# INSTITUTO SAN IGNACIO DE LOYOLA



## EXAMEN FINAL 2607

Propuesta de Página Web “*Tagliatore*”

**CURSO:**

Integración de Aplicaciones

**CARRERA:**

Desarrollo de Software

**DOCENTE:**

JORGE EDUARDO CASTAÑEDA ALBAN

**GRUPO 3:**

Brigith Estefanny Selena Gamero Llanto, 75130288

Ian Eros Mio Ayala, 72865788

Max Jose Murillo Vargas, 46970969

**LIMA – PERÚ**

**2024**

**Índice.**

**Índice 2**

**Introducción 3**

**Resumen 4**

**Marco Teórico 5**

**Descripción del caso de estudio 6**

**Objetivos 7**

**Desarrollo 8**

**Especificaciones del Requerimiento de Software 8**

**Análisis de Software 10**

**Diseño del modelo de la Base de Datos 10**

**Diseño del API bajo metodología RUP 11**

**Recomendaciones 29**

**Conclusiones 30**

**Referencias Bibliográficas 31**

**Introducción.**

En el presente informe se detalla la propuesta página web para el restaurante italiano “Tagliatore”.

La propuesta consiste de una página web dinámica implementada con el framework **Angular**, que es muy eficiente para aplicaciones de una sola página. La aplicación web se conecta a una API RESTful, diseñada con anterioridad, que proporciona los datos y servicios necesarios para su funcionamiento mediante la transferencia de información en formato **JSON**.

El informe detalla cada etapa del desarrollo, desde la definición de requisitos y el análisis de la API, hasta el diseño e implementación de los componentes y servicios en Angular. Se describen las funcionalidades implementadas, los desafíos enfrentados y las estrategias aplicadas para lograr una solución eficiente y de alto rendimiento.

**Resumen.**

Este informe documenta la creación de un sistema integral para el restaurante “Tagliatore”, diseñado para facilitar la gestión de sus procesos internos.

El sistema incluye una página web dinámica desarrollada con Angular, que se conecta a una API RESTful implementada con Node.js y MongoDB. Entre las funcionalidades principales se encuentran la gestión de órdenes, clientes, platillos, categorías y meseros.

La API RESTful utiliza métodos HTTP para el manejo de datos, mientras que las contraseñas de los usuarios se protegen mediante encriptación con JsonWebToken y Bcrypt. También se incorporó un sistema de mensajería en tiempo real para mejorar la comunicación entre los clientes y el restaurante.

El desarrollo se realizó siguiendo la metodología RUP, lo que permitió una solución estructurada y escalable, asegurando la calidad en cada etapa del proyecto.

**Marco Teórico.**

El desarrollo de aplicaciones web modernas se basa en principios de modularidad, escalabilidad y eficiencia. Angular, como framework de frontend, permite la creación de aplicaciones de una sola página (SPA), ofreciendo al usuario una experiencia fluida y rápida.

MongoDB, por su naturaleza NoSQL, se adapta a sistemas que requieren manejar grandes volúmenes de datos no estructurados, mientras que las APIs RESTful permiten la comunicación eficiente entre el frontend y el backend a través de operaciones CRUD.

Por otro lado, la metodología RUP divide el desarrollo en fases iterativas, enfocándose en la documentación de requisitos y asegurando la calidad del producto mediante pruebas continuas. Este enfoque resulta esencial para proyectos con múltiples funcionalidades y usuarios finales exigentes.

**Descripción del caso de estudio.**

*Tagliatore* es un restaurante de comida italiana especializado en gastronomía Toscana y Romana. Sus procesos se han vuelto muy complejos y necesitan un sistema que permita gestionar sus operaciones diarias. Se nos solicitó la elaboración de una página web con un API acompañante para poder realizar operaciones CRUD respecto a ciertos aspectos clave del restaurante.

Se tomó la decisión de para el back end implementar una API RESTful que permita realizar mantenimientos de Platillos, Clientes, Órdenes, Categorías y Meseros en una base de datos de **MongoDB**. La API usará métodos **HTTP** para permitir a la página web interactuar con la Base de Datos.

La tabla de **Meseros** fue diseñada como reemplazo de una tabla de Usuarios originalmente, por lo que se utilizará la información de Meseros para diseñar una página de Login. Las contraseñas son encriptadas mediante el uso de **JsonWebToken** y la librería **Bcrypt**.

Adicionalmente se desea incluir funcionalidad de mensajería en tiempo real (**Chat**) disponible a clientes. (Añadir más información)

Para el front end se solicitó la utilización del framework para páginas web **Angular** y **Bootstrap**. Se propone la creación de componentes para Platillos, Clientes, Órdenes, Categorías y Meseros con sus propios subcomponentes para el manejo de los métodos Create y Update. Esta estructura es modular y escalable si el sistema necesita la adición de nuevas tabla.

Se espera que la adición de esta página permita a *Tagliatore* modernizar su infraestructura tecnológica y mejorar sus procesos, así como sentar una base desde la cual el sistema pueda ser expandido a futuro.

**Objetivos**

* Crear una solución tecnológica que permita al restaurante "Tagliatore" gestionar de manera eficiente sus procesos operativos, como la gestión de órdenes, clientes, platillos, categorías y meseros.
* Implementar una página web utilizando Angular que ofrezca una experiencia fluida y atractiva para los usuarios, permitiendo la interacción en tiempo real con los servicios del restaurante.
* Desarrollar una API RESTful que brinde los servicios necesarios para manejar datos de clientes, platillos, órdenes, usuarios, meseros y categorías, utilizando métodos HTTP para operaciones CRUD.
* Implementar un sistema de autenticación mediante JsonWebToken (JWT) y asegurar las contraseñas de los usuarios utilizando la librería Bcrypt para proteger la información sensible.
* Añadir un sistema de chat en tiempo real para mejorar la comunicación entre los clientes y el restaurante, facilitando el proceso de pedidos y atención.
* Desarrollar la aplicación de manera modular para facilitar la expansión futura, permitiendo la adición de nuevas funcionalidades conforme las necesidades del restaurante crezcan.

**Desarrollo.**

* **Especificaciones del Requerimiento de software.**

#### Requisitos Funcionales

1. **Gestión de Platillos:**
   * El sistema debe permitir la creación, visualización, actualización y eliminación de platillos.
   * Cada platillo debe contener los siguientes datos: nombre, descripción, precio, categoría y una imagen representativa.
   * Los platillos estarán asociados a una categoría, que permitirá filtrar y organizar los platillos de forma eficiente.
2. **Gestión de Clientes:**
   * El sistema debe permitir registrar, consultar, actualizar y eliminar información de los clientes del restaurante.
   * La información de los clientes debe incluir: nombre, apellido, correo electrónico y teléfono.
   * Los clientes pueden realizar pedidos en el sistema, asociándose a una orden.
3. **Gestión de Órdenes:**
   * El sistema debe permitir crear, consultar, actualizar y eliminar órdenes.
   * Cada orden estará asociada a un cliente, un mesero y a uno o varios platillos.
   * Las órdenes deberán tener un estado (por ejemplo: pendiente, en preparación, servida, completada).
4. **Gestión de Meseros :**
   * El sistema debe permitir la gestión de meseros.
   * Los meseros tendrán un rol asignado , lo que determinará el acceso y las acciones que pueden realizar en el sistema.
   * Los meseros podrán visualizar, gestionar y actualizar las órdenes asociadas a ellos.
5. **Gestión de Categorías:**
   * El sistema debe permitir la gestión de categorías de platillos, que permiten agrupar los platillos de acuerdo a su tipo.
   * Las categorías podrán ser creadas, consultadas, actualizadas y eliminadas.
6. **Autenticación de Usuarios:**
   * El sistema debe contar con un sistema de autenticación basado en JSON Web Tokens para garantizar el acceso seguro a las funcionalidades del sistema.
   * Las contraseñas de los usuarios deben ser almacenadas de manera segura utilizando encriptación con Bcrypt.
   * Los meseros podrán iniciar sesión con su correo y contraseña, y se les asignará un token de autenticación.
7. **Sistema de Mensajería en Tiempo Real:**
   * El sistema debe permitir la comunicación en tiempo real entre los clientes y los meseros a través de un sistema de chat.
   * Este chat debe ser accesible desde la interfaz web, y permitir la interacción en tiempo real durante el proceso de pedido.

#### Requisitos No Funcionales

1. **Seguridad:**
   * La aplicación debe cumplir con estándares de seguridad, asegurando la confidencialidad de las contraseñas mediante el uso de técnicas de encriptación como Bcrypt.
   * Las sesiones de usuario deben estar protegidas utilizando JWT para prevenir accesos no autorizados.
2. **Rendimiento:**
   * El sistema debe ser capaz de manejar un gran número de usuarios y operaciones simultáneas, especialmente durante las horas de mayor demanda.
   * La API debe ser eficiente en la gestión de solicitudes y respuestas, minimizando el tiempo de respuesta.
3. **Escalabilidad:**
   * El sistema debe ser escalable para poder agregar nuevas funcionalidades en el futuro sin afectar el rendimiento del sistema.
4. **Interfaz de Usuario:**
   * La interfaz de usuario debe ser intuitiva y fácil de usar, permitiendo a los meseros, administradores y clientes interactuar con el sistema de forma eficiente.
   * La página debe ser responsiva, adaptándose a diferentes dispositivos).
5. **Base de Datos:**
   * La base de datos debe ser eficiente y escalable. Se utilizará MongoDB, una base de datos NoSQL, que permite manejar grandes volúmenes de datos no estructurados.
   * Los datos deben ser almacenados de manera segura y eficiente, garantizando la integridad y disponibilidad de la información.

#### Requisitos Técnicos

1. **Tecnologías Utilizadas:**
   * **Frontend:** Angular, Bootstrap.
   * **Backend:** Node.js, Express.
   * **Base de Datos:** MongoDB.
   * **Autenticación:** JSON Web Tokens (JWT) y Bcrypt.
   * **Mensajería en Tiempo Real:** Socket.io o similar para implementación de chat.
2. **API:**
   * La API será RESTful, permitiendo la comunicación entre el frontend y el backend.
   * Los principales endpoints de la API incluirán:
     + **/platillos** (GET, POST, PUT, DELETE): Gestión de platillos.
     + **/meseros** (GET, POST, PUT, DELETE): Gestión de datos de meseros.
     + **/ordenes** (GET, POST, PUT, DELETE): Gestión de órdenes.
     + **/categorias** (GET, POST, PUT, DELETE): Gestión de categorías de platillos.
     + **/clientes** (GET, POST, PUT, DELETE): Gestión de datos de clientes.
     + **/auth** (POST): Endpoint para autenticación.
     + **/login** (POST): Endpoint para el inicio de sesión de los usuarios.

* **Análisis de Software.**

1. **Diseño del modelo de la Base de Datos.**

* **Platillos**: Incluye datos como nombre, descripción, precio, categoría a la que pertenece e imagen del platillo.
* **Órdenes**: Contiene información sobre el cliente que realiza la orden, el mesero que la atiende, los platillos solicitados (con cantidad) y el estado de la orden.
* **Clientes**: Almacena datos personales de los clientes, como nombre, apellido, correo electrónico y teléfono.
* **Meseros**: Incluye información personal de los meseros, como nombre, contraseña (encriptada con Bcrypt), apellido, correo electrónico y teléfono.
* **Categorías**: Agrupa los platillos según su tipo, con atributos como nombre y descripción de la categoría.

### Relaciones clave:

* **Cliente ↔ Orden**: Un cliente puede tener varias órdenes.
* **Mesero ↔ Orden**: Un mesero puede atender varias órdenes.
* **Platillo ↔ Orden**: Una orden contiene varios platillos.
* **Platillo ↔ Categoría**: Cada platillo pertenece a una categoría.

1. **Diseño del API bajo metodología RUP.**

La API se diseñó siguiendo los principios de RUP, estructurando el desarrollo en fases:

**Inicio:**

-Identificación y definición de los requisitos funcionales, como el manejo de platillos, clientes, órdenes y usuarios.

-Diseño inicial de los endpoints de la API, garantizando que cubran las operaciones CRUD para las entidades principales.

**Elaboración:**

Desarrollo de los métodos CRUD (Create, Read, Update, Delete) para cada entidad:

* **Platillos**: Gestión de los platillos ofrecidos por el restaurante.
* **Clientes**: Manejo de datos de clientes registrados.
* **Órdenes**: Procesamiento de las órdenes realizadas.
* **Mesero**:Gestión de datos personales y asignación de órdenes a cada mesero, Autenticación y manejo de roles (ADMIN, MESERO).
* **Categoría**:Organización de los platillos en categorías para facilitar su gestión y filtrado.

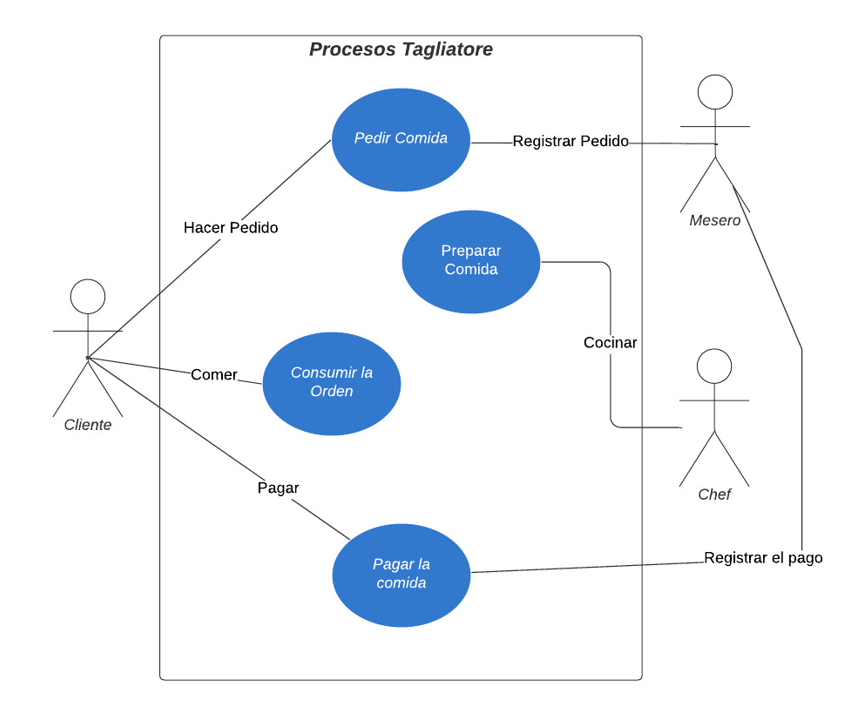
**Construcción:**

-Implementación de pruebas unitarias para verificar la funcionalidad de los endpoints y asegurar la integridad de los datos.

-Uso de herramientas como Postman para pruebas manuales de la API**.**

**Transición:** Despliegue del sistema y ajuste según las necesidades del cliente.

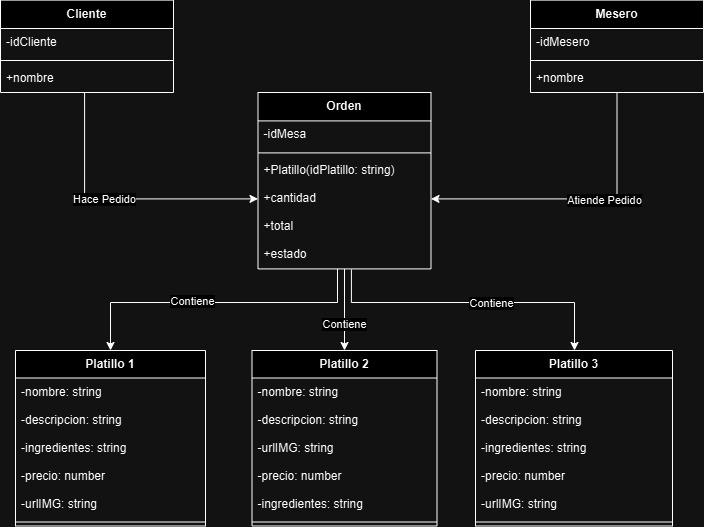
**Diagrama de Casos de Uso:**

****

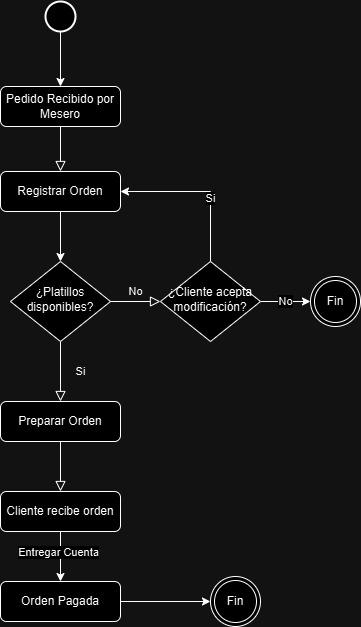
**Diagrama de Clases:**

****

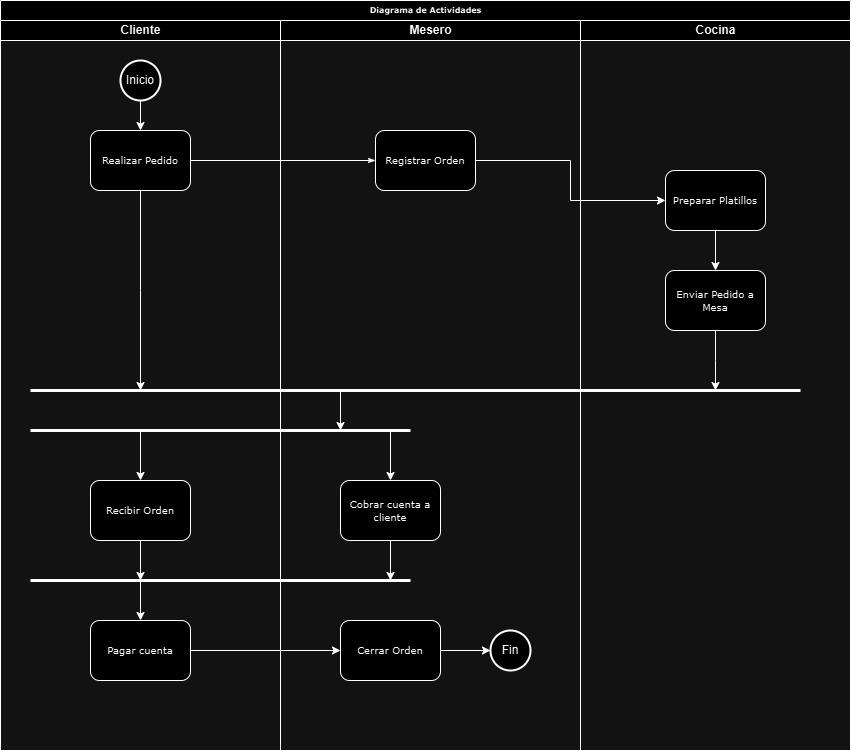
**Diagrama de Objetos:**

****

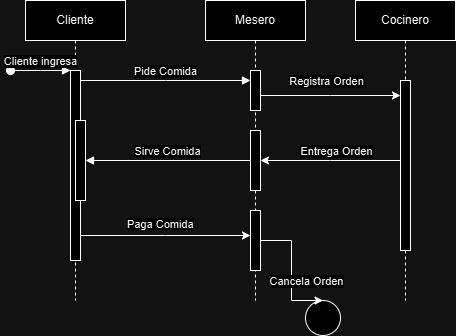
**Diagrama de Estados:**

****

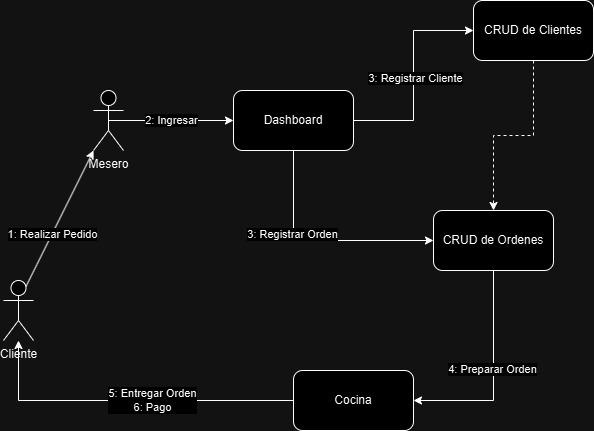
**Diagrama de Actividades:**

****

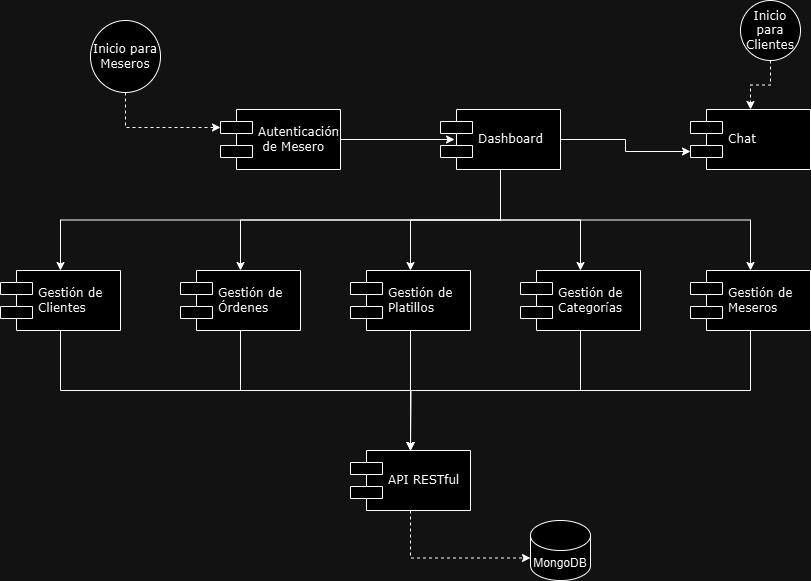
**Diagrama de Secuencia:**

****

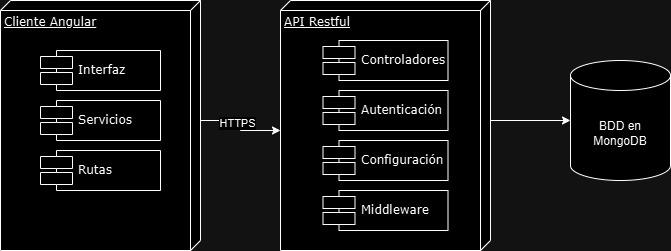
**Diagrama de Colaboración:**

****

**Diagrama de Componentes:**

****

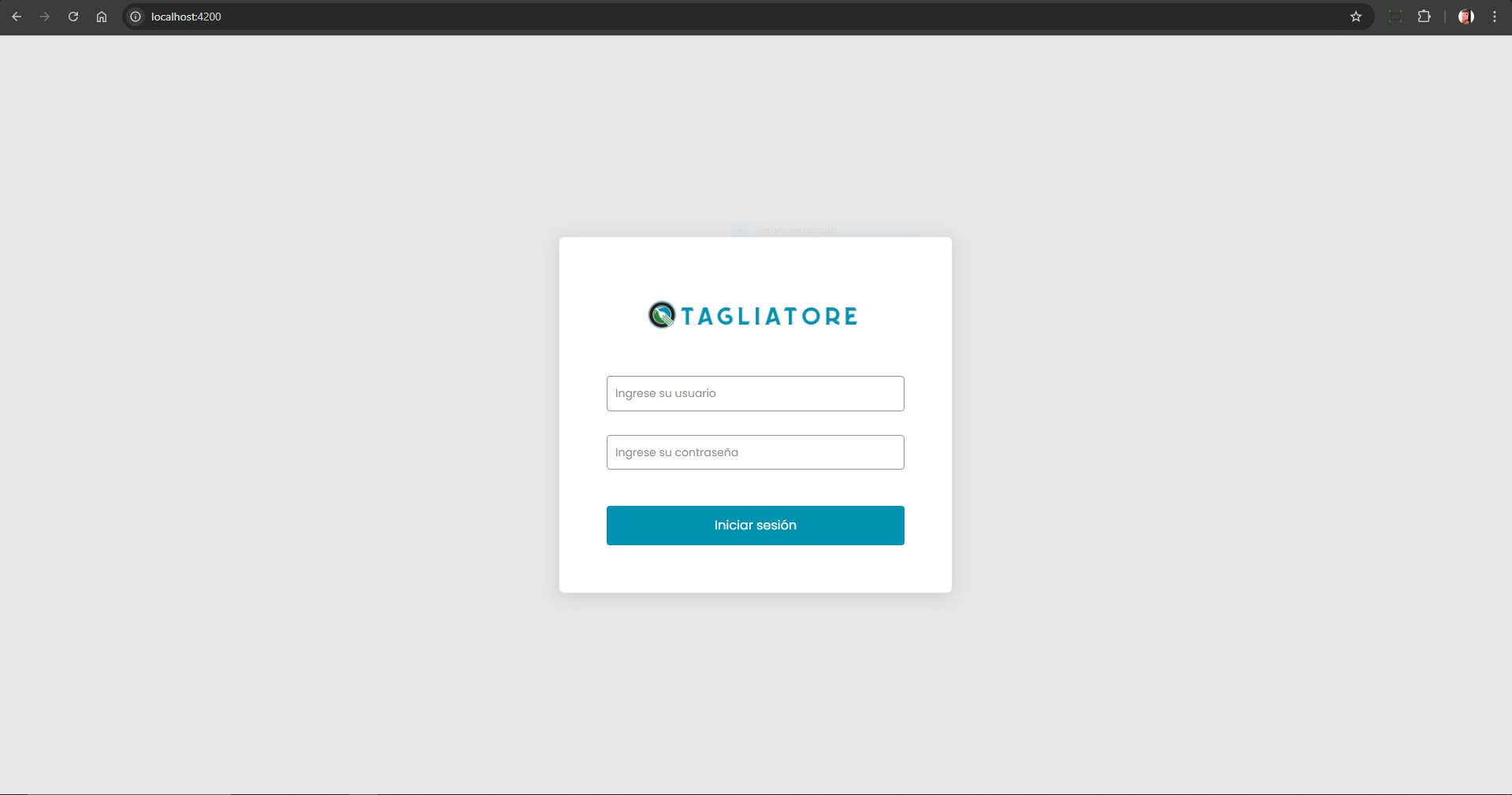
**Diagrama de Despliegue:**

****

**INTERFAZ DE USUARIO**

**LOGIN**

En el proceso de inicio de sesión, el usuario deberá ingresar sus credenciales, que incluyen su correo electrónico y contraseña. La seguridad se garantiza mediante el uso de JWT (JSON Web Token), que será almacenado en una cookie para su uso posterior.

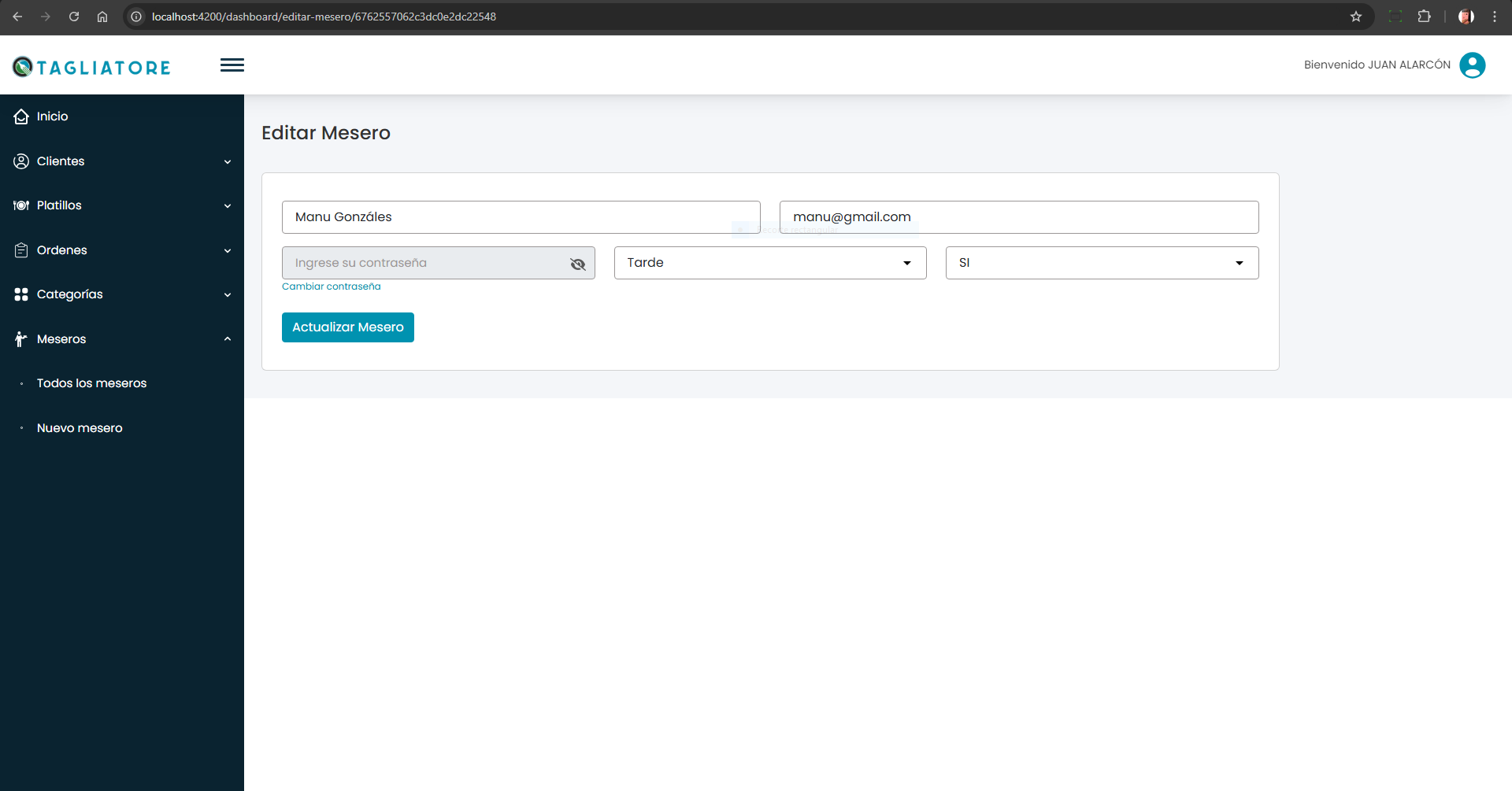
****

**MESEROS**

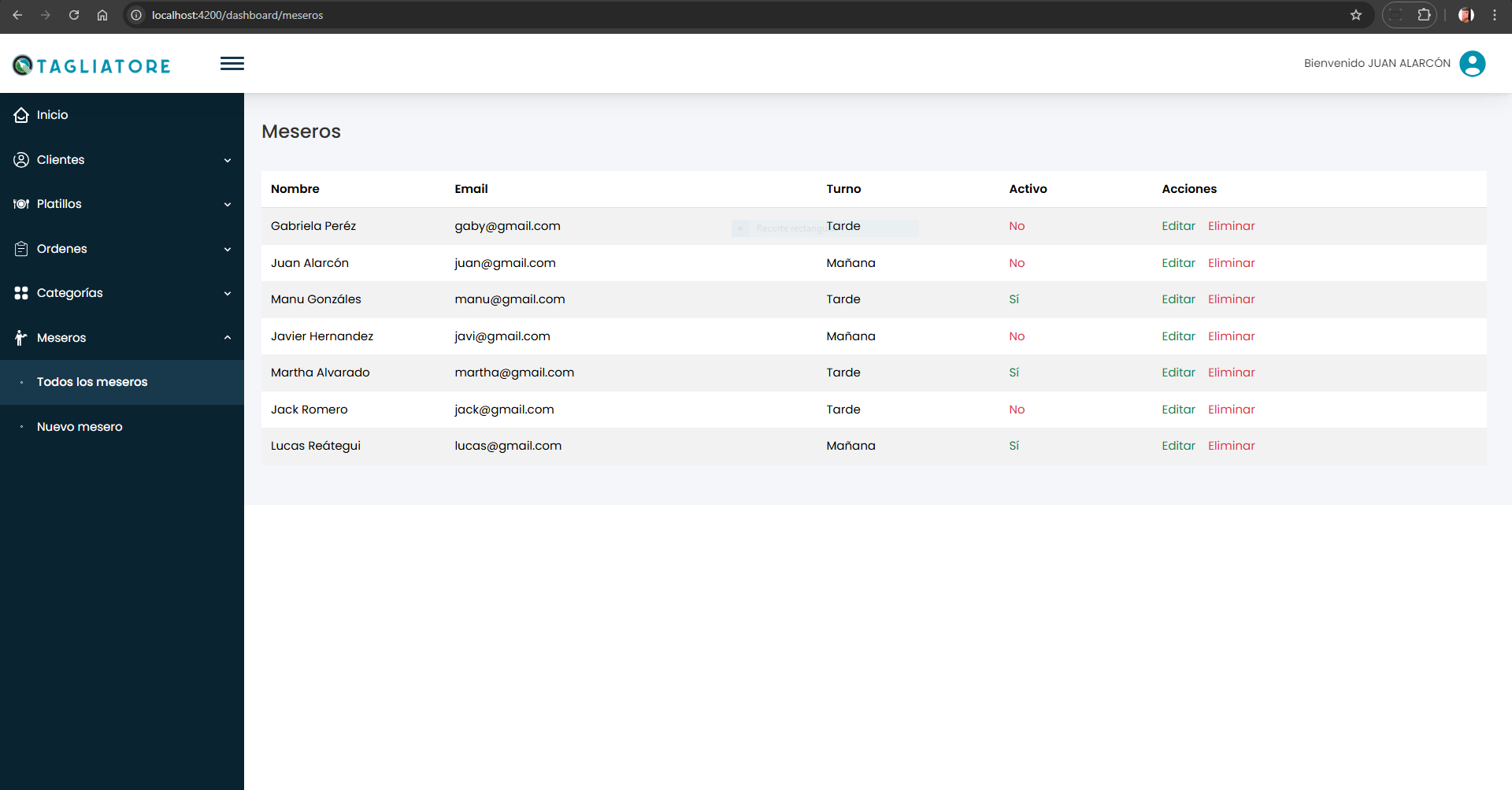
**Crear meseros**: Se solicitan los siguientes datos para crear un mesero: nombre, correo electrónico, contraseña y turno, para que pueda iniciar sesión.

****

**Actualizar meseros**: Permite actualizar la contraseña, cambiar el nombre, el correo electrónico y cambiar el estado del mesero si no esta disponible.

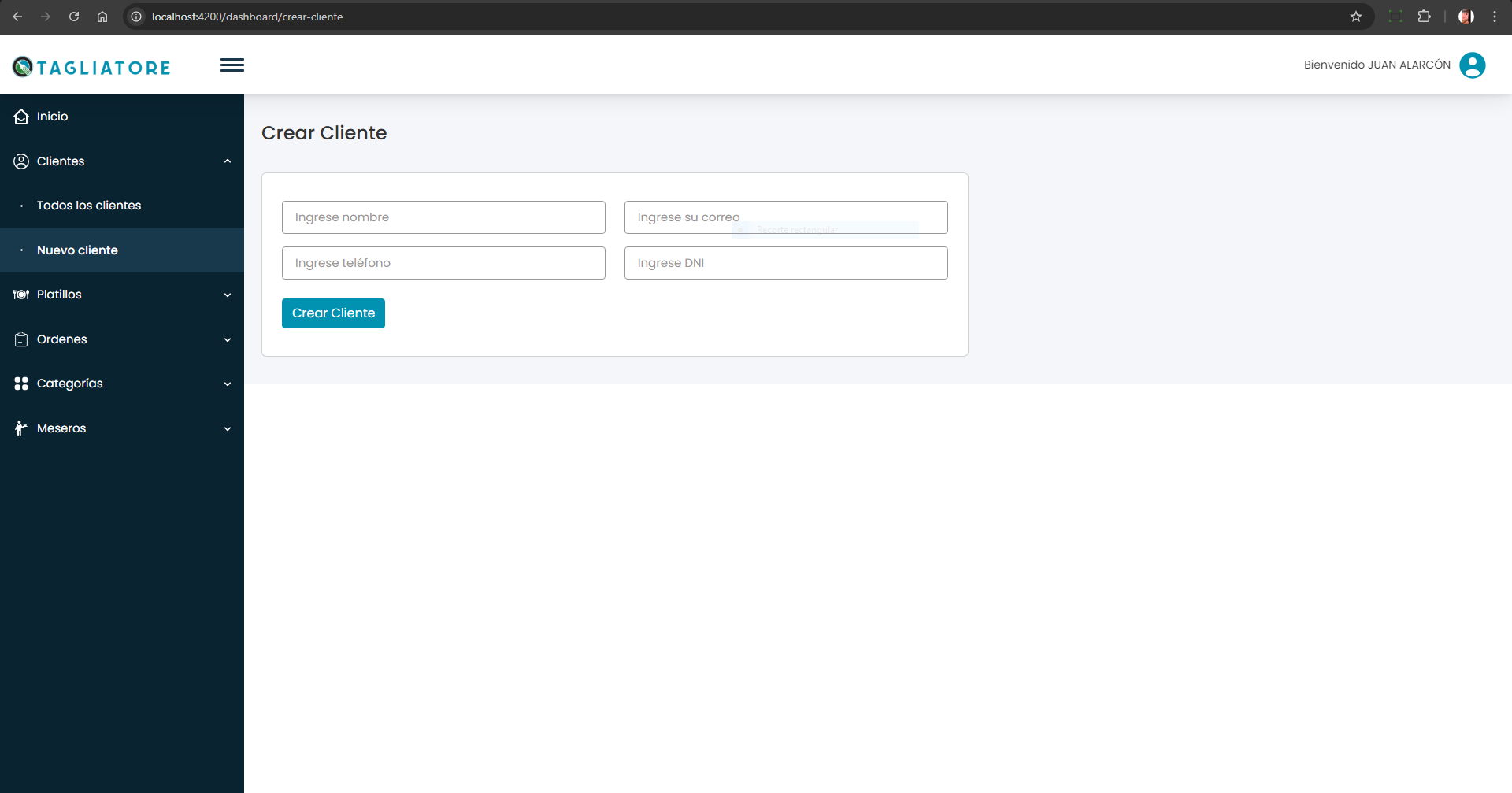
****

**Lista de meseros** Trae toda la lista de mesera y que nos permite visualizar nombre, email, turno , el estado del mesero y las acciones que nos permite editar y eliminar.

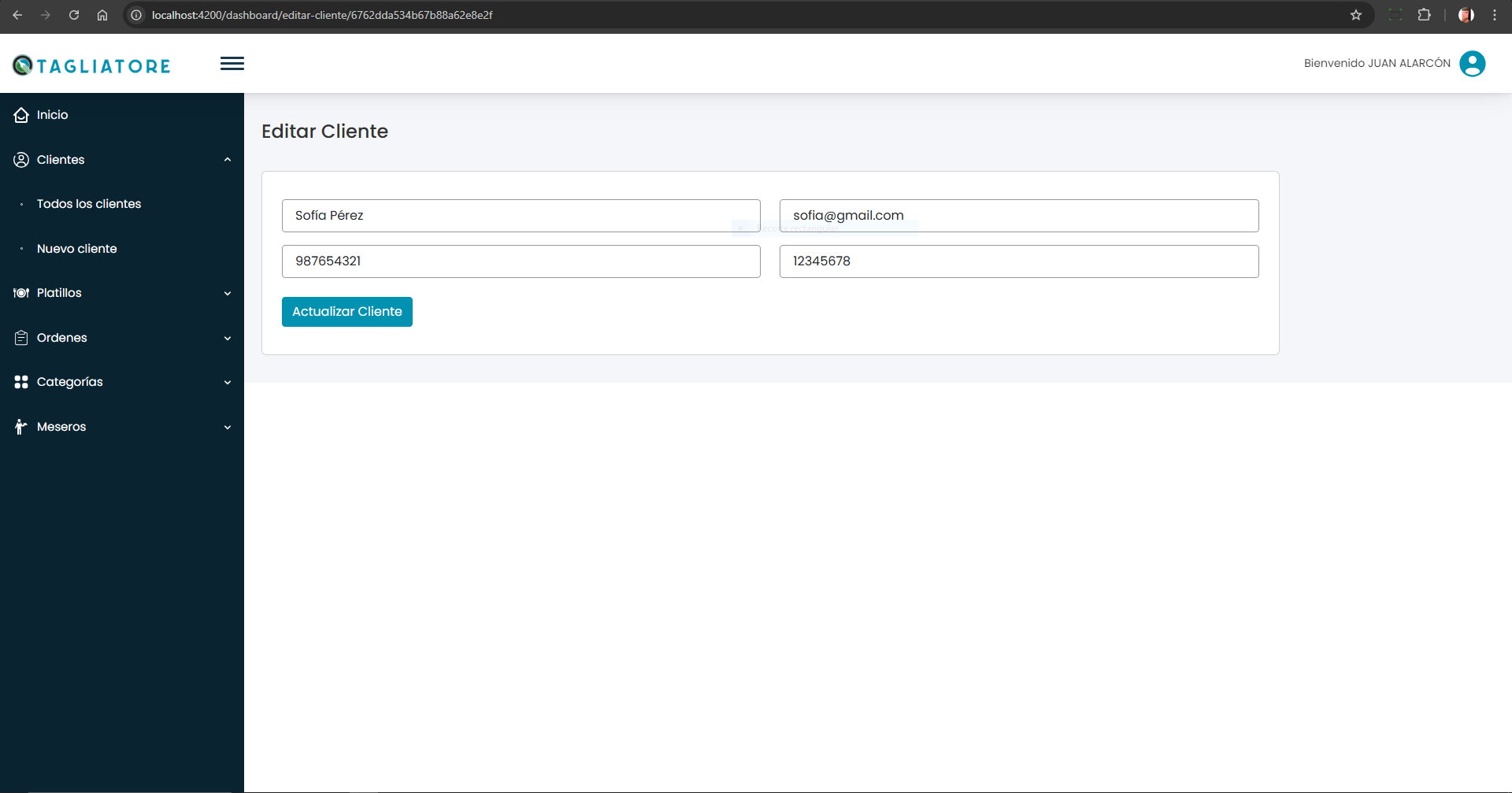
****

**CLIENTE**

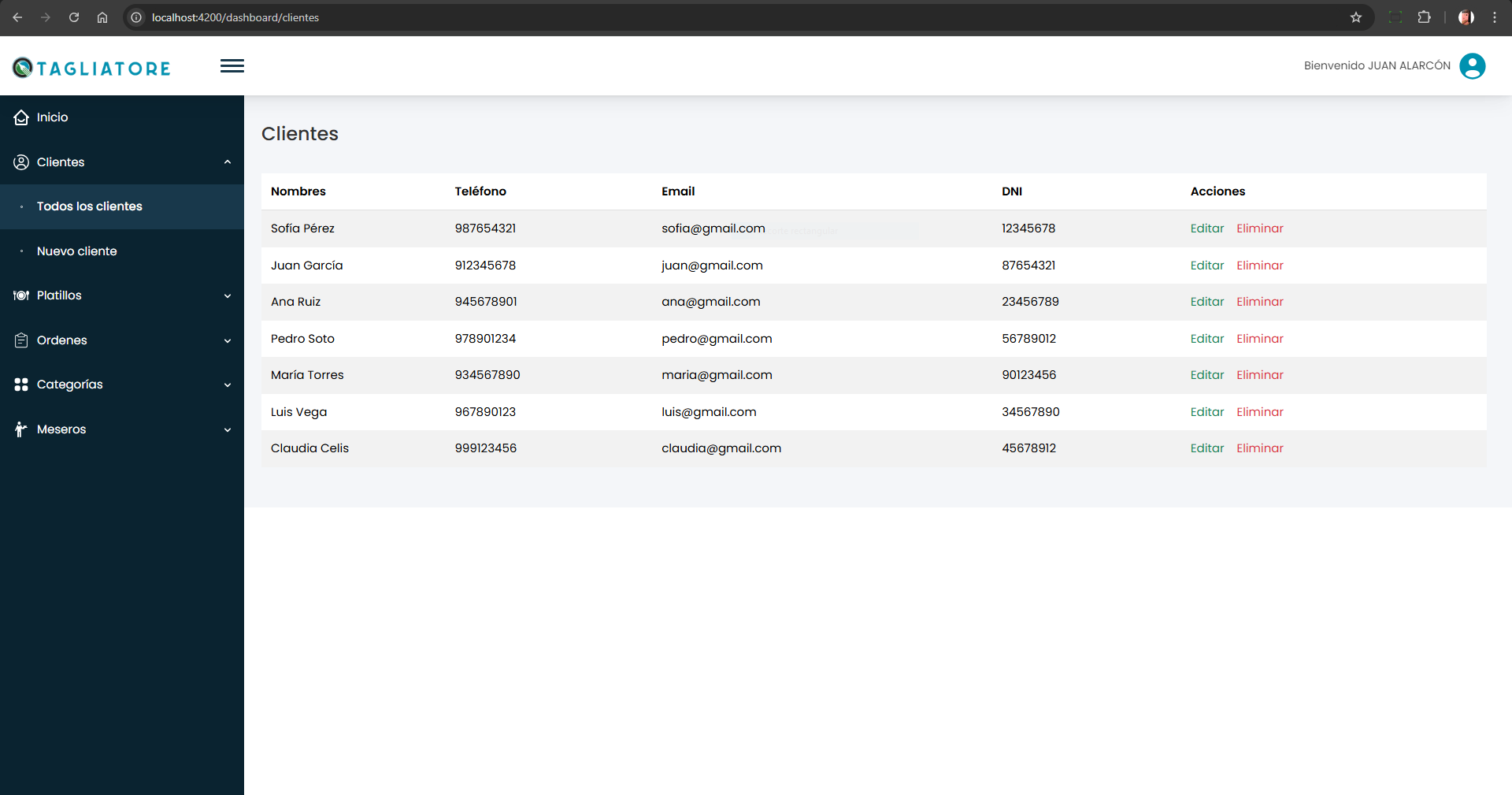
**Crear cliente**: Se solicita ingresar los siguientes nombre, email, número telefónico y dni.



**Actualizar Cliente**: Permite actualizar el nombre, email, numero telefonico y el DNI

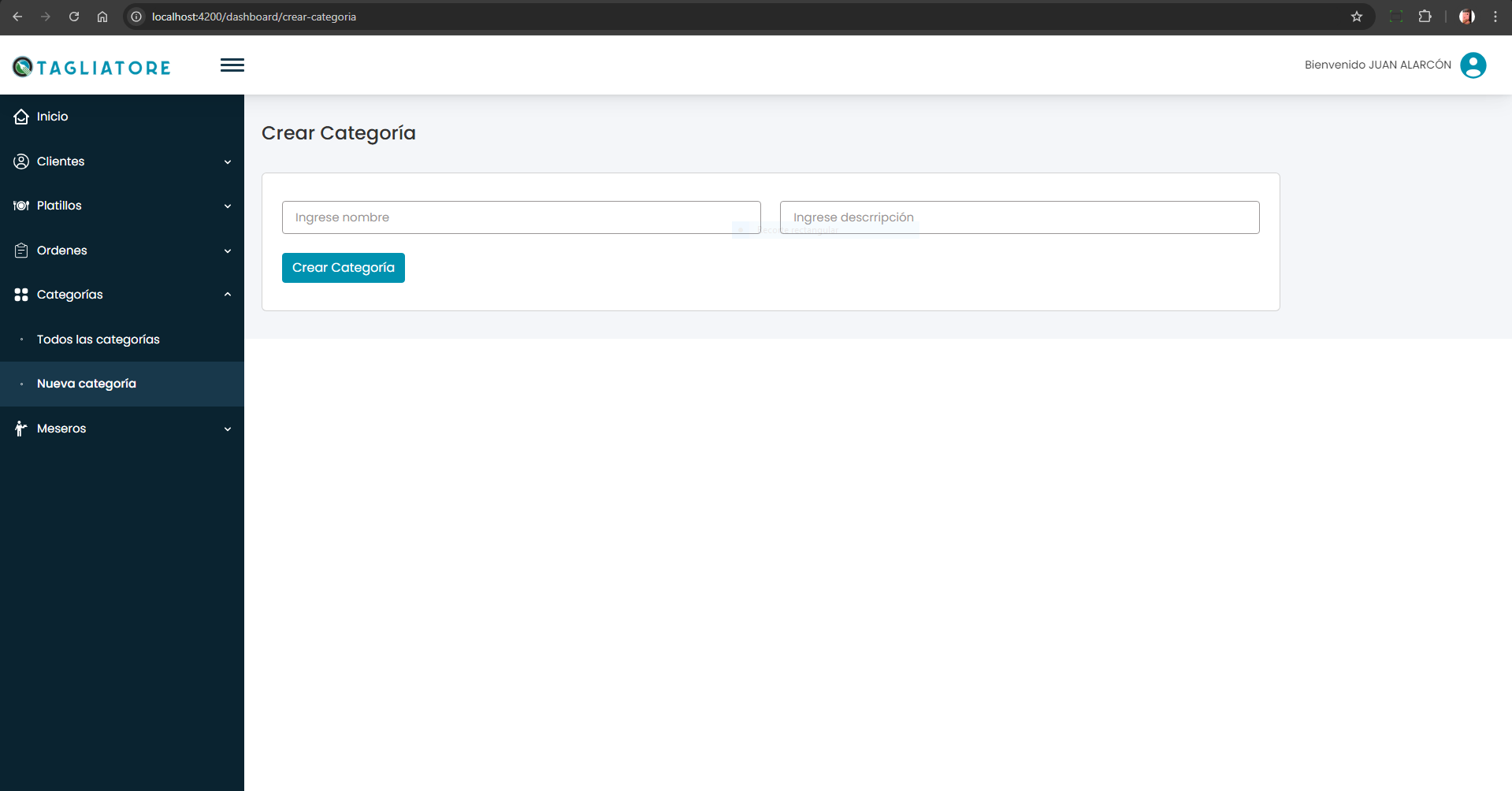
****

**Lista de clientes**:Trae toda la información de cada uno de los cliente y nos permite las acciones de eliminar y editar .

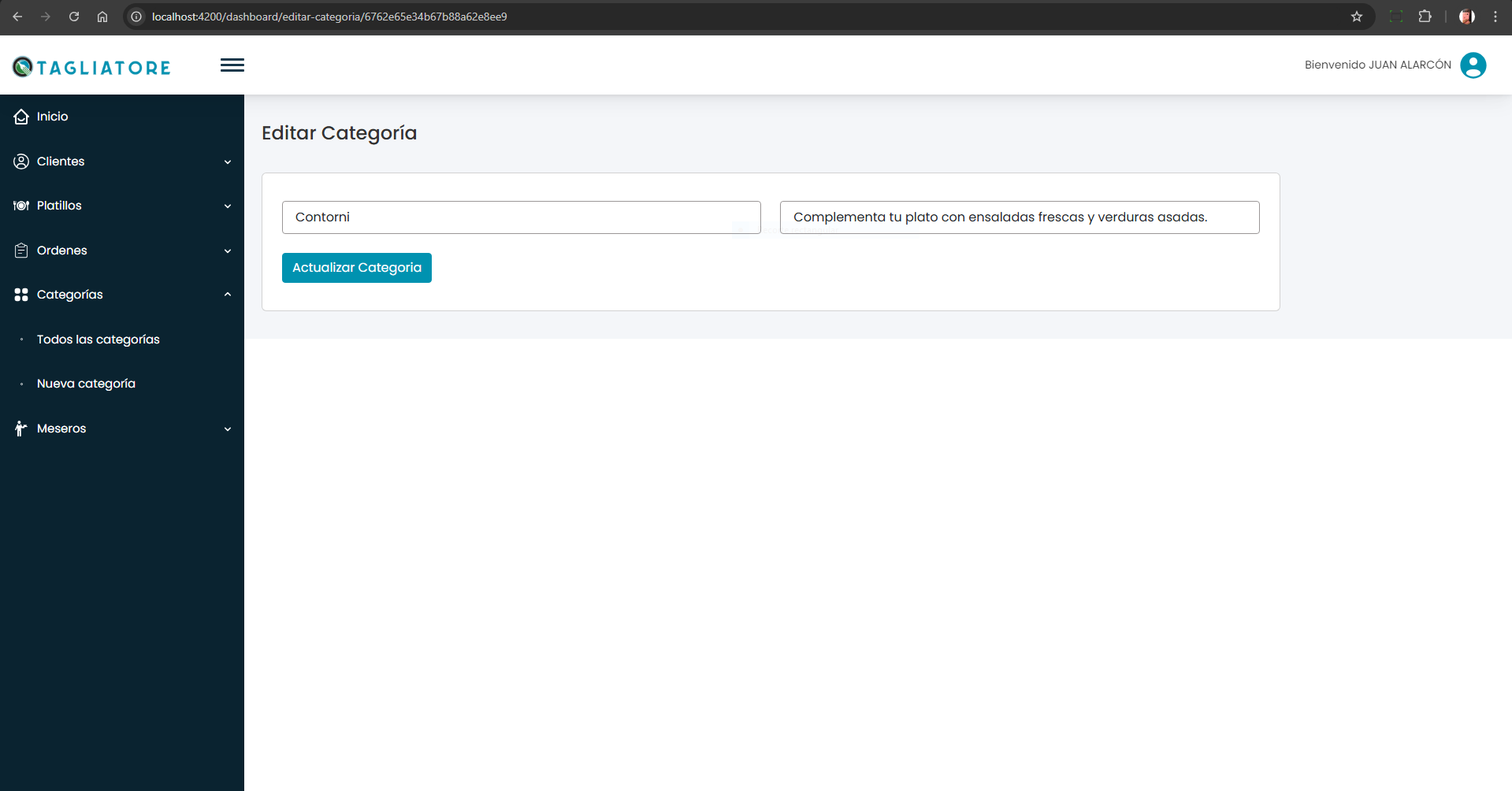
****

**CATEGORÍA**

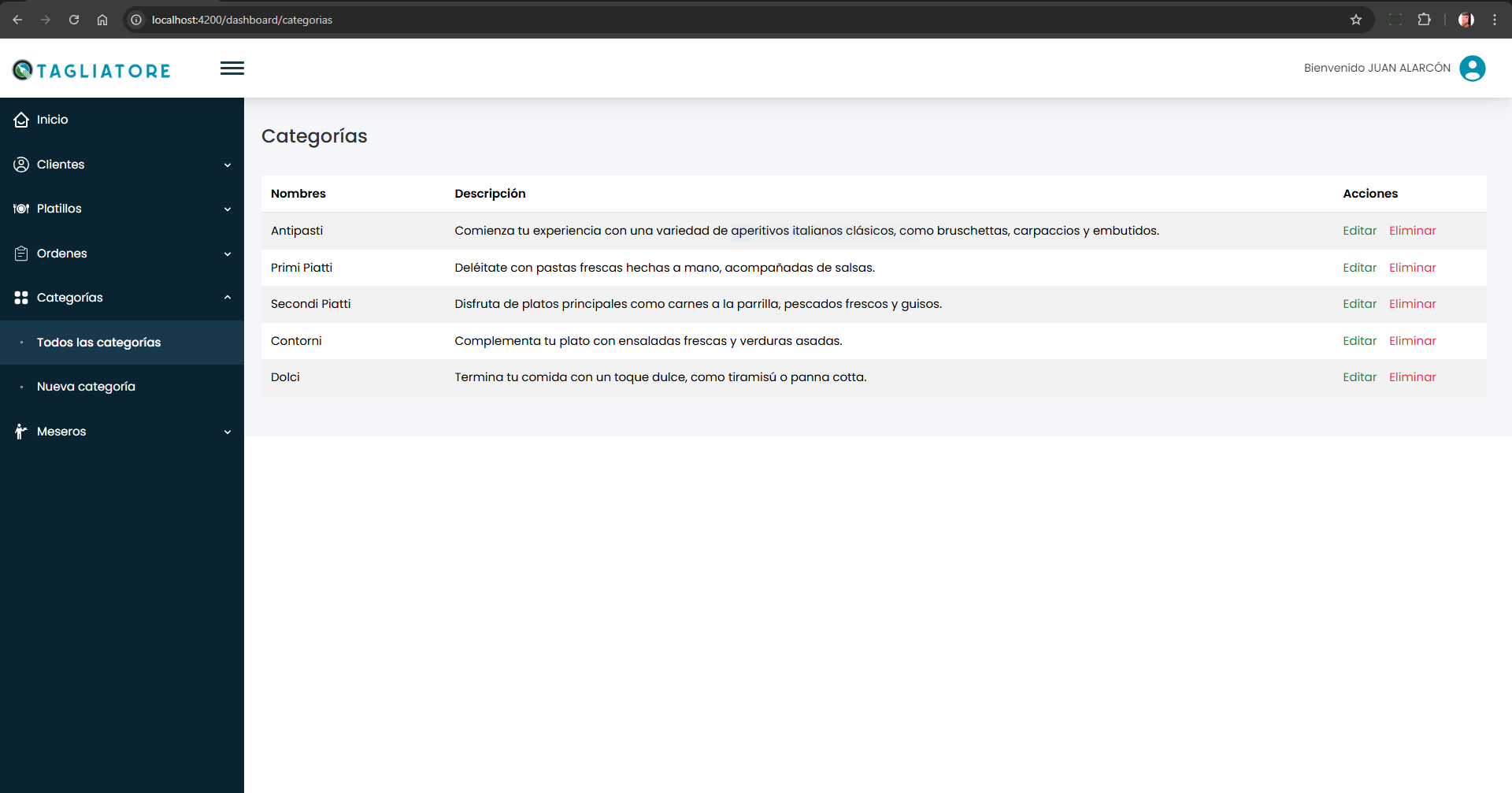
**Crear cliente**: Se solicita ingresar el nombre de la categoría y la descripción.

****

**Actualizar categoría**: Permite editar los campos nombre y la descripción

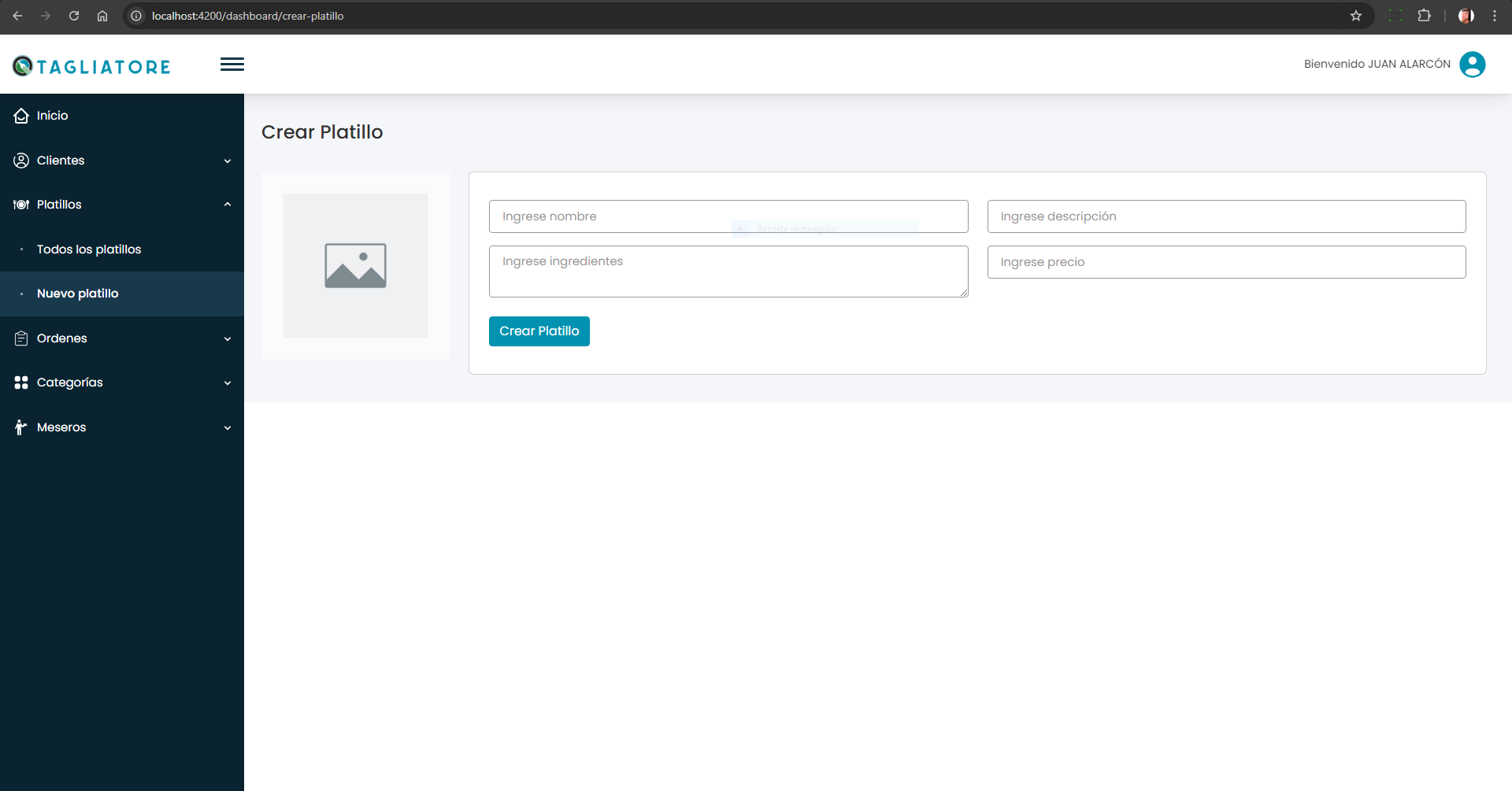
****

**Lista categoría**: Trae todos los datos de la categoría y nos permite las acciones de editar y eliminar.

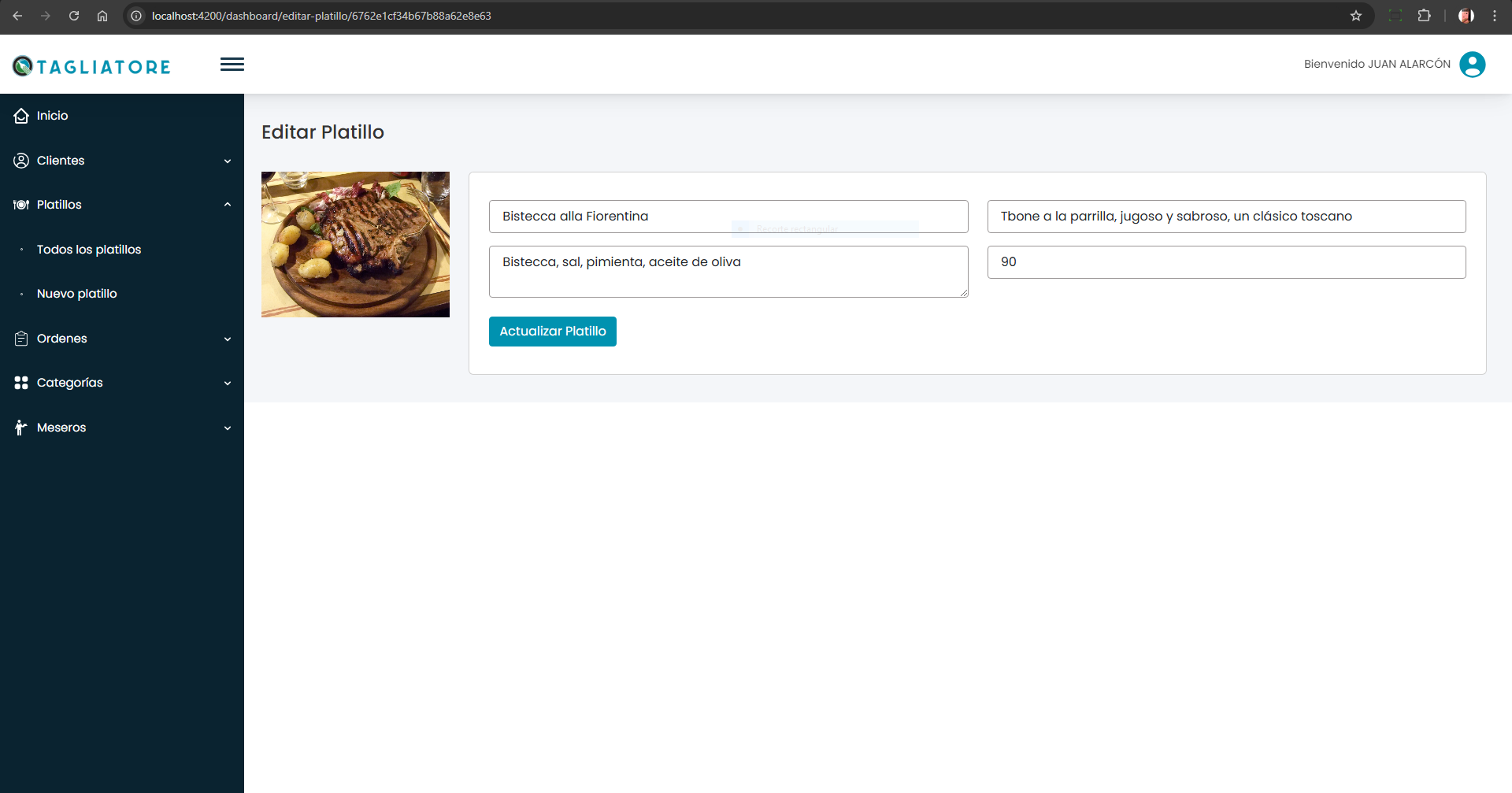
****

**PLATILLOS**

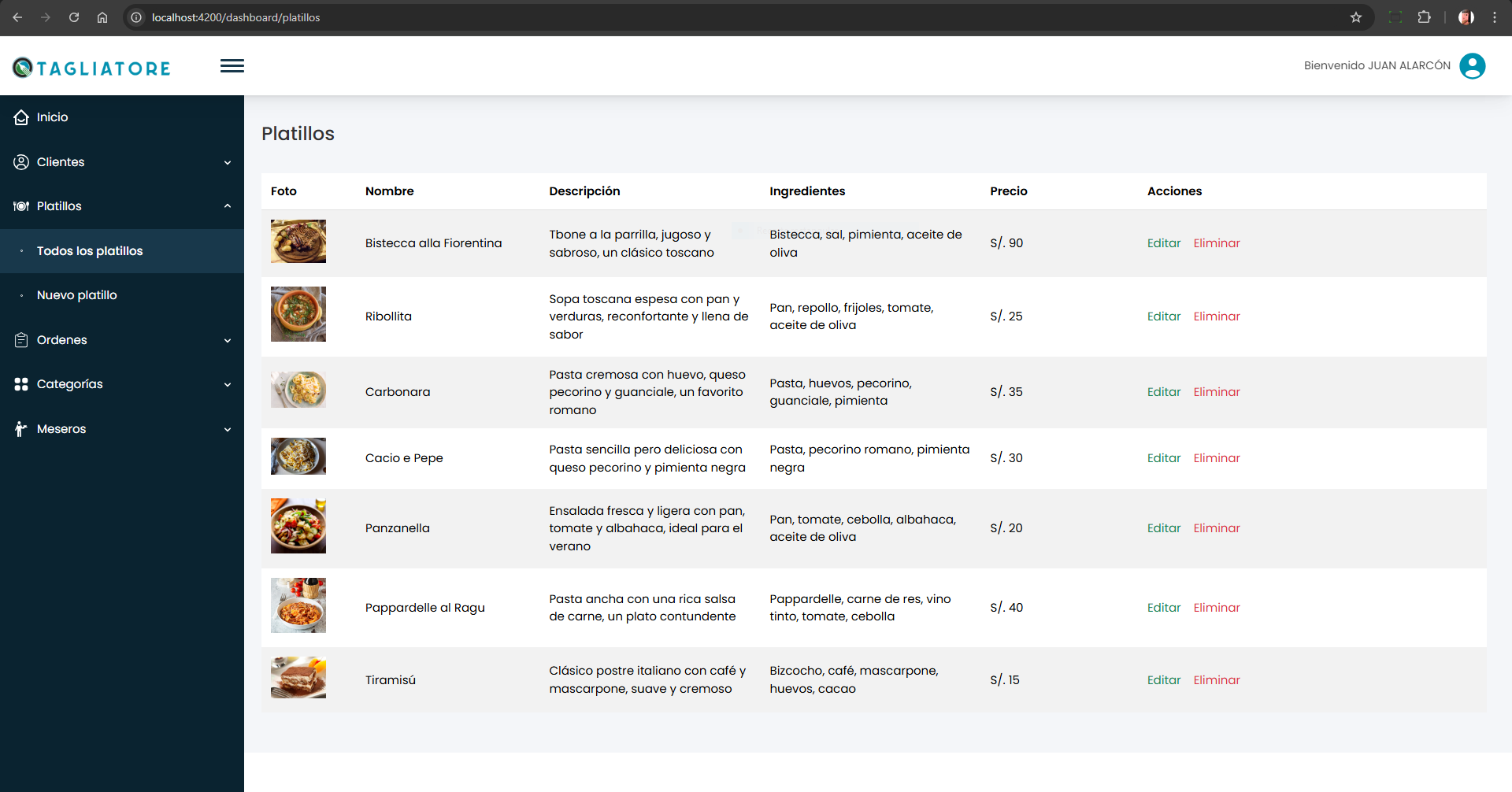
**Crear platillo:** Se solicita ingresar imagen del platillo, nombre del platillo, descripción del platillo, ingresar los ingredientes y el precio.

****

**Actualizar platillo:** Permite editar la imagen, el titulo, la descripción, del ingrediente y el precio.

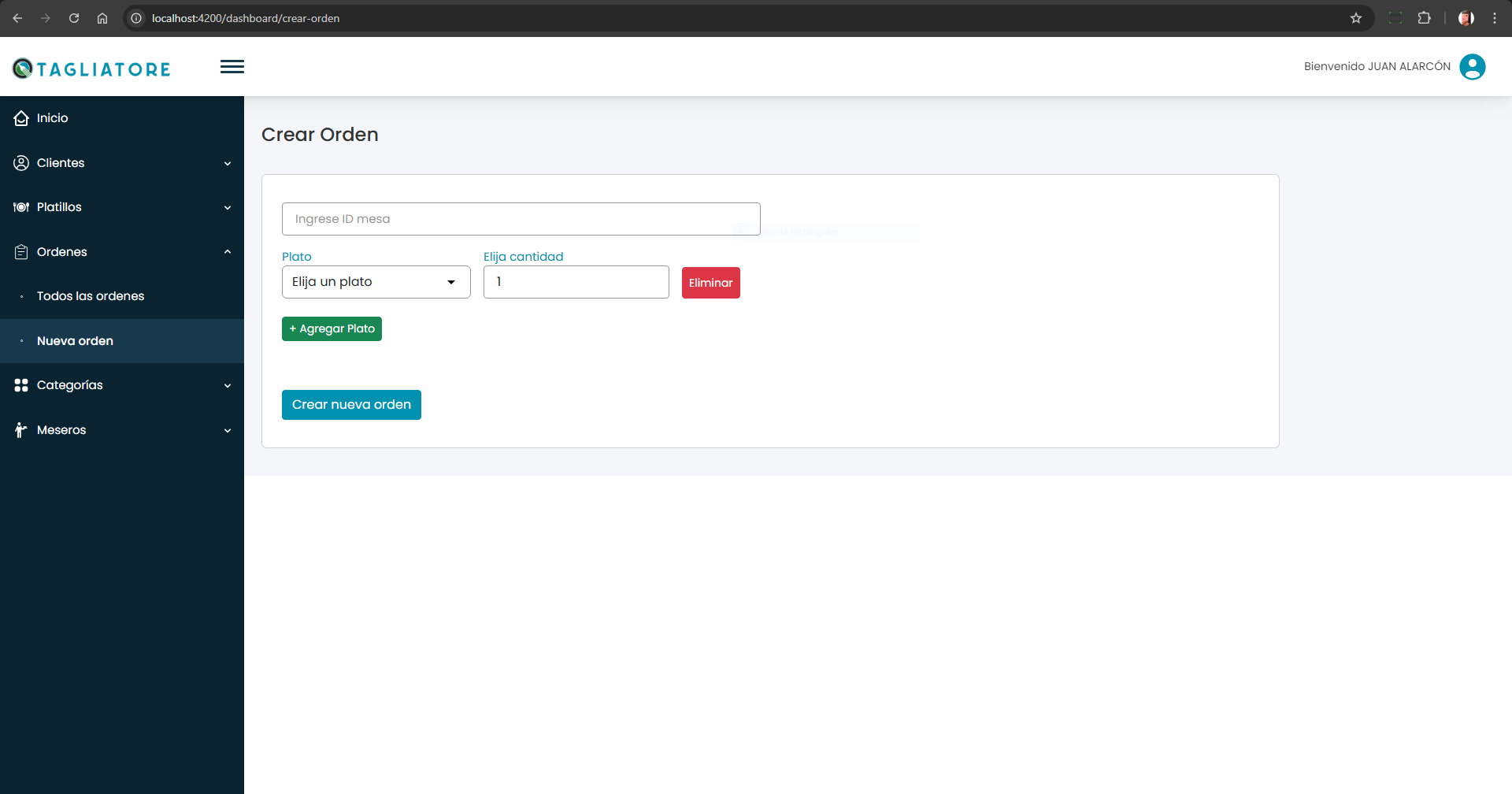
****

**Lista platillo**: Trae todos los platillos con su datos y nos permite la acciones de eliminar y editar.

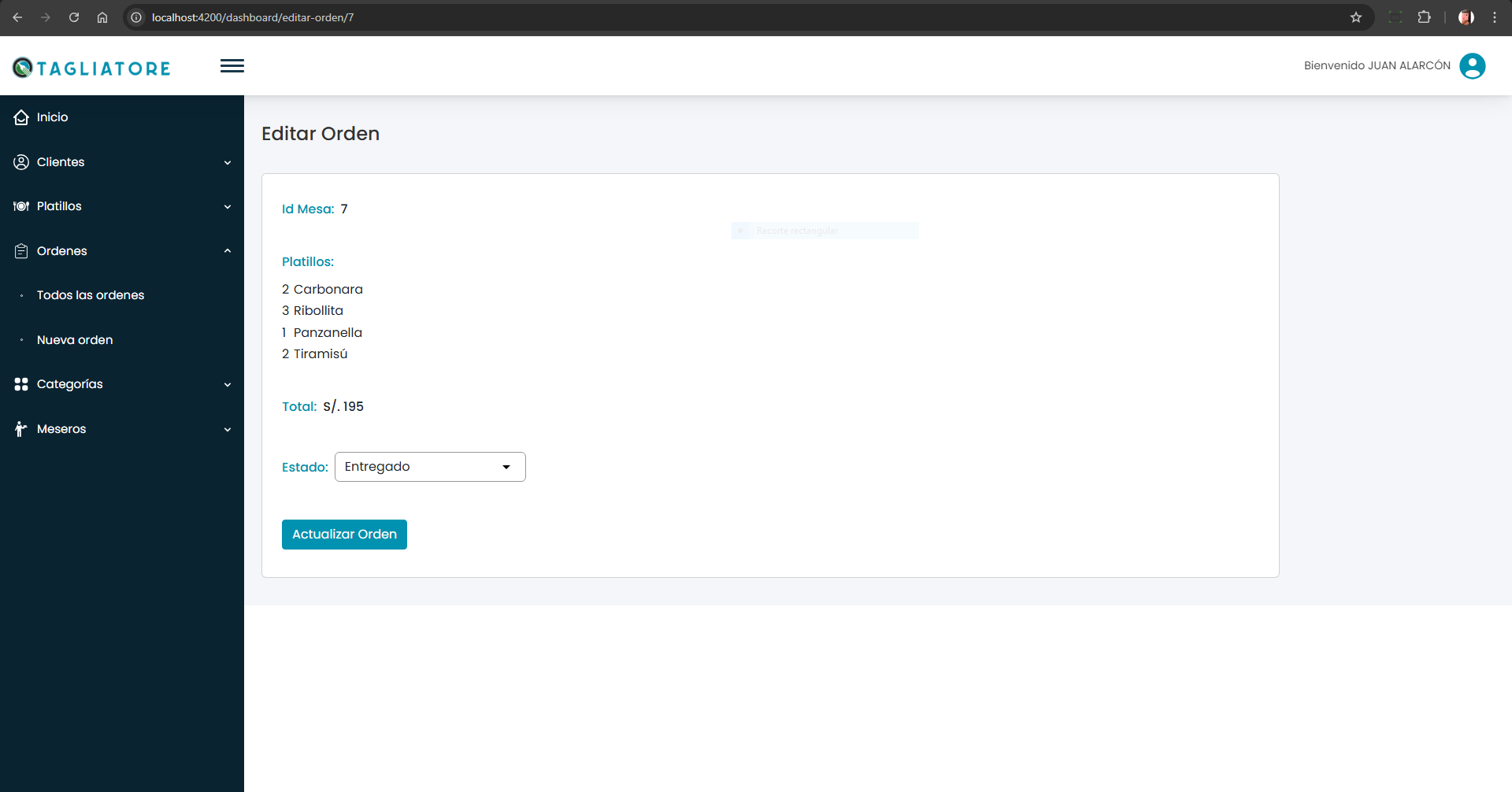
****

**ORDEN**

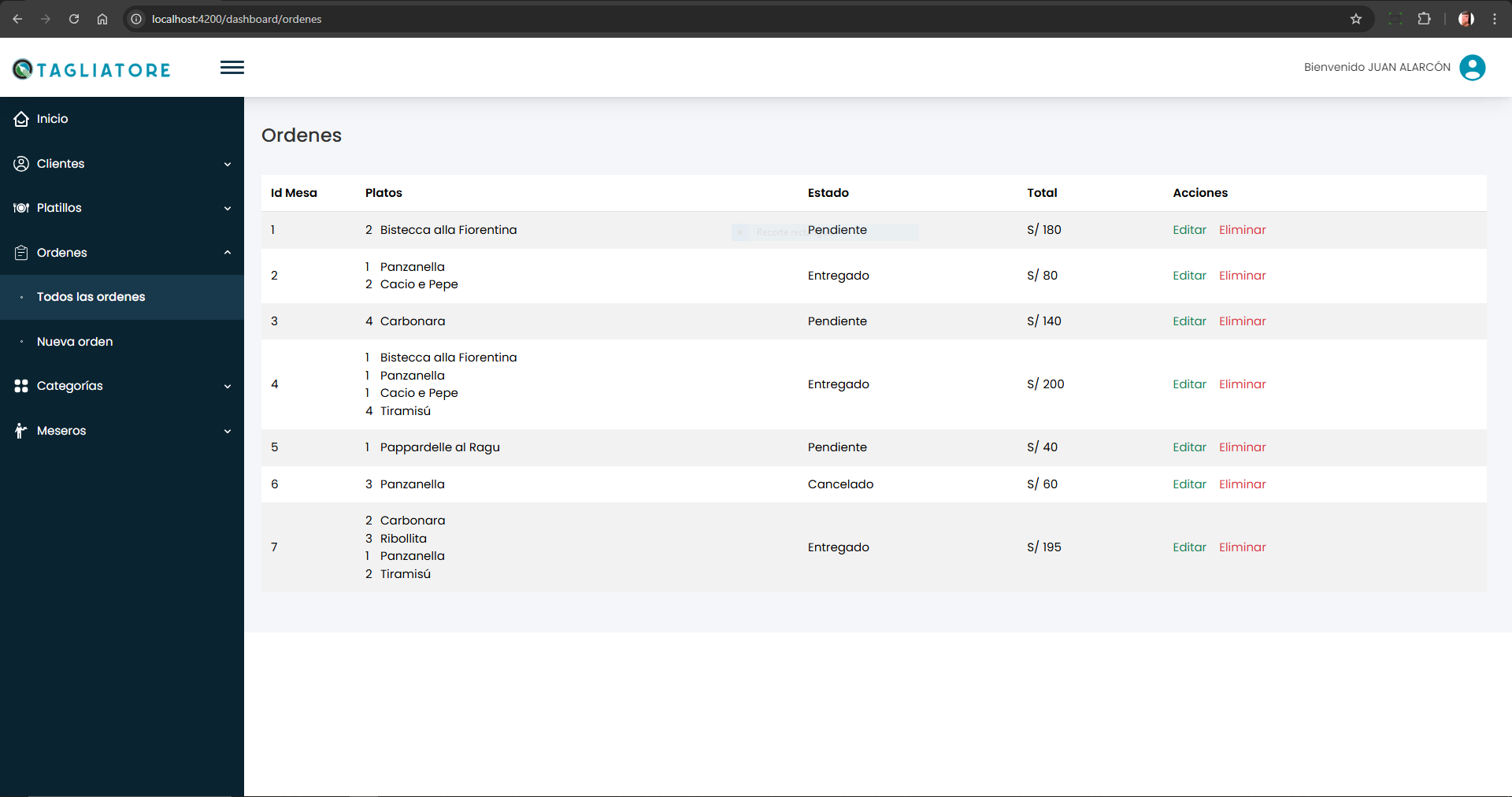
**Crear orden:** Se solicitará ingresar el id de la mesa, seleccionar el plato que desea ordenar permitiendo agregar cantidad y agregar más platos opcionales, eliminar los platos que no desea, al crear una orden estará como pendiente por defecto.

****

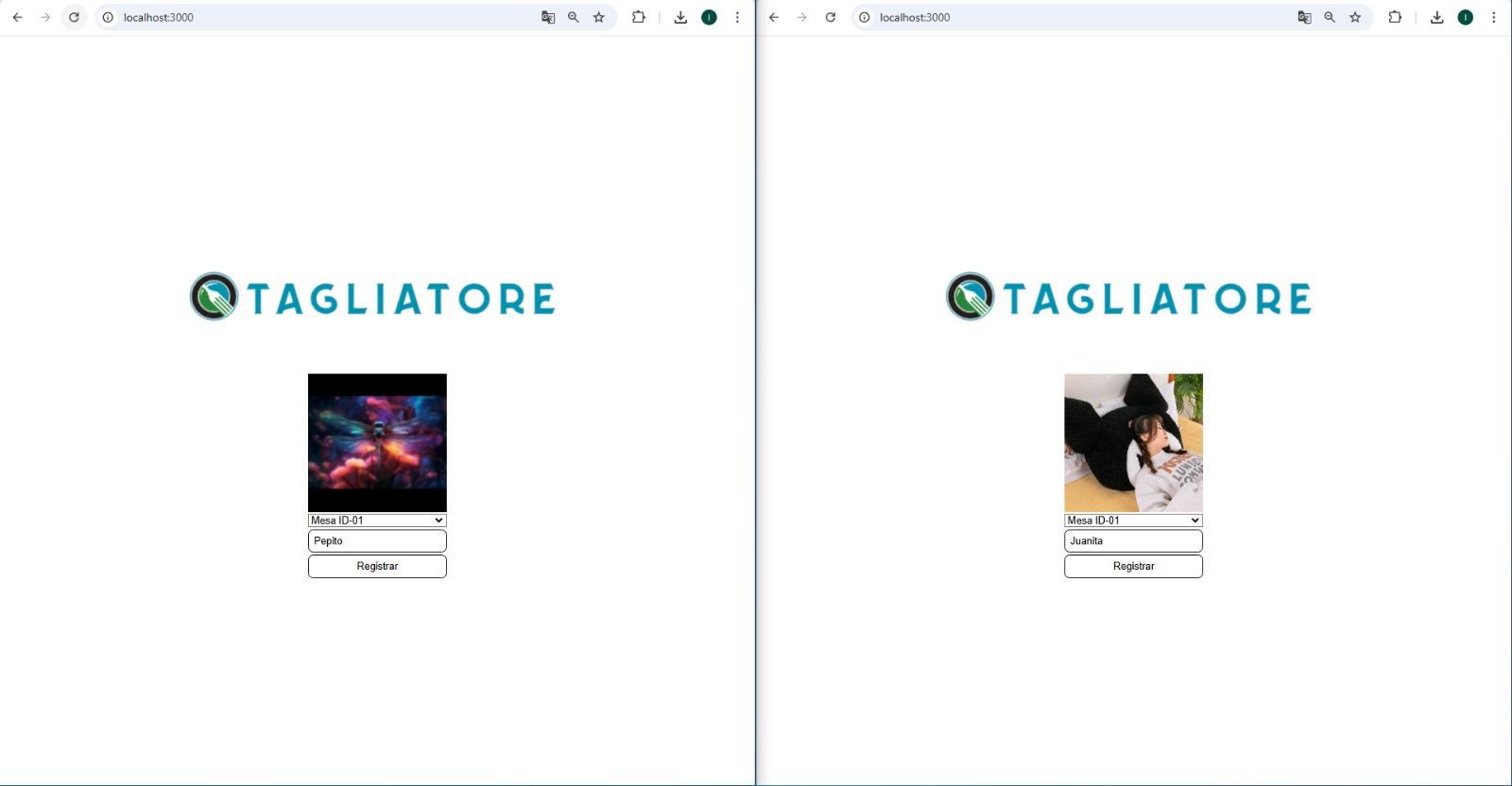
**Actualizar orden:** Nos permitirá cambiar el estado del pedido, por las siguientes opciones pendiente, cancelado y entregado. Visualizar la información del pedido.

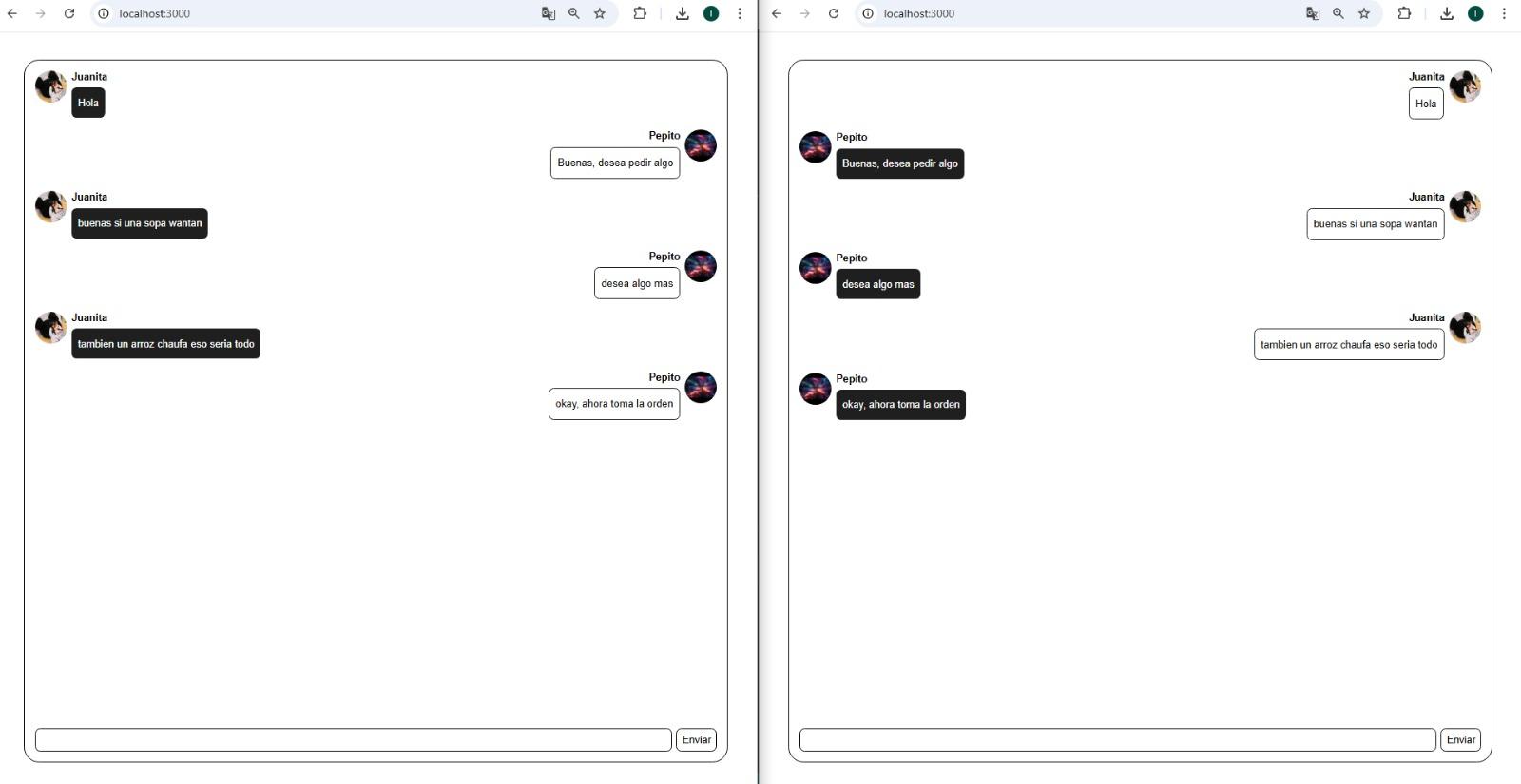


**Listar orden:** Trae toda la información del orden como la ID de la mesa, la lista de platos solicitados en la orden, el estado de la orden y el total de la orden, también tendrá la acciones de eliminar y editar. Al eliminar cambia el estado de la orden por cancelado automáticamente.

****

**CHAT EN TIEMPO REAL - PÚBLICO**

**Publico:** El cliente ingresa con su usuario, perfil y se une a la mesa en el que estará y el mesero le atenderá para tomar su orden. 



**Recomendaciones**

* Incluir una herramienta de análisis para evaluar el desempeño del sistema en tiempo real.
* Ampliar las funcionalidades del chat en tiempo real, permitiendo soporte automatizado.
* Migrar a una solución de despliegue en la nube, como AWS o Azure, para garantizar escalabilidad.
* Capacitar al personal del restaurante en el uso del sistema para maximizar su adopción.

**Conclusiones.**

El desarrollo del sistema para el restaurante “Tagliatore” representa una solución efectiva y moderna para abordar sus necesidades operativas. Gracias al uso de tecnologías como Angular, Node.js y MongoDB, se logró implementar una plataforma dinámica, segura y escalable.

El sistema no solo facilita la gestión diaria del restaurante, sino que también mejora la experiencia del cliente a través de herramientas como el chat en tiempo real. Este proyecto establece una base sólida para futuras mejoras e integraciones.

**Referencias Bibliográficas.**

* MongoDB Inc. (2024). *MongoDB Manual*. Recuperado de<https://www.mongodb.com/docs/manual/>
* Node.js Foundation. (2024). *Node.js Documentation*. Recuperado de <https://nodejs.org/en/docs/>
* Angular. (2024). *Angular Documentation*. Recuperado de <https://angular.io/docs>