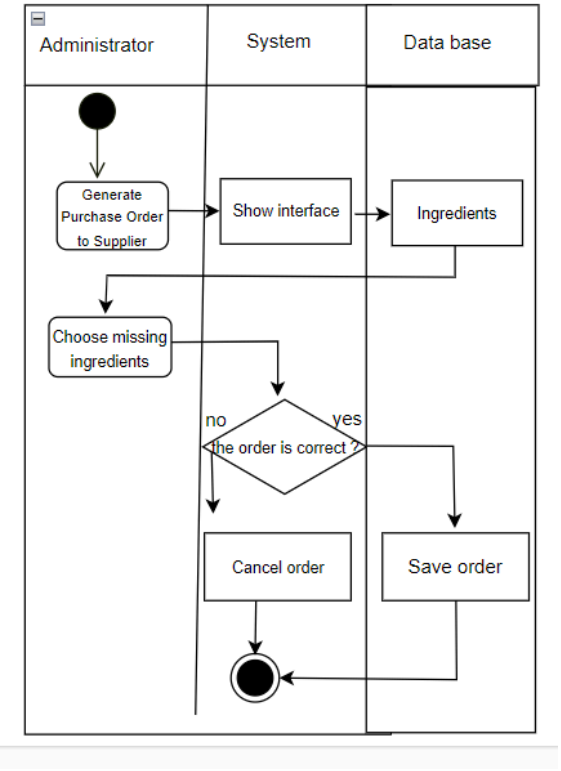
Update ingredient



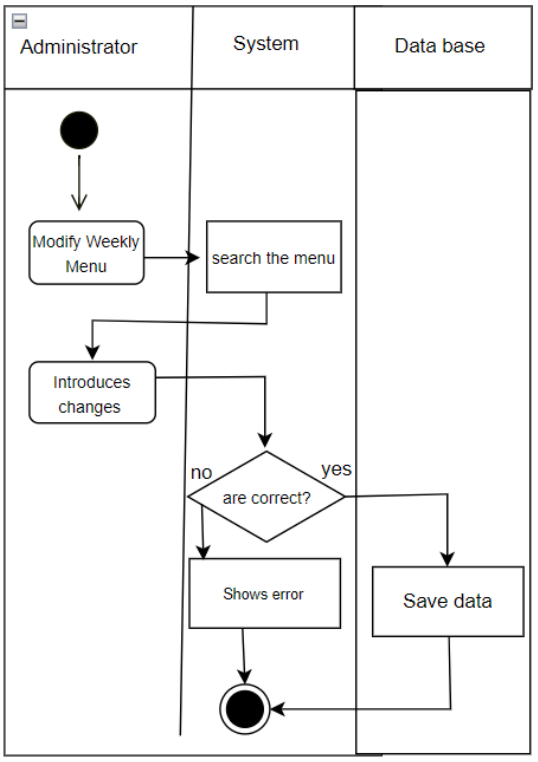
Update ingredient



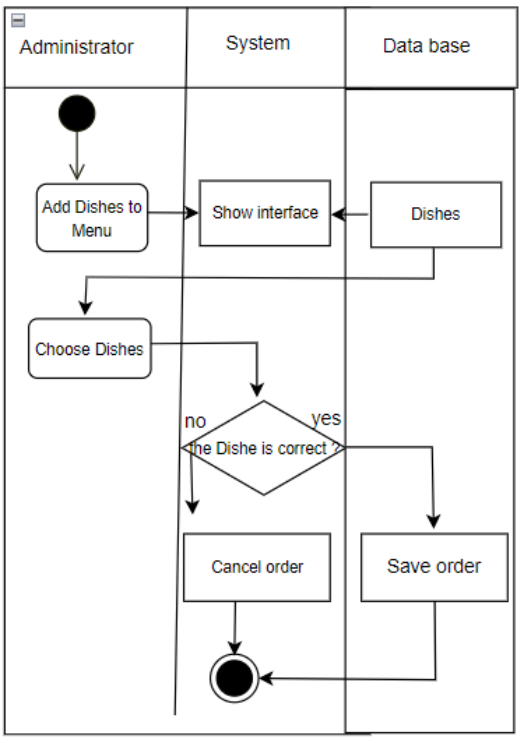
Generate Purchase Order to Supplier



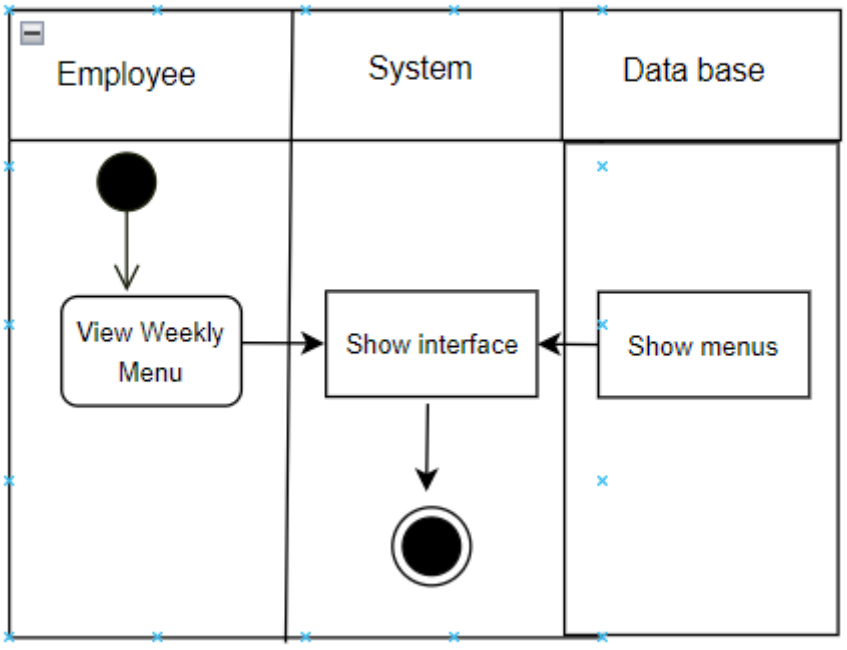
Modify Weekly Menu



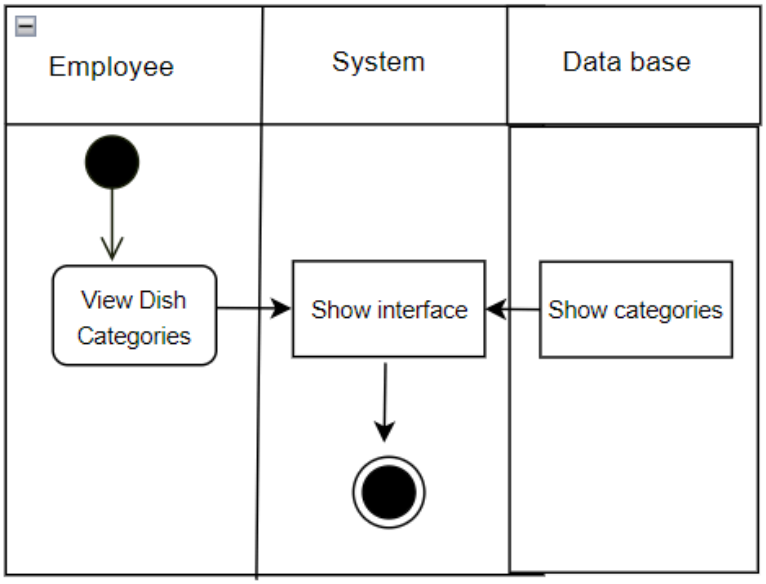
Add Dishes to Menu



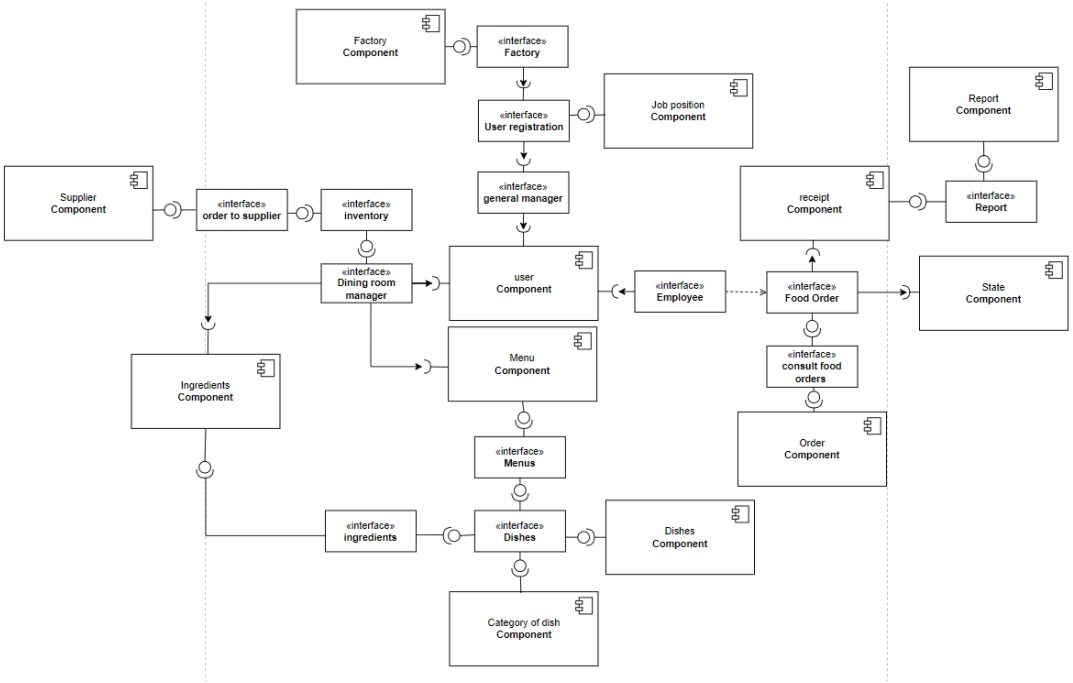
View Weekly Menu



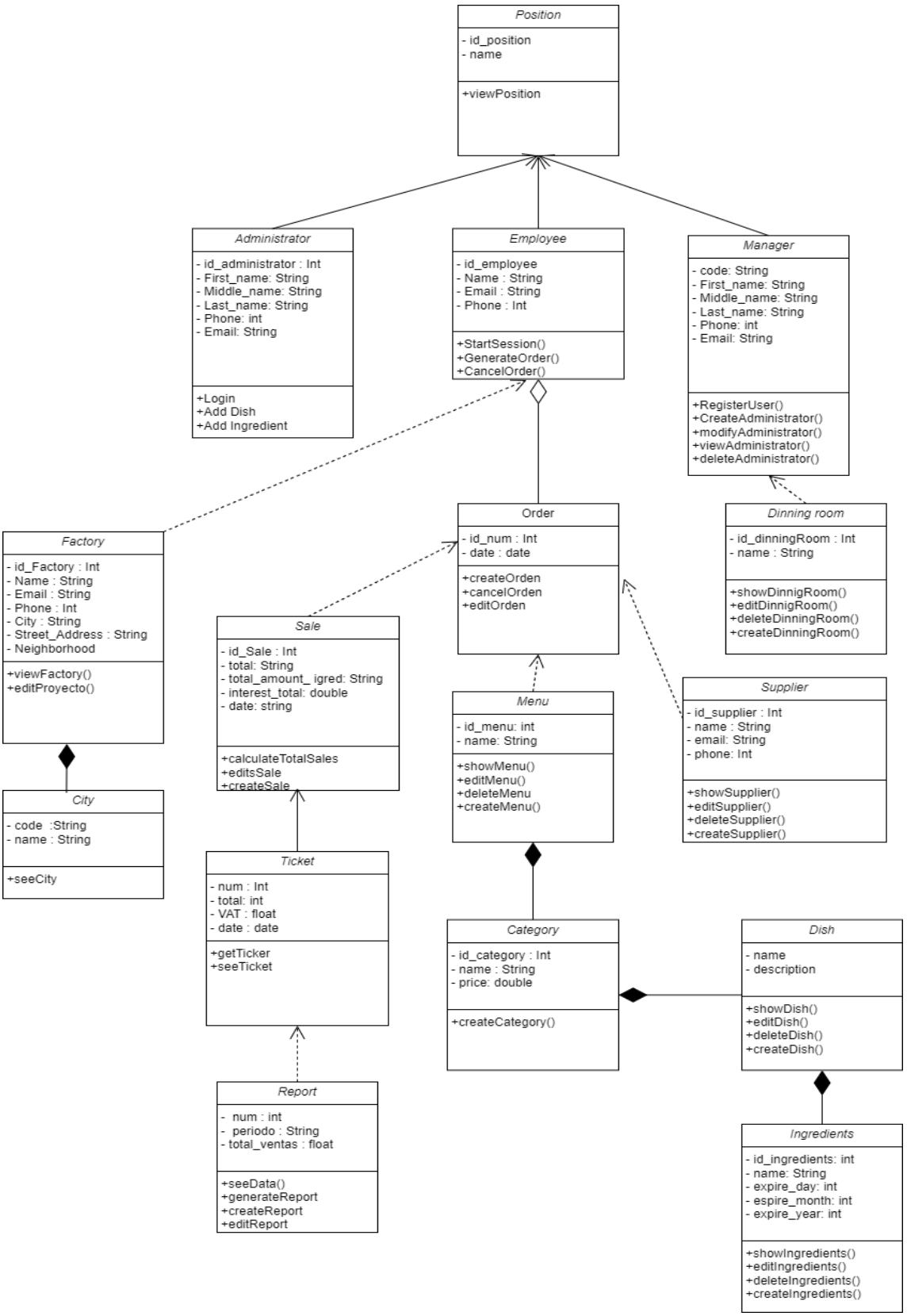
View Dish Categories



4.0 Diagram of components

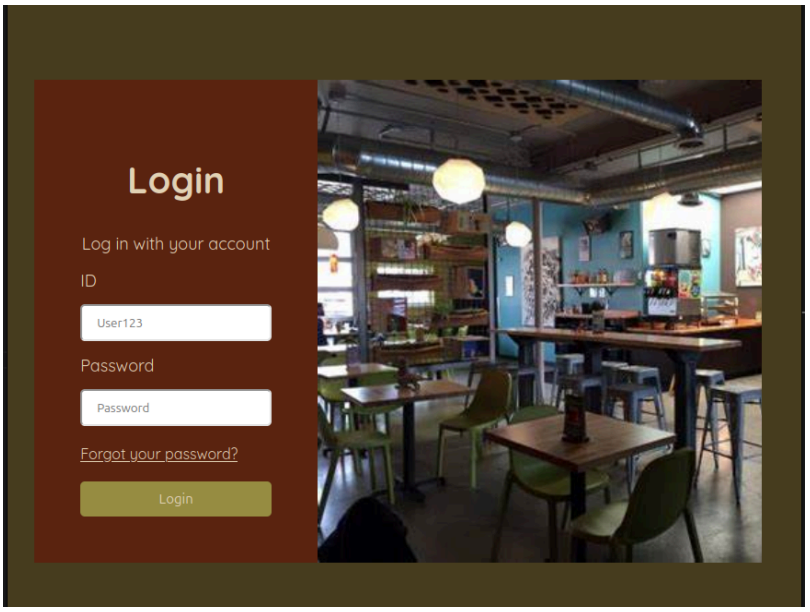


5.0 Class Diagram



6.0 Project progress

Login



In the login we made something that can be seen simple and understandable, as input parameters we ask for the employee's id and a password that will be assigned as well as a button in case you forget the password, finally the button to enter the session.

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Foody</title>
  <link rel="preconnect" href="https://fonts.googleapis.com">
  <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
  <link href="https://fonts.googleapis.com/css2?family=Quicksand:wght@300..700&display=swap" rel="stylesheet">
  <link rel="stylesheet" href="/Project-Foody/Foody/css/styles.css">

</head>

<body>

  <div class="container-form container">

    <form class="form">
      <div class="text-form">
        <h2>Login</h2>
        <p>Log in with your account</p>
      </div>

      <div class="input">

        <label for="Id">ID</label>
        <input placeholder="User123" type="text" id="Id">

      </div>

      <div class="input">

        <label for="Password">Password</label>
        <input placeholder="Password" type="password" id="password">

      </div>

      <div class="forgotten-passw">
        <a href="a">Forgot your password?</a>
      </div>

      <div class="input">
        <input type="submit" value="Login">
      </div>
```

In the code we started with the initial html structure and after putting the text that provides the information to start the account we created a form where the id and password and the button to send the form are placed.

```css
body {
  font-family: var(--text);
  background-color: var(--midnight);
  display: flex;
  align-items: center;
  height: 100vh;
}
.container {
  margin: 0 auto;
  max-width: 1200px;
  width: 95%;
}
.img-form {
  background-image: url('../img/bg.jpg');
  background-position: center center;
  background-size: cover;
  height: 400px;
  flex: 0 0 calc(60%);
  position: relative;
}
.img-form::before {
  content: '';
  position: absolute;
  top: 0;
  bottom: 0;
  left: 0;
  right: 0;
}
```

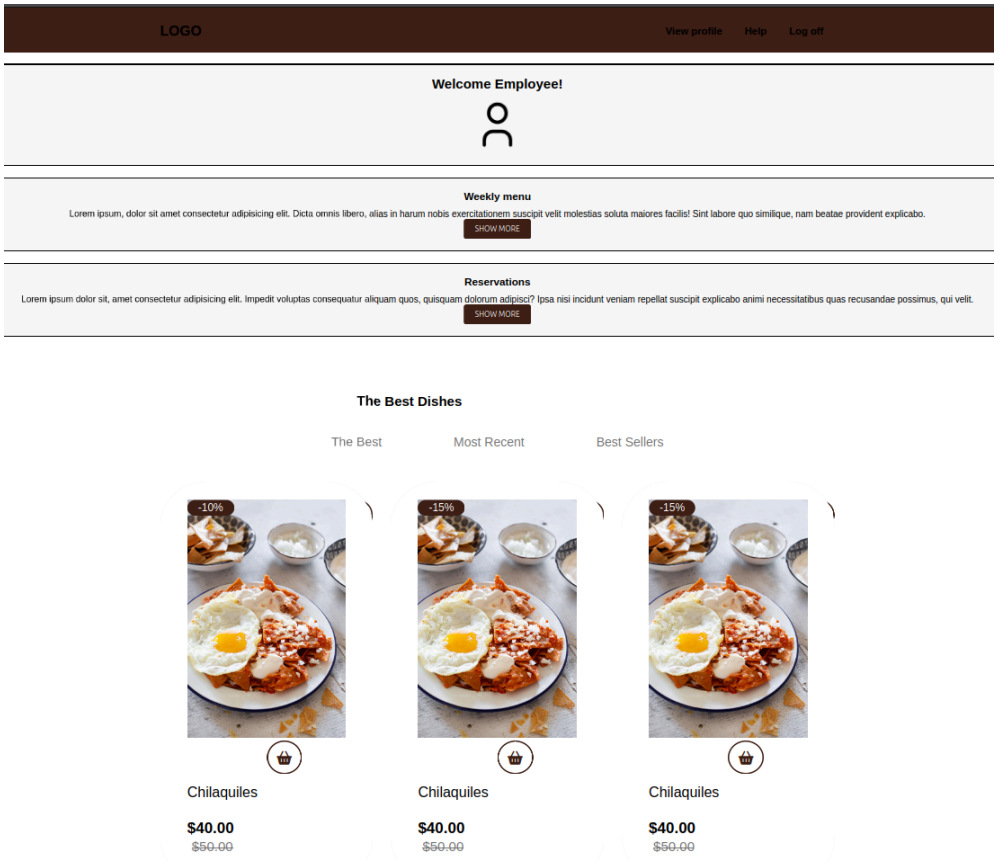
In the CSS of the login we use several classes so that the image of the login is accommodated in a good way and we prefer to insert the image from the CSS and not from the HTML. In the image you can also see everything that takes the body

```php
<?php
require "/php/conectbd.php";
$db = connectDB();

session_start();

if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    $userId = $_POST['Id'];
    $password = $_POST['password'];

    // Consulta para verificar si el usuario existe
    $query = "SELECT * FROM usuarios WHERE id = ?"; // Cambia 'usuarios' al nombre
    $stmt = $db->prepare($query);
    $stmt->bind_param("s", $userId);
    $stmt->execute();
    $result = $stmt->get_result();

    if ($result->num_rows > 0) {
        $user = $result->fetch_assoc();
        if (password_verify($password, $user['password'])) {
            $_SESSION['user_id'] = $user['id'];
            header("Location: /Foody/InterfacesEmpleadohtml/inicioEmpleado.html");
            exit;
        } else {
            $error = "Contraseña incorrecta.";
        }
    } else {
        $error = "ID de usuario no encontrado.";
    }
}
?>
```

Description:This PHP code handles a login process by checking if a user exists in the database and verifying their password. It starts by connecting to the database and initializing a session. When a login form is submitted, it retrieves the user ID and password from the form, then prepares and executes a query to check if a matching user ID exists. If a user is found, it verifies the password, and if correct, it saves the user ID in the session and redirects them to an

employee dashboard page. If the login fails, it sets an error message indicating either an incorrect password or that the user ID was not found.

Home



At the beginning the employee is divided into several sections, first is the header that is the top bar and then displays information about the weekly menu and then there is a button for more information. Then another section is created where it shows the reservations with its respective button that directs to the reservation interface. Finally it shows the dishes that will be the best sellers.

```
<section>
    <div class="rectangle">
        <h2>Welcome Employee!</h2>
        <svg xmlns="http://www.w3.org/2000/svg" class="icon
        icon-tabler icon-tabler-user" width="96" height="96"
        viewBox="0 0 24 24" stroke-width="1.5" stroke="#000000"
        fill="none" stroke-linecap="round" stroke-linejoin="round">
            <path stroke="none" d="M0 0h24v24H0z" fill="none"/>
            <path d="M8 7a4 4 0 1 0 8 0a4 4 0 0 0 -8 0" />
            <path d="M6 21v-2a4 4 0 0 1 4 -4h4a4 4 0 0 1 4 4v2" />
        </svg>
    </div>
</section>

<section>
    <div class="rectangle">
        <h3>Weekly menu</h3>
        <p>Lorem ipsum, dolor sit amet consectetur adipisicing elit.
        Dicta omnis libero, alias in harum nobis exercitationem
        suscipit velit molestias soluta maiores facilis! Sint labore
        quo similique, nam beatae provident explicabo.</p>
        <button><a href="/Foody/InterfacesEmpleadohtml/categoryEmplo.
        html">Show More</a></button>
    </div>
</section>
```

In the html of this interface, the rectangle class was added to adjust all the information inside the rectangles, within these divisions are the title and the information, as well as the button that leads to this information.

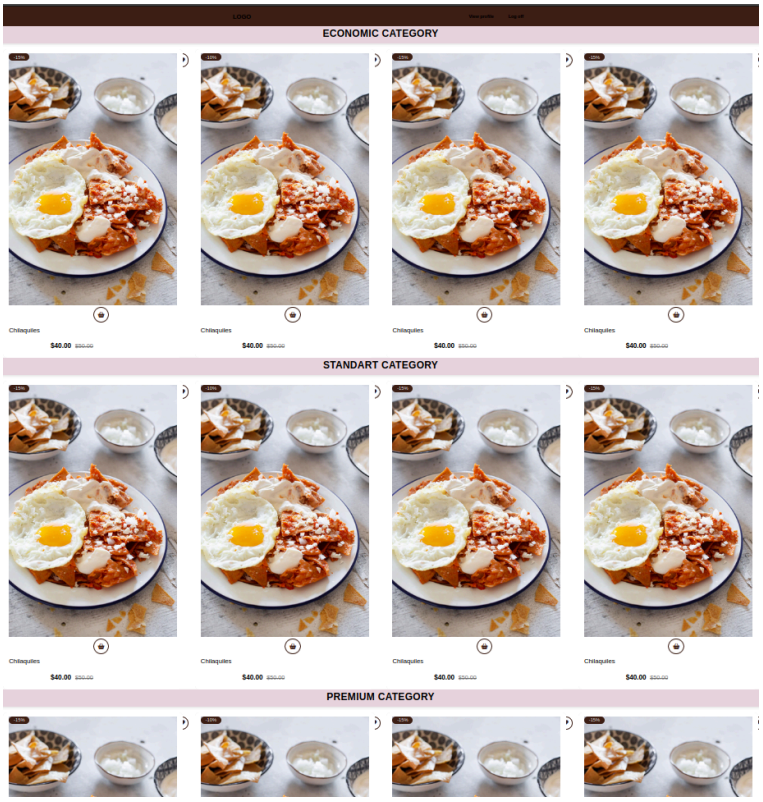
```
.rectangle {
    border: 2px solid #000;
    padding: 20px;
    margin: 20px 0;
    background-color: #f5f5f5;
    border-radius: 10px;
    text-align: center;
}
.rectangle h2, .rectangle h3 {
    margin-bottom: 10px;
    font-weight: bold;
}
.rectangle p {
    font-size: 16px;
}
.rectangle button {
    background-color: #3c1e15;
    color: white;
    padding: 10px 20px;
    border: none;
    border-radius: 5px;
    cursor: pointer;
    text-transform: uppercase;
}
.rectangle button a {
    color: white;
    text-decoration: none;
}
.rectangle button:hover {
    background-color: #d0d0dd;
}
```

In the CSS of the interface of the employee's home we create the class rectangle to set the size and thickness of the border in addition to the information it contains is also the class of the buttons that puts the colors of them.

```php
$products = [
    ['name' => 'Chilaquiles', 'image' => '/Project-Foody/Foody/css/imagenes/chilaquiles rojos.jpg', 'price' => 40.00, 'discount' => 10],
    ['name' => 'Tacos', 'image' => '/Project-Foody/Foody/css/imagenes/tacos.jpg', 'price' => 30.00, 'discount' => 15],
    ['name' => 'Enchiladas', 'image' => '/Project-Foody/Foody/css/imagenes/enchiladas.jpg', 'price' => 50.00, 'discount' => 20],
];

foreach ($products as $product) {
    echo '<div class="card-product">
            <div class="container-img">
                <img src="' . htmlspecialchars($product['image']) . '" alt="' . htmlspecialchars($product['name']) . '">
                <span class="discount">-' . htmlspecialchars($product['discount']) . '%</span>
                <div class="button-group">
                    <span>
```

This PHP code defines an array of products, each with a name, image, price, and discount. It then uses a foreach loop to generate HTML for each product, displaying an image, product name, and discount. The htmlspecialchars function is used to safely output data, protecting against XSS attacks. This setup is ideal for showing products in a card layout with images and discount info.

Category of dishes



This interface shows the categories of dishes that we have and which ones are available, as well as the prices of the dishes and their discount.

```
<section class="category-container">
    <h2>Economic Category</h2>
    <div class="container-products">
        <div class="card-product">
            <div class="container-img">
                <img src="/Project-Foody/Foody/css/imagenes/chilaquiles rojos.jpg" alt="chilaquiles">
                    <span class="discount">-10%</span>
                    <div class="button-group">
                        <span>
                            <i class="fa-solid fa-heart"></i>
                        </span>
                    </div>
            </div>
            <div class="content-card-product">
                <div class="">  <!-- Aqui se pueden poner las estrellas de calificacion en caso de que nos de tiempo -->

                </div>
                <h3>Chilaquiles </h3>
                <span class="add-cart">
                    <i class="fa-solid fa-basket-shopping"></i>
                </span>
                <p class="price">$40.00 <span>$50.00</span></p>
            </div>
        </div>
        <!-- Producto 2 -->
```

In this code you can see that we used the category container class that is responsible for putting the appropriate colors and limits, then in the div we use the card-product class, each product that we want to display on the website will be within this class and within this is put all the information of the products.

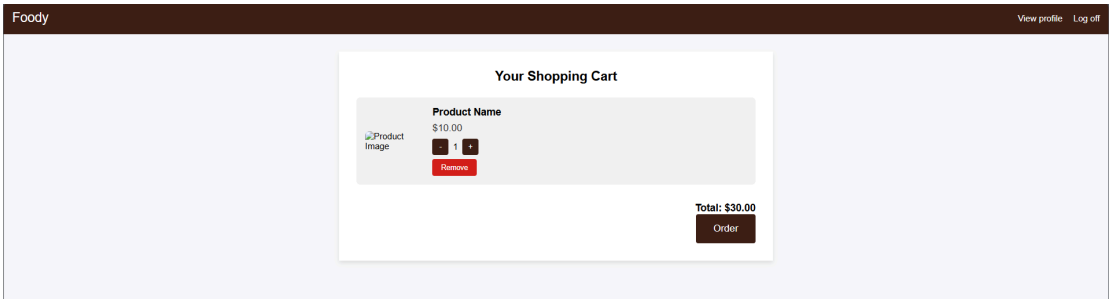
```css
.container-products{
    display: grid;
    grid-template-columns: repeat(auto-fit, minmax(20rem, 1fr));
    gap: 3rem;
}

.card-product{
    background-color: #ffff;
    padding: 1rem 1.5rem;
    border-radius: 1rem;
    cursor: pointer;
    box-shadow: 0 0 .2px rgb(0, 0, 0, 0.1);
}

.container-img{
  position: relative;
}

.container-img img{
    width: 100%;
}

.container-img-reservation img{
    width: 40%; /* Ajusta este valor para cambiar el tamaño de la imagen */
    height: auto; /* Mantiene la proporción de la imagen */
    display: block;
    margin: 0 auto;
}


.container-img .discount{
    position: absolute;
    left: 0;
    background-color: #3c1e15;
    color: #fff;
    padding: 2px 1.2rem;
    border-radius: 1rem;
    font-size: 1.2rem;
}

.card-product:hover .discount{
    background-color: #fff;
}
```

In the CSS we created several classes for the cards of the products and we put that it is centered. We adjust the px and rem in addition to the colors that you have for the page, these classes are used in almost all the interfaces.

```php
$db = connectDB();

// Datos de productos
$products = [
    'Economic' => [
        ['name' => 'Chilaquiles', 'image' => 'chilaquiles_rojos.jpg', 'price' => 40.00, 'original_price' => 50.00, 'discount' => '-10%'],
        ['name' => 'Chilaquiles', 'image' => 'chilaquiles_rojos.jpg', 'price' => 40.00, 'original_price' => 50.00, 'discount' => '-15%'],
    ],
    'Standart' => [
        ['name' => 'Chilaquiles', 'image' => 'chilaquiles_rojos.jpg', 'price' => 40.00, 'original_price' => 50.00, 'discount' => '-10%'],
        ['name' => 'Chilaquiles', 'image' => 'chilaquiles_rojos.jpg', 'price' => 40.00, 'original_price' => 50.00, 'discount' => '-15%'],

    ],
    'Premium' => [
        ['name' => 'Chilaquiles', 'image' => 'chilaquiles_rojos.jpg', 'price' => 40.00, 'original_price' => 50.00, 'discount' => '-10%'],
        ['name' => 'Chilaquiles', 'image' => 'chilaquiles_rojos.jpg', 'price' => 40.00, 'original_price' => 50.00, 'discount' => '-15%'],

    ],
];
?>

<section class="category-container">
    <?php foreach ($products as $category => $items): ?>
        <h2><?php echo $category; ?> Category</h2>
        <div class="container-products">
            <?php foreach ($items as $item): ?>
                <div class="card-product">
                    <div class="container-img">
                        <img src="/Project-Foody/Foody/css/imagenes/<?php echo $item['image']; ?>" alt="<?php echo $item['name']; ?>">
                        <span class="discount"><?php echo $item['discount']; ?></span>
                        <div class="button-group">
                            <span>
```

This PHP code connects to a database, then defines products organized into "Economic," "Standard," and "Premium" categories, each with product details like name, image, price, and discount. It then generates HTML to display each category and its products in a card format, showing the image, name, and discount for each item.

Order



This interface shows the orders that have been registered with their respective names, the price of the product, name and image, two buttons were added for the user to select the number of dishes he wants and a button to remove the dish, the total of the order and the button to generate the order were added.

```
</header>
<main>
    <section class="cart-section">
        <h2>Your Shopping Cart</h2>
        <div class="cart-container">
            <div class="cart-item">
                <img src="/Foody/css/imagenes/producto.jpg" alt="Product Image">
                <div class="item-details">
                    <h3>Product Name</h3>
                    <p>$10.00</p>
                    <div class="quantity">
                        <button class="quantity-btn">-</button>
                        <span class="quantity-value">1</span>
                        <button class="quantity-btn">+</button>
                    </div>
                    <button class="remove-btn">Remove</button>
                </div>
            </div>
        </div>
        <div class="cart-summary">
            <h3>Total: $30.00</h3>
            <button class="checkout-btn">Order</button>
        </div>
    </section>
</main>
</body>
```

In the HTML file we call the class “Cart-section” inside a section, then we put the product information that will be replaced by the variables and finally we put the total of the order.

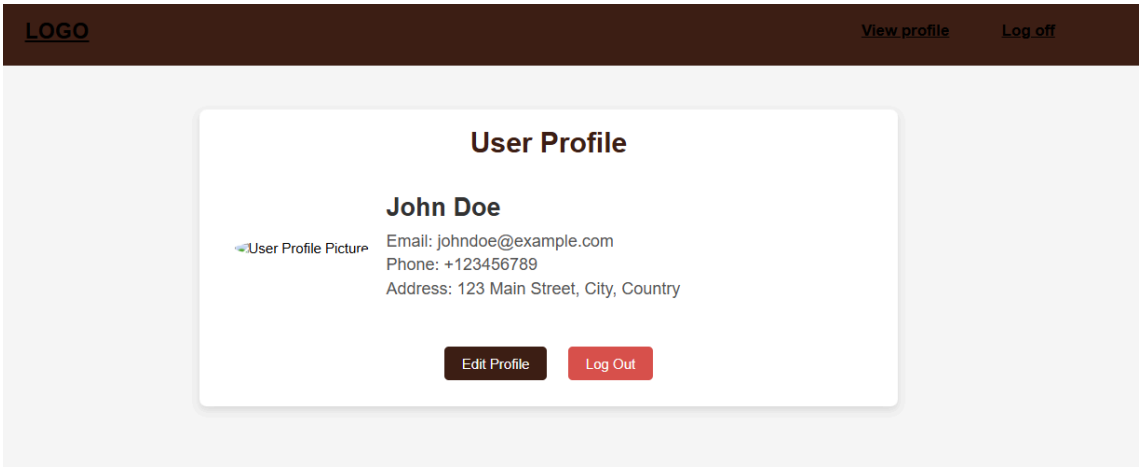
```css
.quantity-btn {
    width: 30px;
    height: 30px;
    background-color: #3c1e15;
    color: white;
    border: none;
    border-radius: 4px;
    cursor: pointer;
}
.remove-btn {
    background-color: #D4211C;
    color: white;
    padding: 0.5rem 1rem;
    border: none;
    border-radius: 4px;
    cursor: pointer;
}
.cart-summary {
    margin-top: 2rem;
    text-align: right;
}
.checkout-btn {
    background-color: #3c1e15;
    color: white;
    padding: 1rem 2rem;
    border: none;
    border-radius: 4px;
    cursor: pointer;
    font-size: 18px;
}
```

In the CSS of the order interface are the classes we used for the “Remove” button that we set the color of the letters and the color of the button and the letter sizes.

```php
<?php
include "/php/header.php";
require "/php/conectbd.php";
$db = connectDB();

$userId = 1;
$cartQuery = "SELECT * FROM cart WHERE user_id = $userId";
$result = $db->query($cartQuery);

$cartItems = [];
$totalPrice = 0;

if ($result) {
    while ($row = $result->fetch_assoc()) {
        $cartItems[] = $row;
        $totalPrice += $row['price'] * $row['quantity'];
    }
}
?>
```

Here a query is created to verify the orders of each employee and an if was created so that while the user is increasing or decreasing the amount of products, the price is modified in the same way.

View profile



In this interface we will use it so that the user can verify his information and if he detects an error he can modify it and also from this interface he can log out of the session.

```
</section>
<section class="profile-container">
    <div class="profile-header">
        <h2>User Profile</h2>
    </div>
    <div class="profile-content">
        <div class="profile-pic">
            <img src="/Foody/images/profile-pic.jpg" alt="User Profile Picture">
        </div>
        <div class="profile-details">
            <h3>John Doe</h3>
            <p>Email: johndoe@example.com</p>
            <p>Phone: +123456789</p>
            <p>Address: 123 Main Street, City, Country</p>
        </div>
    </div>
    <div class="profile-actions">
        <button class="btn-edit">Edit Profile</button>
       <a href="/Foody/index.html"><button class="btn-logout">Log Out</button></a>
    </div>
</section>
```

In this html the user profile is created in which different methods are used to organize the information that is presented such as "p" to show the user information, with a button to edit profile and another to close a session that is linked to another interface by means of a link “a”.

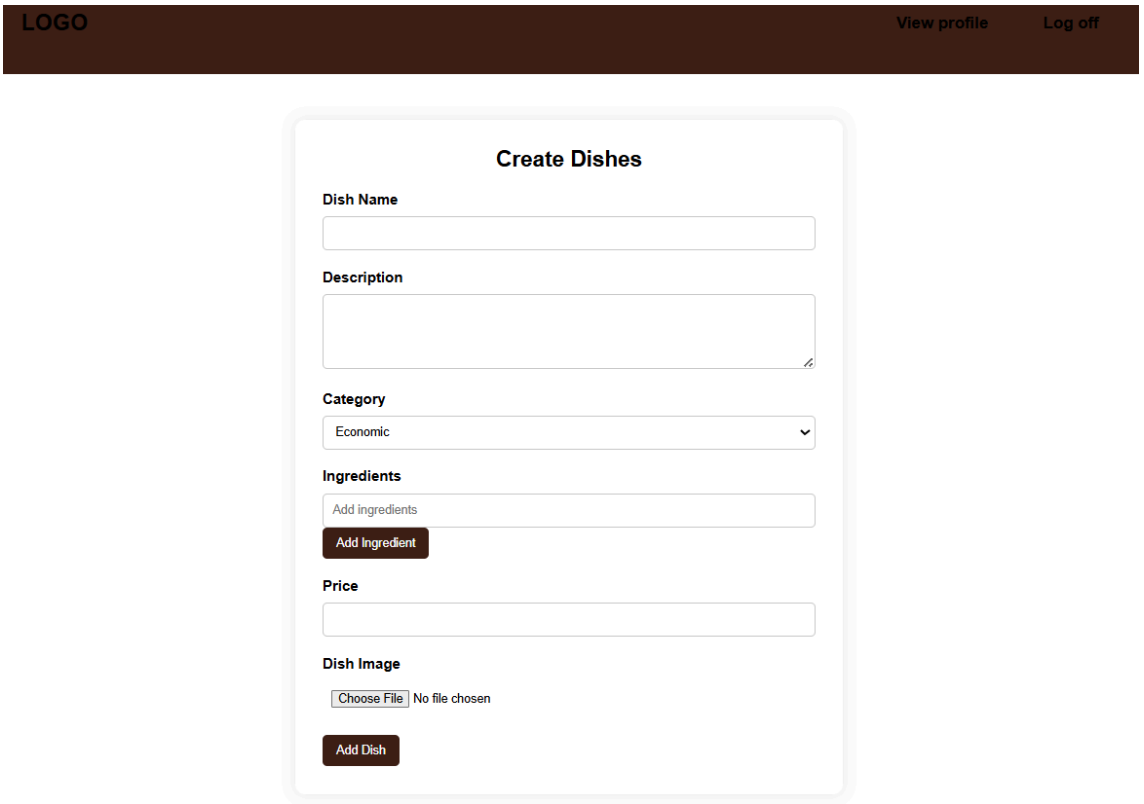
```php
<?php
include "/php/header.php";
require "/php/conectbd.php";
$db = connectDB();

session_start();
$user_id = $_SESSION['user_id'];

// Consulta para obtener los datos del usuario
$query = "SELECT nombre, email, telefono, direccion FROM usuarios WHERE id = ?";
$stmt = $db->prepare($query);
$stmt->bind_param("i", $user_id);
$stmt->execute();
$result = $stmt->get_result();

// Verifica si el usuario existe
if ($result->num_rows > 0) {
    $user = $result->fetch_assoc();
} else {
    echo "User not found";
    exit;
}
?>
```

This PHP script connects to a database to retrieve user information based on the user's session ID. After including necessary files and establishing a database connection, it starts a session and retrieves the user's ID from the session data.

```css
.profile-container {
    width: 100%;
    max-width: 800px;
    margin: 50px auto;
    background-color: #fff;
    border-radius: 8px;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
    padding: 20px;
}
.profile-header {
    text-align: center;
    padding-bottom: 20px;
}
.profile-header h2 {
    font-size: 2rem;
    color: #3c1e15;
}
.profile-content {
    display: flex;
    align-items: center;
    justify-content: space-around;
    padding: 20px;
}
.profile-pic img {
    width: 150px;
    height: 150px;
    border-radius: 50%;
    object-fit: cover;
```

In this CSS, the “profile-container” class ensures that it fits on different screens, and the automatic margin centers the container on the page. The “profile-content” class organizes the image and profile details inline, vertically and horizontally.

Add dishes



This interface will be used to create new dishes in which the manager is able to directly type in the name, dish description and price of the dish, with a category options section, add a visual representation of the dish using files, and two buttons in which one can add ingredients while the other adds the dish.

```
</section>
    <section class="create-dish-container">
        <h2>Create Dishes</h2>
        <form action="#" method="POST" enctype="multipart/form-data">
            <div class="form-group">
                <label for="name">Dish Name</label>
                <input type="text" id="name" name="name" required>
            </div>

            <div class="form-group">
                <label for="description">Description</label>
                <textarea id="description" name="description" rows="4" required></textarea>
            </div>

            <div class="form-group">
                <label for="category">Category</label>
                <select id="category" name="category" required>
                    <option value="economic">Economic</option>
                    <option value="Standart">Standart</option>
                    <option value="premium">Premium</option>
                </select>
            </div>
```

In this HTML, forms are used for data input, as well as the POST method to ensure that the information is not reflected in the URL, and the required attribute to prevent empty fields.

```
.create-dish-container {
    background-color: #fff;
    padding: 30px;
    max-width: 600px;
    margin: 50px auto;
    border-radius: 10px;
    box-shadow: 0 0 15px rgba(0, 0, 0, 0.1);
}

.create-dish-container h2 {
    text-align: center;
    margin-bottom: 20px;
    font-size: 24px;
}

.form-group {
    margin-bottom: 20px;
}

label {
    font-weight: bold;
    margin-bottom: 5px;
    display: block;
}

```

This CSS code styles a form container, headings, and labels to create a clean, centered, and visually appealing design. It uses a white background, rounded corners, and a subtle shadow effect to add depth to the container. The layout organizes elements with appropriate margins and spacing, making the content look structured and easy to read. Overall, the code enhances the aesthetics and usability of the interface for a better user experience.

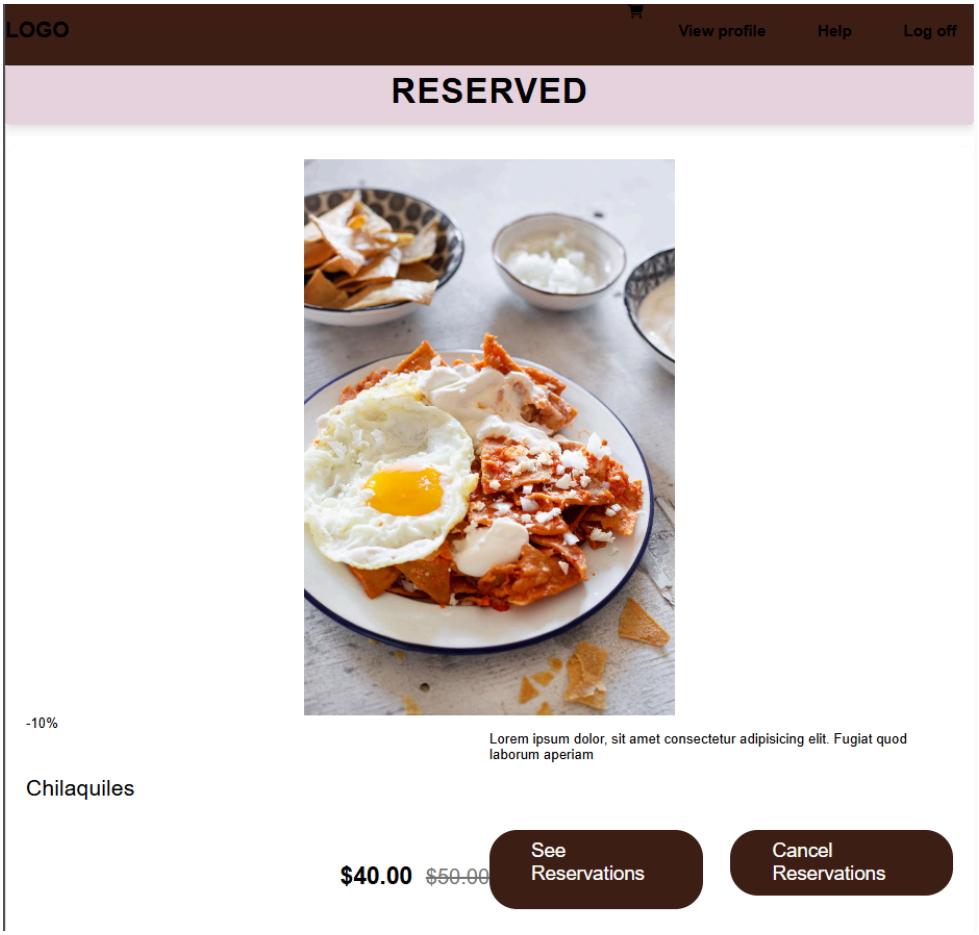
```php
<?php
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $db = mysqli_connect("localhost", "root", "", "Foody");

    if (!$db) {
        die("Database connection failed: " . mysqli_connect_error());
    }

    $name = mysqli_real_escape_string($db, $_POST['name']);
    $description = mysqli_real_escape_string($db, $_POST['description']);
    $category = mysqli_real_escape_string($db, $_POST['category']);
    $ingredients = mysqli_real_escape_string($db, $_POST['ingredients']);
    $price = (float)$_POST['price'];

    if (isset($_FILES['image']) && $_FILES['image']['error'] === UPLOAD_ERR_OK) {
        $image = addslashes(file_get_contents($_FILES['image']['tmp_name']));
    } else {
        $image = null;
    }
    $query = "INSERT INTO dishes (name, description, category, ingredients, price)
    VALUES ('$name', '$description', '$category', '$ingredients', $price)";
    if (mysqli_query($db, $query)) {
        echo "<p>Dish added successfully!</p>";
    } else {
        echo "<p>Error: " . mysqli_error($db) . "</p>";
    }
    mysqli_close($db);
}
?>
```

This PHP code processes a form submission to add a new dish to a MySQL database. It establishes a database connection, retrieves and sanitizes form data (such as the dish name, description, category, ingredients, price, and image), and then inserts this data into the database. The code also handles errors, checking for issues with the database connection or file upload, and provides feedback on whether the dish was successfully added. Finally, it closes the database connection. Overall, this code manages the entire process of securely adding a new dish entry to the database

```
#ingredients-list {
    list-style-type: none;
    padding-left: 0;
    margin-top: 10px;
}

#ingredients-list li {
    background-color: #eee;
    margin: 5px 0;
    padding: 5px;
    border-radius: 5px;
    display: flex;
    justify-content: space-between;
    align-items: center;
}

#ingredients-list li button {
    background-color: red;
    color: #fff;
    border: none;
    padding: 5px;
    border-radius: 3px;
    cursor: pointer;
}
```

This CSS visually modifies a form to create dishes with styles for the title, inputs, buttons and lists. It uses settings to make the form look good on different screen sizes.

Reservation



This interface shows the orders that were previously reserved, the order that was reserved is shown, and there is a button to know more details of the dish and another one to cancel the reservation.

```php
<section class="category-container">
    <h2>Reserved</h2>
    <div class="container-products">
        <?php foreach ($reservations as $reservation): ?>
            <div class="card-product">
                <div class="container-img-reservation">
                    <img src="<?php echo htmlspecialchars($reservation['image']); ?>" alt="<?php echo htmlspecialchars($reservation['name']); ?>">
                    <span class="discount"><?php echo htmlspecialchars($reservation['discount']); ?></span>
                </div>
                <div class="content-card-product">
                    <h3><?php echo htmlspecialchars($reservation['name']); ?></h3>
                    <p><?php echo htmlspecialchars($reservation['description']); ?></p>
                    <p class="price"><?php echo htmlspecialchars($reservation['price']); ?> <span><?php echo htmlspecialchars($reservation['original_price']);
                    <div class="container-options">
                        <span>See Reservations</span>
                        <div class="container-options-cancel">
                            <span>Cancel Reservations</span>
                        </div>
                    </div>
                </div>
            </div>
        <?php endforeach; ?>
    </div>
</section>
```

This PHP and HTML code generates a section displaying a list of reserved items. It uses a loop to iterate over an array of reservations, with each reservation containing details like an image, name, description, price, discount, and original price.

Each reservation is displayed within a structured layout. The image and discount information appear at the top, followed by the name, description, and pricing details. The pricing section shows the discounted price alongside the original price.

```php
<?php
include "/php/header.php";
require "/php/conectbd.php";
$db = connectDB();

$reservations = [];
$deliveredOrders = [];
$canceledOrders = [];
$reservationQuery = "SELECT * FROM reservations";
$result = $db->query($reservationQuery);
if ($result) {
    while ($row = $result->fetch_assoc()) {
        $reservations[] = $row;
    }
}

$deliveredQuery = "SELECT * FROM status_orders";
$result = $db->query($deliveredQuery);
if ($result) {
    while ($row = $result->fetch_assoc()) {
        $deliveredOrders[] = $row;
    }
}

$canceledQuery = "SELECT * FROM canceled";
$result = $db->query($canceledQuery);
if ($result) {
    while ($row = $result->fetch_assoc()) {
        $canceledOrders[] = $row;
    }
}
?>
```

This PHP script retrieves data from three different database tables—reservations, status_orders, and canceled—and organizes the data into separate arrays for later use. After connecting to the database, it defines empty arrays for reservations, deliveredOrders, and canceledOrders to store the data from each table.