**PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR**

**FACULTAD DE INGENIERÍA**

**ESCUELA DE SISTEMAS**

**TEMA:**

# APLICATIVO WEB PARA LA GESTIÓN DE LAS RESERVAS DE UN RESTAURANTE

**AUTOR:**

ERICK XAVIER REA YANEZ

TRABAJO PREVIA A LA OBTENCIÓN DEL TÍTULO DE INGENIERO DE SISTEMAS DE INFORMACIÓN

**QUITO DM, 2022**

## Dedicatoria

Desde el momento en el que empecé a estudiar, mis padres y hermanos han sido un gran apoyo para poder finalizar mi plan de estudios. No ha sido una tarea fácil acabar todos los semestres, ha habido dificultades, altas y bajas, pero el llegar a realizar este trabajo de titulación es la culminación de una meta planteada desde el inicio de mi carrera. Por esta razón, este trabajo está dedicado a mi familia por todo el apoyo que me ha brindado en este proceso.

**Agradecimiento**

Agradezco a mi familia por la confianza que depositaron en mí cuando empecé esta carrera y por estar presentes en cada dificultad que se presentó.

A la Pontificia Universidad Católica del Ecuador y sus profesores también les agradezco por todas las enseñanzas que me han impartido a mí y mis compañeros.

Por último, a mis amigos que he conocido durante la universidad, estar acompañado de ellos ha sido muy agradable. En lo personal, pienso que no se puede alcanzar una meta individualmente, estar con ellos ha sido un placer tanto en lo personal como profesional.

## Resumen

El avance tecnológico ha permitido que distintos campos puedan resolver diferentes problemáticas, en este caso se toma en cuenta que existen procesos al momento de recibir comensales en un restaurante que pueden ser optimizados. El realizar reservas es un punto a favor de un restaurante, debido a que los comensales estarían seguros de que serán atendidos.

La pandemia de los años recientes, provocada por el virus SARS COVID-19, obligó a los diferentes puntos comerciales a implementar alternativas tecnológicas para poder seguir con su funcionamiento. En este caso, crear una herramienta que permita controlar el aforo es de importancia, ya que el cliente se sentiría seguro del lugar que va a visitar una vez que haya realizado su reserva.

El presente proyecto tiene como objetivo crear una aplicación web que permita gestionar las reservas en un restaurante de tal forma que el aforo sea controlado y que, además, los clientes estén seguros de que van a ser atendidos.

**Tabla de Contenido**

## Índice de Figuras

## Índice de Tablas

# Parte I

## I. Justificación

En el año 2020, el 29 de febrero se oficializó el primer caso de contagio de COVID-19 en la República del Ecuador (World Bank Group, 2021). Desde esta fecha se incrementó la demanda de herramientas digitales que permitiesen continuar las actividades económicas, educativas y sociales de forma remota (Zaballos et al., 2020).

La industria de restaurantes desde el año en el que inicio la pandemia se adaptó usando diferentes aplicaciones para entregar sus productos. Aunque, el porcentaje de aforo de sus locales era controlado según las decisiones de cada país o ciudad. En Ecuador durante la fecha 3 de septiembre de 2020, los restaurantes tuvieron un aforo permitido del 30% (González, 2020). El aforo incrementó de manera gradual. En la fecha 18 de marzo de 2022, Ecuador volvió a tener 100% de aforo en sus actividades (Primicias, 2022). El presente proyecto plantea la creación de un aplicativo que permita el control de aforo mediante reservas de lugares dentro de un restaurante.

## II. Definición del problema

Los clientes al visitar un restaurante no conocen si tendrán asientos disponibles o no. Por lo que un cliente puede preferir quedarse en casa en lugar de arriesgarse a que el servicio se le sea negado por falta de espacio (Alexandrov, 2012).

En el estudio realizado por Alexandrov (2012), se encontró que las reservas abordan el problema mencionado al garantizar asientos para los comensales, lo que genera más demanda en un mercado inactivo. También encontró que las reservas son valiosas para el restaurante cuando no hay una alta demanda de servicio, pero que estas se vuelven costosas cuando la demanda es

alta, es decir que pueden tener un efecto negativo para los clientes y el restaurante. Los clientes no valoran las reservas cuando la actividad del restaurante es baja, pero las valorarían más, por ejemplo, en noches con una alta afluencia de clientes.

En función del problema mencionado se plantea la siguiente pregunta principal de investigación:

- ¿Qué solución tecnológica es viable para realizar reservas?

Y las siguientes preguntas secundarias:

- ¿Qué requiere el cliente para hacer una reserva?
- ¿Qué necesita el restaurante para poder manejar las solicitudes de reservas que reciba?
- ¿Existe un límite de comensales dentro de las instalaciones de un restaurante?

## III. Objetivo General

Desarrollar un aplicativo web para gestión de reservas de un restaurante.

## IV. Objetivos Específicos

- Determinar los requerimientos del aplicativo web para realizar reservas.
- Diseñar la base de datos para almacenar la información del restaurante, clientes y reservas.
- Codificar el aplicativo web.

## V. Alcance

El desarrollo del presente trabajo tendrá como alcance el gestionar las reservas de un restaurante. Se plantea la creación de un aplicativo escalable que pueda manejar las reservas de

múltiples restaurantes a la vez, pero dentro de los límites de este desarrollo se manejará solamente un restaurante, con múltiples sucursales y múltiples mesas. Será posible la gestión del restaurante de modo que pueda editarse la información de este. Además, será posible agregar múltiples sucursales junto con la cantidad de mesas que este local permita. Cada sucursal deberá tener ingresado un plano de las instalaciones para poder ubicar correctamente las mesas. Los lugares de cada local podrán ser unificados en caso de necesitar un mayor número de asientos.

El aplicativo será capaz de manejar múltiples usuarios y múltiples roles. Para el desarrollo se manejará dos roles principales. El primer rol será el "administrador", este usuario se encargará de gestionar información privada del restaurante. Asimismo, el usuario mencionado podrá gestionar los usuarios registrados dentro del sistema. El segundo rol será el "cliente", este tipo de usuario podrá hacer reservas de los lugares registrados en el sistema.

La función de reservaciones podrá ser realizado por los usuarios "administrador" y "cliente". El primer usuario podrá aceptar, crear, cancelar o rechazar todas las reservaciones existentes. Mientras que el segundo usuario solamente podrá crear y cancelar sus propias reservaciones.

## VI. Metodología

El presente trabajo tendrá como primera metodología a la investigación aplicada, ya que se buscar aplicar los conocimientos adquiridos. La primera instancia del trabajo de titulación será recopilar información correspondiente para desarrollar el aplicativo.

Para el desarrollo se utilizará la metodología Scrum el cual sirve para gestionar un proyecto y reducir su complejidad en función de las necesidades de los clientes; todo el equipo de Scrum

coopera para cumplir los requerimientos del proyecto y obtener resultados de manera incremental (Francia, 2017, párr. 1).

Esta metodología permite crear un producto mediante diferentes fases, en las cuales se obtiene un incremento de producto al final de cada una de ellas.

# Capítulo I: Fundamentos Teóricos

El desarrollo del presente proyecto requiere conocer diferentes conceptos para poder desarrollar una aplicación web.

## 1.1. Lenguajes de programación

Un lenguaje de programación es una serie de instrucciones definidas por una sintaxis, la cual establece e indica reglas de escritura de este, y por la semántica de los tipos de datos, instrucciones, definiciones y otros elementos para crear un programa informático (Mathieu, 2014, p. 5).

### *1.1.1. PHP*

Arias (2013) define a este lenguaje como:

> PHP es un acrónimo recursivo para "PHP: Hypertext Preprocessor", originalmente Personal Home Page, es un lenguaje interpretado libre, usado originalmente solamente para el desarrollo de aplicaciones presentes y que actuaran en el lado del servidor, capaces de generar contenido dinámico en la World Wide Web. (p. 6)

### *1.1.2 JavaScript*

Es un lenguaje de programación usado para crear páginas web dinámicas (Pérez, 2019, p. 5). Este tipo de páginas como su nombre lo dice pueden cambiar con las diferentes interacciones que un usuario tenga con la página. Por ejemplo, hacer que el texto cambie o se oculte al cargar un nuevo estado, animaciones al momento de colocar el apuntador sobre un elemento, activar y desactivar alertas al hacer clic sobre un botón, entre otros (Pérez, 2019, p. 5).

Es un lenguaje de programación interpretado, esto significa que su código no es compilado para ser ejecutado.

## 1.2. Desarrollo Web

Dentro del desarrollo web predominan las aplicaciones web, el cual es un software que reside dentro de un servidor, al cual un usuario puede acceder a través una red de internet local o externa mediante el navegador web (Zofío Jiménez, 2013).

### *1.2.1. HTML*

HyperText Markup Language (HTML) o lenguaje de etiquetas de hipertexto, es un lenguaje de marcas que delimitan partes de un objeto, que sirven para describir el contenido y la estructura de las páginas web, son interpretadas y renderizadas a través de navegadores de internet (Zofío Jiménez, 2013).

#### 1.2.1.1. Etiquetas.

Son las marcas que definen los elementos de HTML. Los signos “menor que” (<) y “mayor que” (>) son usados para delimitar las etiquetas.

##### *1.2.1.1.1. Etiquetas Cerradas.*

Son etiquetas con un principio y un final. Por ejemplo,  Etiqueta cerrada .

##### *1.2.1.1.2. Etiquetas Abiertas*

Son etiquetas que cuenta solo de una etiqueta para simbolizar el principio y el final de esta. Por ejemplo, <hr> o <br>.

#### 1.2.1.2. Atributos

Son características que pueden ser asignadas de manera opcional a las etiquetas. Se colocan dentro de la etiqueta de inicio. Se componen de una palabra reservada y un valor. Por ejemplo, <input text=”number” id=”etiqueta”>

#### 1.2.1.3. Comentarios

Son anotaciones y aclaraciones colocadas en el código fuente con el fin de otorgar información. Un comentario se realiza entre los símbolos “<!--” y “ -->”.

### *1.2.2. CSS*

Cascading Style Sheets (CSS) o hojas de estilo en cascada, son un instrumento usado para dar estilo a los elementos HTML. Este mecanismo fue creado para separar el contenido y la estructura de un documento HTML de su estilo y formato de presentación (Zofío Jiménez, 2013)

## 1.3. Frameworks

Los *frameworks* o marcos de trabajo son un diseño reusable de todo o una parte de un sistema que está representado por clases abstractas y pueden instanciarse de una o más formas (Johnson, 1997, p. 39).

Un *framework* busca alivianar la sobrecarga asociada con el desarrollo web. Existen diferentes tipos de *frameworks*, los cuales proporcionan bibliotecas para el acceso a bases de datos, plantillas de sistemas, administración de sesiones, estilos predefinidos para las interfaces, entre otros. Los marcos de trabajo son interfaces de desarrollo, un conjunto de herramientas de software para crear sitios web, aplicaciones y servicios dinámicos (Pater, 2015). Estos, a su vez, son herramientas que sirven para lograr el desarrollo de una aplicación web y que el desarrollador pueda enfocarse en la lógica del sistema.

### *1.3.1. Laravel*

Laravel es un *framework* de aplicaciones web basado en PHP que provee una estructura y punto de inicio para crear una aplicación. El propósito de esta herramienta es que el desarrollador pueda enfocarse en la lógica del negocio, por lo que brinda características como inyección de dependencias, una capa de abstracción de base de datos, colas, *scheduled jobs*, pruebas unitarias, entre otras (Otwell, s. f.).

Se puede usar a Laravel como un *framework full stack* lo que significa que puede manejar las peticiones de enrutamiento y renderizar interfaces mediante plantillas “Blade”. También es posible su integración con otros *frameworks* de front-end como Livewire, Inertia.js, Vue, entre otros.

### *1.3.2. Livewire*

Es un *framework full stack* construido para Laravel, permite la creación de interfaces dinámicas de manera simple utilizando las funciones de Laravel. Esta herramienta funciona realizando peticiones AJAX, Asynchronous JavaScript And XML o JavaScript y XML ascríncronos., las cuales solicitan una acción al servidor y, al ser realizada, el cliente recibe una respuesta y se encarga de mostrarla.

Livewire utiliza una función antirrebote inteligente que limita la cantidad de solicitudes simultáneas enviadas al servidor. Esto ayudará a que la funcionalidad en tiempo real sea más eficiente (Lea, 2020).

### *1.3.3. Alpine.js*

Este es un *framework* robusto y minimalista construido para vincular comportamiento de JavaScript (JS) a las etiquetas HTML (Borgen, 2020). Caleb Porzio, creador de Alpine.js, declara que el objetivo de este marco de trabajo es llenar el vacío entre vanilla JS o jQuery y frameworks robustos como Vue y React.

Este *framework* es ideal cuando se necesita una pequeña cantidad de código JS sin tener que instalar librerías mediante NPM, sistemas de gestión de paquetes.

### *1.3.4. Tailwind CSS*

Tailwind es un framework de "utilidad primero", este sigue un enfoque basado en restricciones, donde las clases de utilidad representan definiciones de estilo CSS elementales (Kimm, 2021). El desarrollador se encarga de definir el diseño de la interfaz utilizando múltiples clases de utilidad atómica en las etiquetas HTML. Es decir, Tailwind CSS permite diseño de interfaces de manera personalizada y consistente mediante la introducción de una organización y configuración de utilidades que promueven el diseño sistemático (Kimm, 2021)

## 1.4. Metodologías de Desarrollo de Software

El desarrollar software de calidad requiere atender necesidades y requisitos que los usuarios demanden, y que, además, sea fácil de usar. Por tal razón, el cumplir parámetros de calidad vuelve a la creación de software compleja y, debido a ello, desde que se inició la fabricación de computadoras y hubo la necesidad de programas que funcionaran, surgieron también las metodologías de desarrollo de software (MDS) (Maida, 2015).

### *1.4.1. Metodologías Tradicionales*

El trabajo de Velásquez et al. (2019) definió a las metodologías tradicionales como:

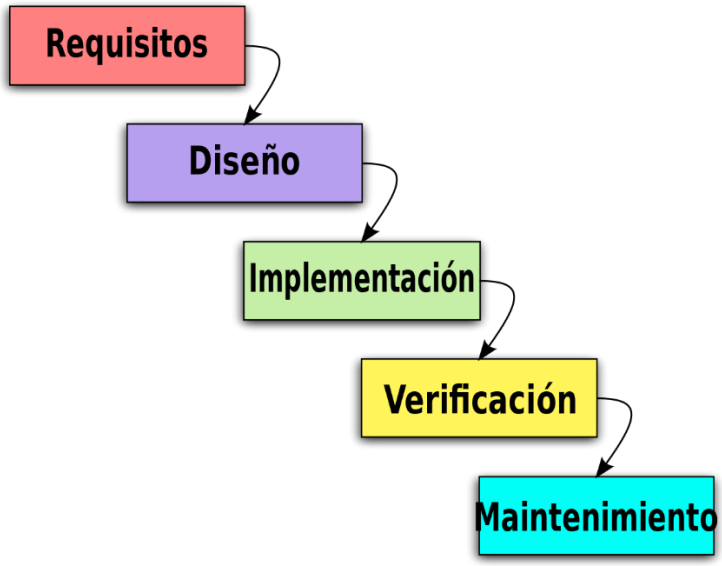
Estas metodologías imponen una disciplina de trabajo sobre el proceso de desarrollo del software, buscando conseguir un software más eficiente y predecible; este enfoque es considerado tradicional por ser el primero que se empleó para desarrollar software (Paz, 2016). Exige que se preste gran atención a la planificación total de todo el trabajo a realizar y, una vez que está todo detallado, se inicia el desarrollo del producto (Gómez et al., 2010).

#### 1.4.1.1 Modelo en Cascada

Este modelo fue desarrollado en 1970 y es llamado "modelo en cascada" debido a que su presentación gráfica es similar a una cascada (Kumar & Bhatia, 2014). Dicha representación se puede observar en la Figura 1.

**Figura 1**

*Modelo en Cascada*




*Nota*. Adaptado de *Modelo Cascada* [Imagen], por Paul Smith, 2012, Wikipedia (http://en.wikipedia.org/wiki/File:Waterfall_model_%281%29.svg). CC BY 3.0.

Este modelo sirvió de base para otros. Se centra en reforzar el “definir antes que codificar” (Sommerville, 2004, como se citó en Kumar & Bhatia, 201)

#### 1.4.1.2. Prototipado

Armenta et al. (2018) definen al modelo de prototipos como:

> Pertenece a la rama de los modelos evolutivos tales como el modelo incremental, en este modelo se hace el uso de prototipos los cuales represe tan partes esenciales del software, sin embargo, a diferencia del modelo incremental los prototipos no son funcionales y en su mayoría únicamente corresponden a representaciones del diseño visible para el usuario y no del diseño interno tales como algoritmos y manejo de la Información, a al desarrollador una mejor comprensión de lo que hay que hacer (Krar, 2009). (p. 9)

La principal característica de este modelo es la retroalimentación, permite crear una muestra de la solución, comprobar si será funcional tomando en consideración la opinión del cliente, y se hacen las modificaciones necesarias hasta llegar a un modelo final que solvente las necesidades del cliente (Armienta et al., 2018, p. 9)

**Figura 2**

*Modelo de Prototipos*


*Nota*. Ciclo del modelo de prototipos.

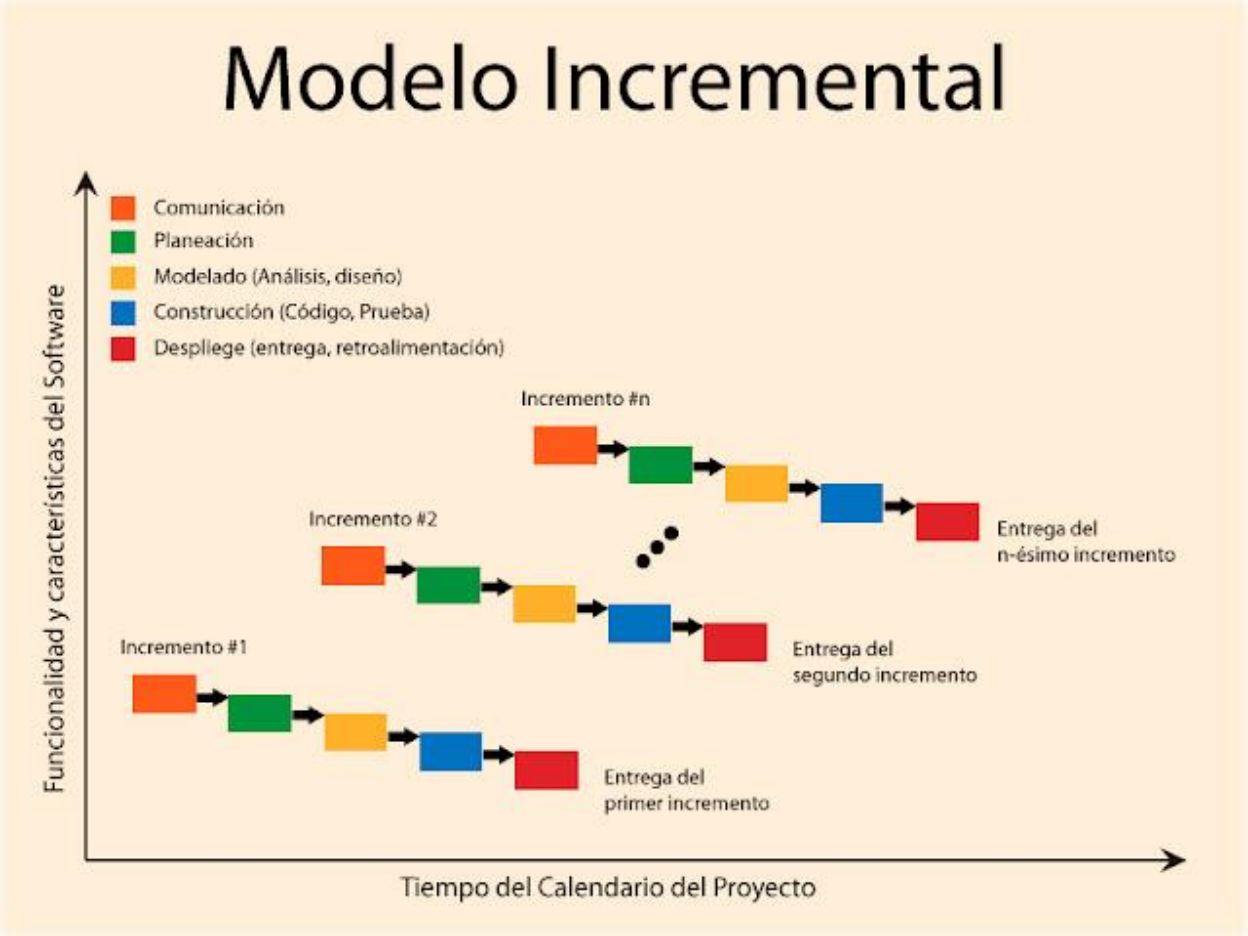
#### 1.4.1.3. Incremental

En el trabajo de Gamboa (2018) explica que el modelo incremental aparece de la siguiente forma:

> Harlan Mills en el año 1980. Se basa en el desarrollo a partir del incremento de la funcionabilidad del programa, se puede considerar un precursor de las modernas metodologías iterativas. El primer incremento es a menudo un desarrollo esencial, apenas con los requisitos básicos, cada incremento representa una entrega escalable. (p. 27)

**Figura 3**

*Modelo Incremental*




*Nota*. Adaptado de *Modelo Incremental* [Imagen], por Marlady Ortiz, 2012, Blogger (https://isw-udistrital.blogspot.com/2012/09/ingenieria-de-software-i.html).

#### 1.4.1.4. Espiral

"El modelo espiral en el desarrollo del software es un modelo meta del ciclo de vida del software donde el esfuerzo del desarrollo es iterativo, tan pronto culmina un esfuerzo del desarrollo comienza otro" (Galo, 2011, p. 2). Cuenta con cuatro pasos principales:

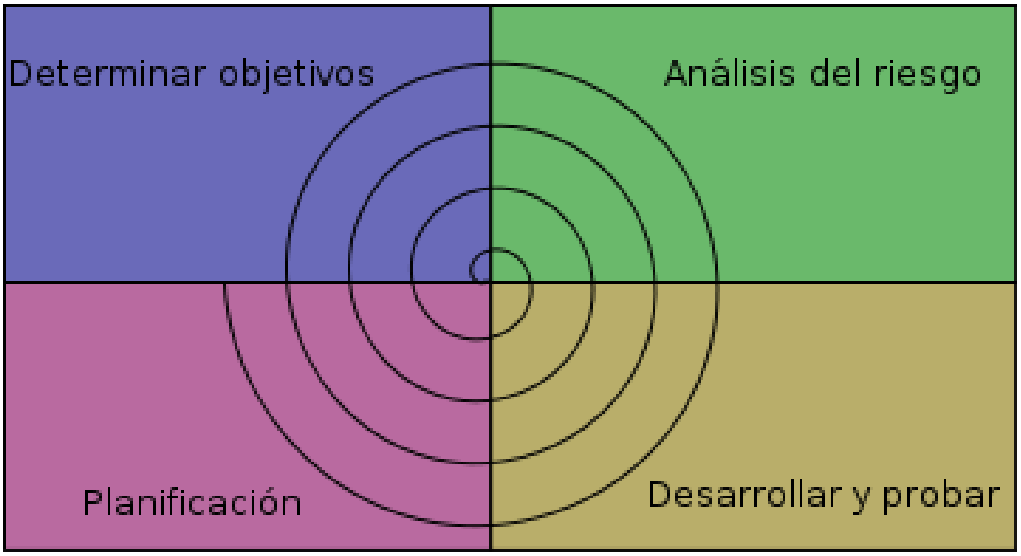
- Determinar o fijar los objetivos
- Análisis del riesgo
- Desarrollar, verificar y validar

- Planificar

A diferencia de los otros modelos, el modelo en espiral evalúa los riesgos. Un riesgo es aquello que pueda afectar a un proyecto durante su desarrollo.

**Figura 4**.

*Modelo en Espiral*




*Nota.* Fases del modelo en espiral.

### *1.4.2. Metodologías Ágiles*

El término “ágil” es acuñado en el año 2001 junto con la “Alianza Ágil”, organización creada para promover el desarrollo ágil de software y ayudar a las empresas a implementar estas metodologías (Canós et al., 2003, p. 2). A diferencia de las metodologías tradicionales, rígidas y limitadas por la documentación, el objetivo de estas metodologías es permitir que un equipo pueda desarrollar software con la capacidad de adaptarse a los cambios que nazcan durante el desarrollo (Canós et al., 2003, p. 2).

Las metodologías ágiles deberían seguir estas reglas: primero, el individuo y las interacciones del equipo de desarrollo sobre el proceso y las herramientas; segundo, desarrollar software funcional más que conseguir una buena documentación; tercero, colaboración con el cliente más que la negociación del contrato; y cuarto, responder a los cambios más que seguir estrictamente un plan (Canós et al., 2003, pp. 2-3).

#### 1.4.2.1. Extreme Programming (XP)

Joskowicz (2008) explica a *extreme programming* como:

> La metodología XP define cuatro variables para cualquier proyecto de software: costo, tiempo, calidad y alcance. Además, se especifica que, de estas cuatro variables, sólo tres de ellas podrán ser fijadas arbitrariamente por actores externos al grupo de desarrolladores (clientes y jefes de proyecto). El valor de la variable restante podrá ser establecido por el equipo de desarrollo, en función de los valores de las otras tres. Este mecanismo indica que, por ejemplo, si el cliente establece el alcance y la calidad, y el jefe de proyecto el precio, el grupo de desarrollo tendrá libertad para determinar el tiempo que durará el proyecto. (p. 7)

#### 1.4.2.2. Scrum

Scrum es un *framework* iterativo e incremental usado en diferentes proyectos para realizar un producto. Este se realiza en ciclos llamados *sprints*, duran entre una a cuatro semanas y la finalización de uno empieza otro. Un *sprint* no puede cambiar su fecha de finalización una vez se ha definido. Cada uno de estos ciclos cuentan con diferentes fases. Al inicio el equipo que conforma parte del Scrum debe definir que requerimientos se pueden alcanzar durante el *sprint*. Durante la duración del *sprint* el equipo debe reunirse cada día para informar de los avances

logrados con la finalidad de tomar medidas en caso de que no se esté cumpliendo lo planeado. Una vez alcanzada la fecha límite del *sprint* se debe presentar los avances a las partes interesadas del proyecto. Recogidas las observaciones de estas partes se las debe integrar a ciclos posteriores. La metodología busca desarrollar productos que en cada incremento puedan considerarse potenciales a entregar, aunque no estén terminados (Deemer et al., 2009, p. 5).

# Capítulo II: Requerimientos y Análisis de las Posibles Soluciones

## 2.1. Especificación de Requerimientos de Software (IEEE830)

Al seguir el estándar IEEE930 para la especificación de requisitos del sistema (ERS) se seguirá mediante la estructura presentada a continuación.

### *2.1.1. Introducción*

El siguiente apartado presentará los requerimientos del aplicativo web que será desarrollado para gestionar las reservas de un restaurante.

**2.1.1.1. Propósito.** Se pretende conocer los requerimientos necesarios para que el aplicativo cuente con las características necesarias para gestionar las reservas.

Este documento va dirigido a todo aquel que desee conocer los requerimientos que fueron tomados en cuenta para crear el sistema.

**2.1.1.2. Ámbito del Sistema.** El aplicativo web será llamado “RESOW” de manera provisional, dicho nombre resulta de la unión de las palabras *reserve* (reservar) y *now* (ahora).

El aplicativo web se encargará de gestionar las reservas de restaurantes, los clientes podrán observar los lugares disponibles junto con la capacidad del lugar. Los administradores del restaurante podrán administrar las reservas que los comensales realicen. El beneficio principal del sistema será que los clientes podrán asegurar un lugar al restaurante que desean visitar y la plataforma estará disponible 24/7 para quien lo desee. Las reservaciones se realizarán, al menos, con un día de antelación. La primera versión del aplicativo podrá manejar solamente un sistema, pero, se plantea que este sea escalable y que pueda mejorarse hasta el punto de manejar múltiples restaurantes.

**2.1.1.3. Definiciones, Acrónimos y Abreviaturas.** Estos son definidos a lo largo del documento de titulación.

**2.1.1.4. Referencias.** Estas se colocarán en el apartado referencias de documento de titulación.

**2.1.1.5. Visión general del documento.** En este apartado de ERS se determinará los actores del sistema, las acciones que podrán realizar los actores, las características del sistema y las características futuras del sistema.

### *2.1.2. Descripción General*

En esta sección se describirá el contexto del producto, lo que lo afecte y sus requisitos (Agut, 2001).

**2.1.2.1. Perspectiva del Producto.** El aplicativo será capaz de adaptarse al tamaño del restaurante que necesite implementar reservas. Se considerará el número de sucursales en las que se divida el restaurante, a su vez, se mantendrá un control del aforo de cada local. El sistema se mantendrá disponible en un dominio de internet. El aplicativo deberá beneficiar a ambas partes, al cliente y al personal que conforme el restaurante. La reservación debe contar con doble confirmación por parte del sistema y manual para no perjudicar al restaurante. En caso de no cumplir con la reserva, el restaurante podrá colocar una penalización por incurrir en dicha falta.

**2.1.2.2. Funciones del Producto.** Además de manejar las reservas de lugares disponibles y ocupados, el sistema contará con funciones adicionales como:

- Gestión de la información del restaurante.
- Gestión de las sucursales del restaurante.

- Gestión de las mesas de cada restaurante.
- Gestión de usuarios.
- Gestión de reservas por local.
- Gestión de reservaciones por parte del cliente.
- Gestión de reservas por parte del restaurante.
- Gestión de amonestaciones por no presentarse a una reserva confirmada.

**2.1.2.3. Características de los usuarios.** El sistema distinguirá dos tipos de usuario, los cuales son administrador y cliente. Los usuarios administradores pertenecerán al restaurante. Este usuario deberá conocer la estructura de las sucursales o solamente de la sucursal a la que pertenezca. Su función será la de manejar información que el restaurante mostrará hacia el cliente y además confirmará una reserva que esté pendiente.

El usuario cliente, pertenecerá al público general y podrá visualizar los locales y lugares disponibles para que pueda apartar la localización que solicite. El aplicativo deberá ser intuitivo al momento de usarse, ya que el público general puede tener diferentes experiencias con la tecnología que no permitan que el sistema sea usado correctamente.

**2.1.2.4. Restricciones.** Los usuarios no podrán manipular código fuente del sistema, debido a que este contará con restricciones de seguridad propias de la programación orientada a objetos. El sistema se colocará en un servidor que solo podrá ser accedido mediante credenciales que no forman parte del aplicativo. Un usuario con rol asignado no podrá observar ni manipular acciones que no sean inherentes a su rol.

**2.1.2.5. Suposiciones y Dependencias.** El aplicativo web será instalado dentro de una plataforma que soporte el alojamiento y procesamiento de información de este. El renderizado

del aplicativo se realizará por parte del navegador web del usuario cliente. Para alojar la información del sistema, se utilizará un motor de bases de datos relacional.

**2.1.2.6. Requisitos Futuros.** El sistema debe contar con las capacidades necesarias para poder ser escalable de tal forma que permita el aumento de restaurantes que puedan colocar información de estos para que puedan realizar reservas.

### *2.1.3. Requisitos Específicos*

Esta sección contendrá todos los requerimientos de manera detallada para que el desarrollo sea claro para quien lo construirá (Agut, 2001).

**2.1.3.1. Interfaces Externas.** Agut (2001) define a esta sección como aquella que define los requisitos que afecten a la interfaz de usuario e interfaz con otros sistemas y/o interfaces de comunicación.

***2.1.3.1.1. REQ01: Interfaz Responsive.*** Las interfaces de usuarios serán responsive, es decir, podrán ser visualizadas en dispositivos de diferente tamaño como monitores, teléfonos inteligentes, entre otros dispositivos.

**2.1.3.2. Funciones.** Esta sección abarca las acciones o funciones que deberá llevar a cabo el software a desarrollar.

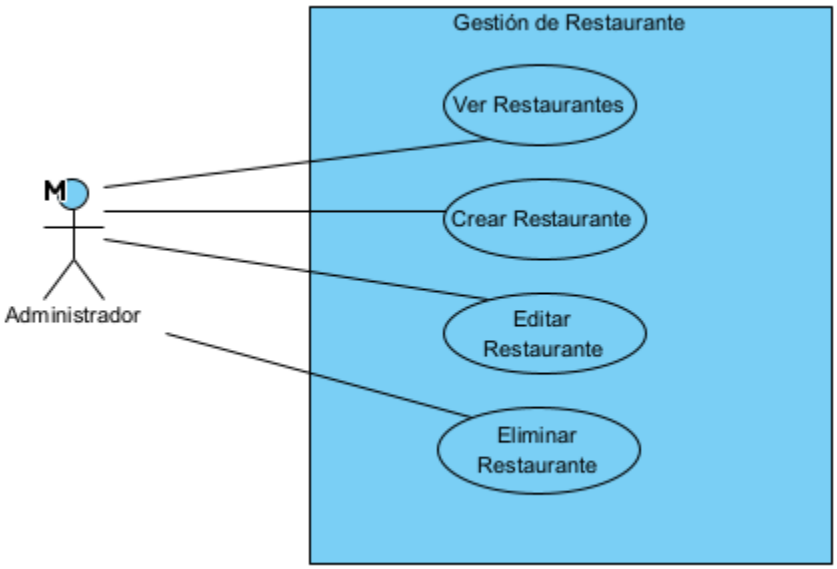
***2.1.3.2.1. REQ02: Versión Completa.*** El sistema deberá tener una versión final estable que pueda realizar reservas sin errores.

***2.1.3.2.2. REQ03: Documentación.*** La documentación del código se realizará dentro del mismo código del programa.

***2.1.3.2.3. REQ04: Gestión de Restaurante.*** Se debe poder administrar la información de un restaurante de manera que se pueda crear, ver, actualizar y eliminar información del restaurante.

**Figura 5**

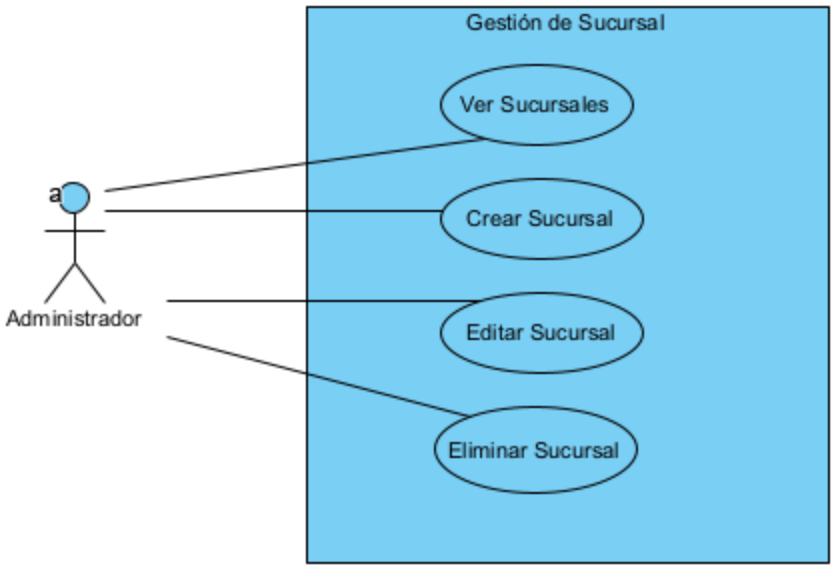
*Caso de uso del REQ04*




***2.1.3.2.4. REQ05: Gestión de Sucursales.*** Se debe poder administrar la información de los locales/sucursales de un restaurante de manera que se pueda crear, ver, actualizar y eliminar información de estos.

**Figura 6**

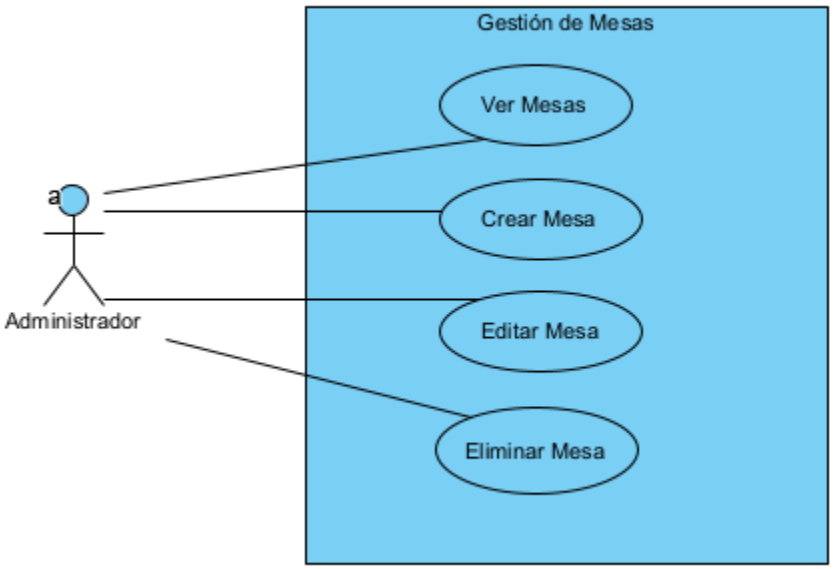
*Caso de uso del REQ05*




***2.1.3.2.5. REQ06: Gestión de Mesas.*** Se debe poder administrar los asientos/mesas de cada sucursal de manera que se pueda crear, ver, actualizar y eliminar información.

**Figura 7**

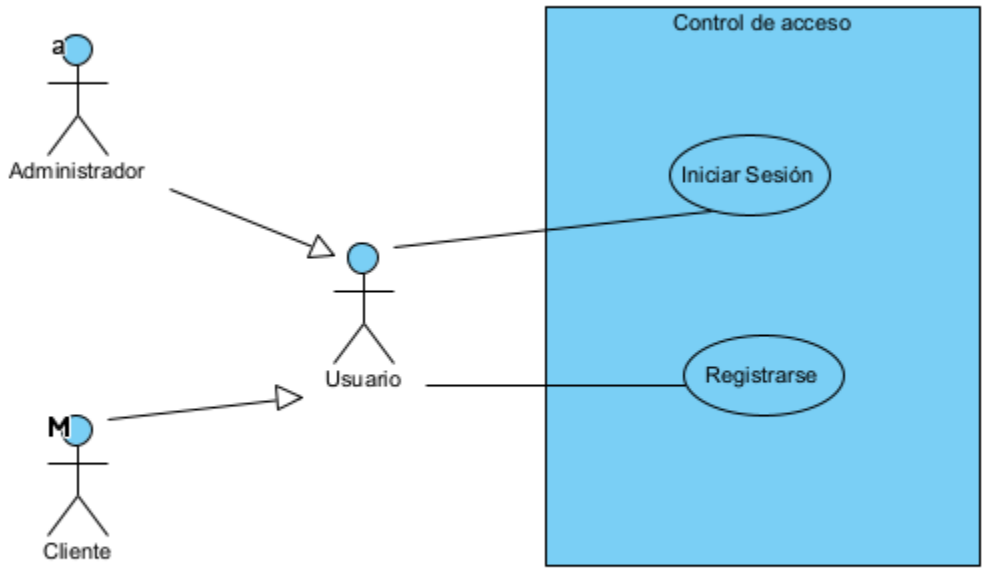
*Caso de uso del REQ06*



***2.1.3.2.6. REQ07: Control de Acceso.*** Los usuarios deberán contar con una interfaz de acceso a la que puedan ingresar mediante usuario y contraseña.

**Figura 8**

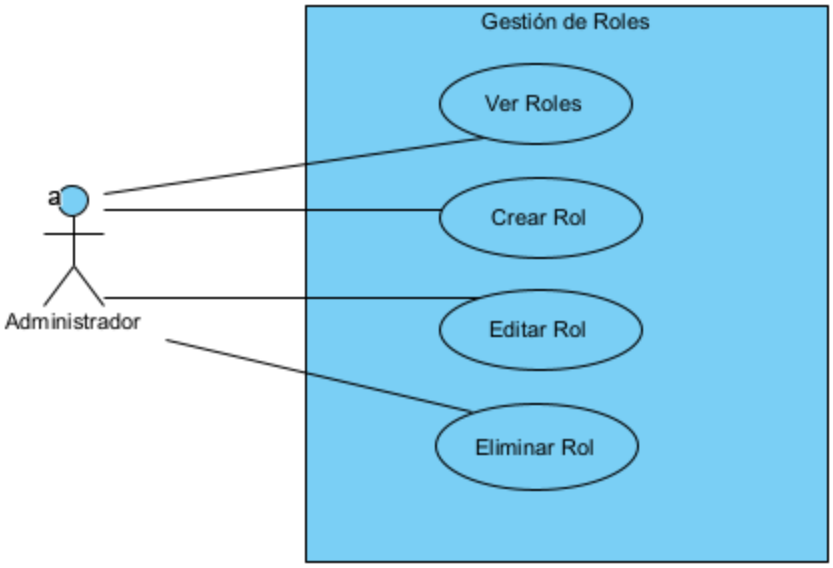
*Caso de uso del REQ07*




***2.1.3.2.7. REQ08: Gestión de Roles.*** Los usuarios tendrán roles que les permitirán realizar acciones restringidas al tipo de rol que se les haya asignado. La información podrá ser creada, observada, actualizada o eliminada. El sistema deberá contar con dos usuarios principales administradores y clientes. Además, existirá un usuario super administrador que será creado junto con el sistema.

**Figura 9**

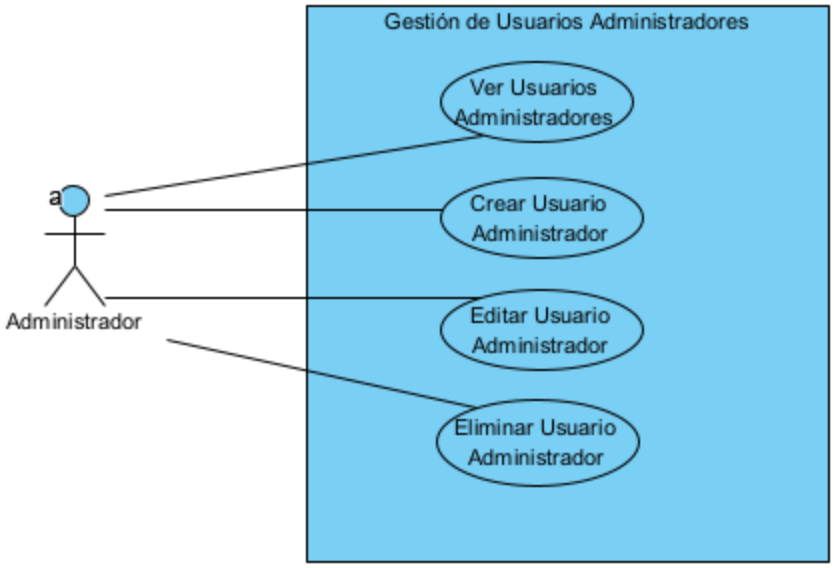
*Caso de uso del REQ08*




***2.1.3.2.8. REQ09: Gestión de Usuarios Administradores.*** Este tipo de usuario podrá ser creado mediante el usuario super administrador o administrador. La información de este tipo de usuario podrá ser creada, vista, actualizada y eliminada.

**Figura 10**

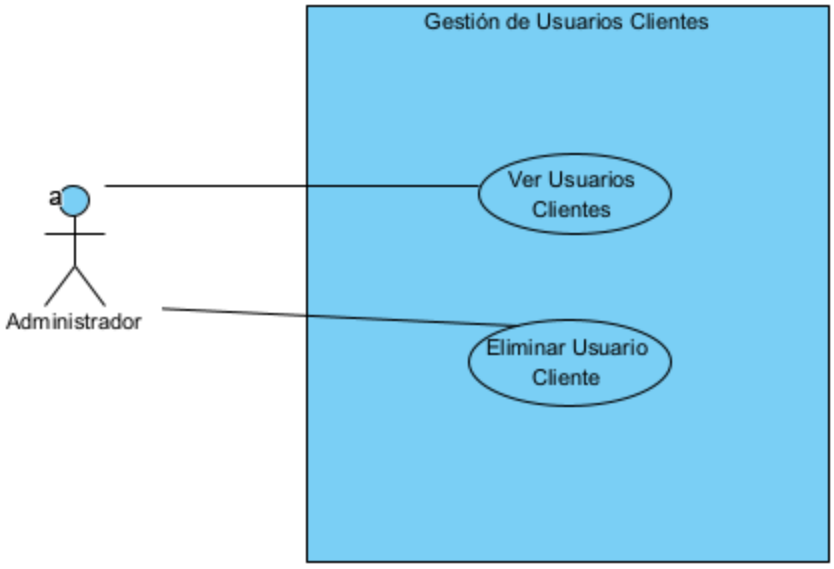
*Caso de uso del REQ09*




***2.1.3.2.9. REQ10: Gestión de Usuarios Clientes.*** Este tipo de usuario cuenta con la opción de registro. A su vez, podrá actualizar o desactivar su cuenta. El usuario administrador o superadministrador solamente podrá crear, ver o eliminar este tipo de usuario.

**Figura 11**

*Caso de uso del REQ10*



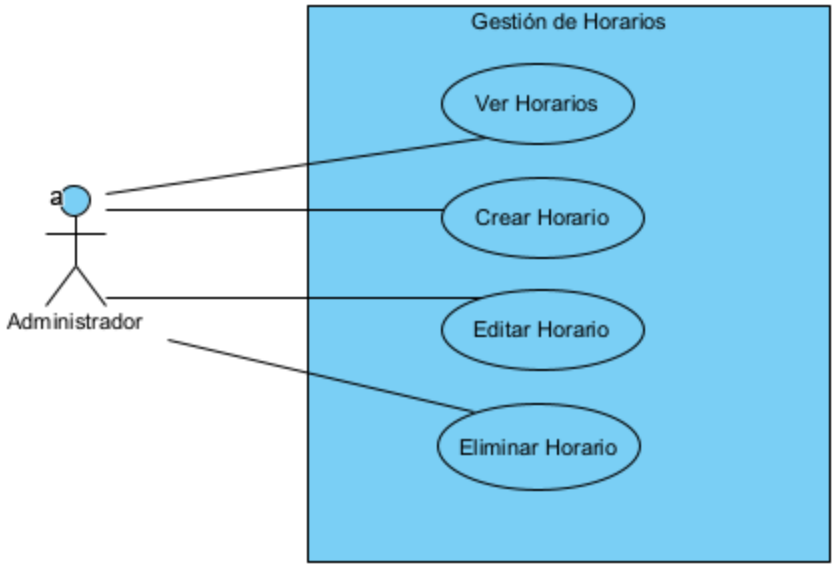

***2.1.3.2.10. REQ11: Reservas.*** Las reservas contarán con diferentes estados. Estos estados serán:

- Pendiente, estado de reserva asignado cuando esta fue creada.
- Confirmada, reserva que ha sido confirmada de manera manual o automática.
- Cancelada, reserva que ha sido cancelada previo a ser confirmada.
- Completada, reserva que confirma que el cliente ha llegado al restaurante.

***2.1.3.2.11. REQ12: Horario.*** Las reservas se podrán realizar durante diferentes días de la semana y con un horario definido por el restaurante.

**Figura 12**

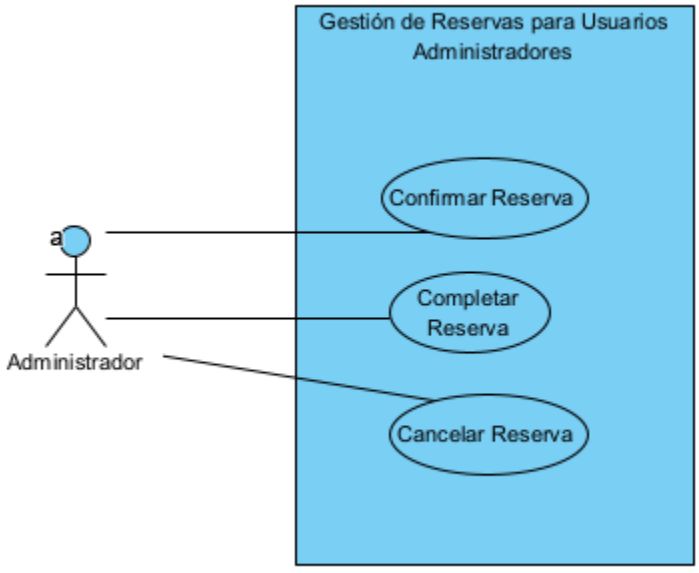
*Caso de uso del REQ12*




***2.1.3.2.12. REQ13: Gestión de Reservas para Usuarios Administradores.*** Los usuarios administradores podrán cambiar el estado de una reserva.

**Figura 13**

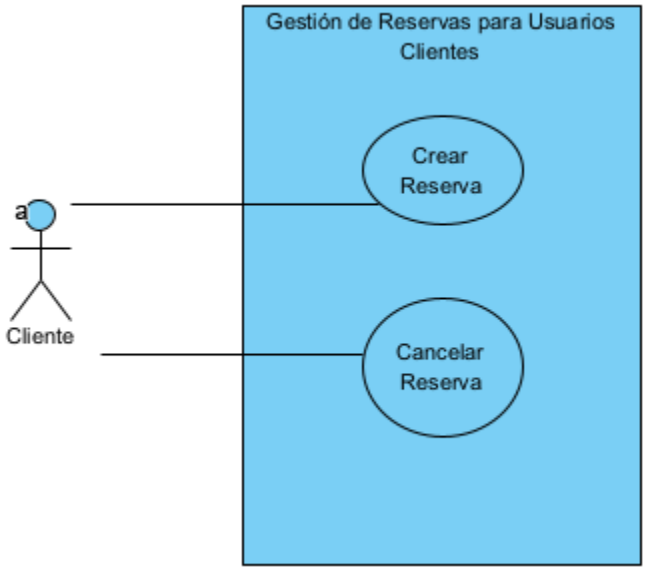
*Caso de uso del REQ13*



***2.1.3.2.13. REQ14: Gestión de Reservas para Usuarios Clientes.*** Los usuarios clientes podrán crear sus propias reservas, cancelarlas u observarlas.

**Figura 14**

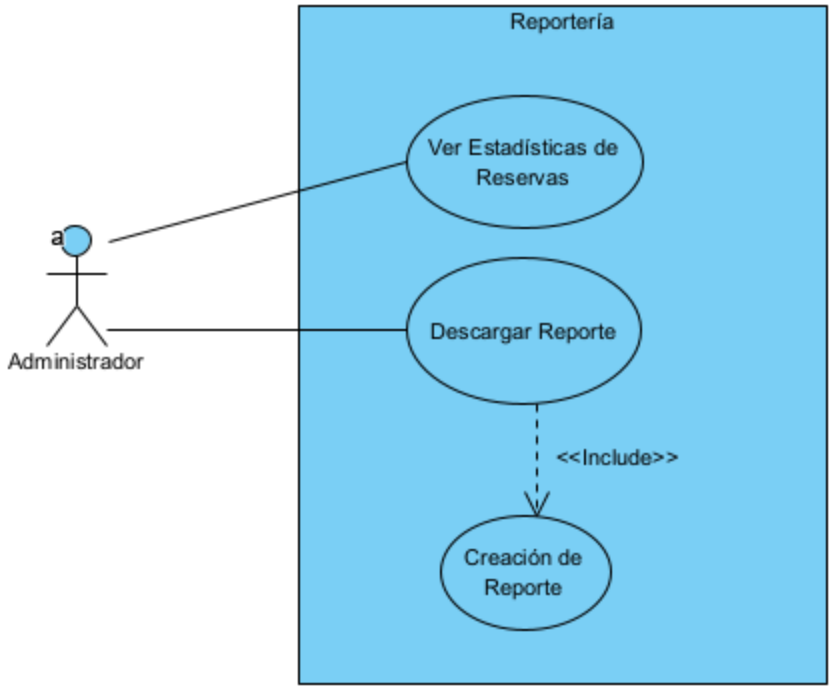
*Caso de uso del REQ14*




***2.1.3.2.14. REQ15: Reportería.*** El sistema podrá generar gráficos estadísticos de las reservas que el restaurante ha tenido.

**Figura 15**

*Caso de uso del REQ15*




**2.1.3.3. Requisitos de Rendimiento.**

***2.1.3.3.1. REQ16: Multiusuario.*** El sistema debe ser capaz de soportar más de un usuario conectado a la vez sin que sufra una disminución de rendimiento.

**2.1.3.4. Restricciones de Diseño.**

***2.1.3.4.1. REQ17: Plataforma de alojamiento.*** Al alojar el sistema en una plataforma tercerizada, el sistema tendrá que adaptarse al tipo de hardware que este provea según sus planes de servicios.

**2.1.3.5. Atributos del Sistema.**

***2.1.3.5.1. REQ18: Concurrencia.*** El aplicativo web debe contar con la capacidad de procesar requerimientos de manera concurrente, es decir, varias tareas se deben poder realizar al mismo tiempo.

***2.1.3.5.2. REQ19: Escalabilidad.*** El aplicativo debe contar con la capacidad de adaptarse a requerimientos futuros como el aceptar múltiples restaurantes.

***2.1.3.5.3. REQ20: Seguridad.*** El sistema debe poder salvaguardar la información de sus usuarios.

## 2.2. Análisis de las Soluciones Existentes (software libre y/o propietario)

### *2.2.1. OpenTable*

OpenTable es:

Con una red global de millones de comensales, más de 60,000 restaurantes asociados y la mejor plataforma de administración de restaurantes, OpenTable te ayuda a desarrollar tu negocio. Las soluciones fáciles de usar te ayudan a llenar las mesas, ejecutar y optimizar los turnos constantemente, fortalecer la conexión con los comensales y generar más ingresos. (Capterra, 2022)

Las funcionalidades de OpenTable recopiladas por Capterra (2022) son:

- Acceso móvil
- Alertas y notificaciones
- Asistencia al cliente
- CRM
- Creación de informes/análisis
- Creador de menú
- Gestión de cocinas y de menús
- Gestión de inventarios

- Gestión de pedidos
- Gestión de reservas
- Gestión de tablas
- Notas de reserva
- Pagos en línea
- Pedidos en línea
- Procesamiento de pagos
- Punto de venta (POS)
- Reserva en línea
- Reservas de grupo
- Seguimiento de pedidos
- Sistema de calendarios o recordatorios. (Sección de Funciones de OpenTable)

### *2.2.2. Resy*

Resy, según el artículo "Resy Network" (2021), puede ayudar a un restaurante a administrar reservas, listas de espera, eventos y más sin cargos de cobertura e integración de terminal de punto de venta o "*Point of Sale*" (POS). Para los restaurantes de servicio completo que buscan expandir su marca más allá del stand anfitrión, agilizar las operaciones, disminuir las tasas de ausencia y aumentar la rentabilidad.

Las funcionalidades de Resy recopiladas por Capterra (2021) son:

- Acceso móvil
- Alertas y notificaciones
- Asistencia al cliente

- CRM
- Creación de informes/análisis
- Facturación
- Gestión de cocinas y de menús
- Gestión de inventarios
- Gestión de reservas
- Gestión de tablas
- Notas de reserva
- Pedidos en línea
- Procesamiento de pagos
- Punto de venta (POS)
- Reserva en línea
- Reservas de grupo
- Sistema de calendarios o recordatorios. (Sección de Funciones de Resy OS)

### *2.2.3. Comparación de las Soluciones Existentes*

**Tabla 1**

*Comparación entre Resy y OpenTable*

| | **Resy** | **OpenTable** |
|---|---|---|
| **Gestión de reservas y listas de espera** | Si | Si |
| **Plano de planta personalizable** | Si | Si |
| **Widget de reserva en línea** | Si | Si |
| **Integración con Sistemas de punto de venta.** | Si | Si |
| **Prepago de Reservas y Depósitos** | Si | Si |
| **Encuestas posteriores a la cena** | Si | Si |
| **Gestión de capacidad** | Si | Si |

| | | |
|---|---|---|
| **Gestión de turnos** | Si | Si |
| **Reservas de Google, Instagram y TripAdvisor** | Si | Si |
| **Soporte para comidas sin contacto** | Si | No |
| **Soporte para comida para llevar y entregas** | Si | Si |
| **Gestión de invitados y CRM (Gestión de la Relación con el Cliente)** | Si | Si |
| **Análisis de Rendimiento** | Si | Si |
| **Auto incorporación** | No | No |
| **Aplicación web multiplataforma, aplicación para iPad, iOS y Android** | Si | No |
| **Precios** | $899 al mes | $449 al mes |

*Nota.* Alnasur (2019, 14 marzo).

Estos dos softwares propietarios tienen características muy similares, lo cual hace que sean buenas opciones para restaurantes con la capacidad de pago para sus servicios. La solución que se plantea en este trabajo de titulación busca ser más accesible al proporcionar solamente el apartado de reservas.

## 2.3. Desarrollo a la Medida

Al considerar la especificación de requerimientos del primer punto de este Capítulo II, se puede observar que el aplicativo será creado de manera general, por lo que el sistema debe ser capaz de adaptarse a cualquier restaurante que cuente con al menos una sucursal o matriz. Las reservas son una respuesta a que los comensales puedan ser rechazados al ir a un restaurante sin prever que no habría lugar.

### *2.3.1. Estudio del Entorno del Software*

En este apartado se definirá las herramientas a utilizar para el desarrollo del aplicativo web.

**2.3.1.1. Estudio de Modelos Arquitectónicos.** Los modelos arquitectónicos son definidos por Clements (1996) como las partes que componen un sistema, las especificaciones de comportamiento de esos componentes y los patrones y mecanismos de interacción entre ellos. Un solo sistema generalmente se compone de más de un tipo de componente: módulos, tareas, funciones, etc. Una arquitectura puede elegir el tipo de componente más apropiado o informativo para mostrar, o puede incluir múltiples vistas del mismo sistema, cada uno ilustrando diferentes componentes.

***2.3.1.1.1. Modelo Vista Controlador (MVC).*** El modelo-vista-controlador es un tipo de modelo arquitectónico que busca separar la lógica del programa (controlador) de la capa de presentación (vista) y que además cuenta con una capa de datas (modelo).

El modelo, como lo define Deacon (2009), es la esencia inmutable de la aplicación. En términos orientados a objetos, esto consistirá en el conjunto de clases que modelan y soportan el problema subyacente y que, por lo tanto, tenderán a ser estables y de larga duración como el problema mismo.

La vista, también definida por Deacon en 2009, es un conjunto de clases que tendrán la función de interfaces. Estas interfaces pueden tener las siguientes formas:

- La vista GUI/widget (interfaz gráfica de usuario)
- La vista CLI (interfaz de línea de comandos)
- La vista API (interfaz del programa de aplicación)

Esta capa debe conocer la existencia de los modelos, para poder mostrarlos o abrir un segmento que permita su modificación.

Por último, el controlador es un objeto/clase que puede manipular una vista. Este apartado maneja la entrada mientras que la vista modifica la salida. El controlador conoce las vistas, pero la vista no conoce el controlador.

***2.3.1.1.2. Modelo Vista Presentador (MVP).*** Diéguez (2015) define a este patrón como:

> MVP es un patrón de diseño que surge para ayudar a realizar pruebas automáticas de la interfaz gráfica, para ello la idea es codificar la interfaz de usuario lo más simple posible, teniendo el menor código posible, de forma que no merezca la pena probarla. En su lugar, toda la lógica de la interfaz de usuario, se hace en una clase separada (que se conoce como Presentador), que no dependa en absoluto de los componentes de la interfaz gráfica y que, por tanto, es más fácil de realizar pruebas.

Este patrón arquitectónico permite que una funcionalidad pueda tener diferentes vistas y que estás sean intercambiables (Diéguez, 2015). Tomado de sus siglas, podemos definir sus partes como: el modelo, que es el apartado donde se ejecuta la lógica de negocio; la vista, el apartado gráfico de la aplicación con la que interactúa el usuario; y el presentador, el intermediario de la vista y el modelo, que elige de qué forma se presentará la vista (Diéguez, 2015).

***2.3.1.1.3. Comparación de las Arquitecturas MVC y MVP.*** En la tabla 2 se presenta las características de los dos modelos explicados:

**Tabla 2**

*Comparación entre las características de MVC Y MVP*

| | **Contaminación de los Id de clientes** | **Acoplamiento entre vista y modo** | **Pruebas de unidad de soporte** | **Admite multiplexación en la lógica de** | **Complejidad** |
|---|---|---|---|---|---|

| | | | | | |
|---|---|---|---|---|---|
| | | | | **la capa de presentación** | |
| **MVC** | No | Si | Si | No | Alta |
| **MVP** | No | No | Si | Si | Relativamente alta |

*Nota*. Gu y Tang (2014, p. 394)

MVC provee un mayor control de los documentos HTML, en relación con el estándar web y la accesibilidad, integración con frameworks de JavaScript y realizar diferentes pruebas unitarias (Gu y Tang, 2014). Por esta razón, esta arquitectura será escogida para el desarrollo.

**2.3.1.2. Selección de Framework a usar para el Desarrollo del *Backend*.** En el Capítulo II se introdujo al *framework* Laravel, este marco de trabajo será utilizado debido a su afinidad con el modelo arquitectónico MVC.

En la tabla 3 se presenta las características de Laravel comparadas con el *framework* Slim desarrollado también con el lenguaje de programación PHP.

**Tabla 3**

*Comparación entre los Frameworks Laravel y Slim*

| | **PHP Laravel Framework** | **PHP Slim Framework** |
|---|---|---|
| **Categoría** | Full Stack framework | Micro framework |
| **Autor** | Taylor Otwell | Jos Lockhart |
| **Versión más actual** | 5.8.10 | 3.12.1 |
| **Fecha de lanzamiento** | 2019, 26 de febrero | 2019, 16 de abril |
| **Licencia** | MIT License | MIT License |
| **Requerimientos** | >= PHP 7.1.3 | >= PHP 5.6 |
| **ORM** | Eloquent | Eloquent |
| **Generador de código** | CLI | CLI |
| **Marco de seguridad** | Si | Si |
| **Marco de plantilla** | Si | Si |
| **Motor de plantillas** | Blade | Twig, PHP View |
| **Generador de CRUD** | CRUD Booster | Scaffold |
| **Sitio web** | www.laravel.com | www.slimframework.com |
| **Lenguaje** | PHP 7 | PHP 7 |
| **Framework MVC** | Si | Si |

| | | |
|---|---|---|
| **Framework de pruebas** | PHP Unit | PHP Unit |
| **Validaciones** | Si | No |
| **Subir información** | Si | Si |
| **Base de datos** | Si | Si |
| **Autenticación** | Si | No |
| **Migración de DB** | Si | No |
| **Validación de formularios** | Si | No |
| **RAD** | Si | Si |
| **Número de archivos** | 7958 | 2185 |
| **Número de carpetas** | 1377 | 390 |
| ***Pingback*** | Si | Si |
| **Movilidad** | Si | Si |
| **Rastreador** | Si | Si |
| **Autofocus** | No | Si |
| **Instalación por medio de Composer** | Si | Si |
| **Formularios** | Objetos | Objetos |
| **Búsqueda de textos** | Si | Si |
| **Público objetivo** | Proyectos grandes | Proyectos Pequeños |

*Nota*. Sunardi (2019, p. 137).

Al conocer las características disponibles en Laravel se lo ha escogido debido a su integración con diferentes herramientas como bases de datos hasta otros frameworks de *frontend.*

**2.3.1.3. Selección de Framework a usar para el Desarrollo del *Frontend*.** En la actualidad existen varios *frameworks* usados para la capa gráfica presentada al usuario. Debido a la selección de Laravel como framework principal para el *backend*, se ha escogido a Livewire, framework presentado en el Capítulo II, para mostrar las vistas de usuarios. A su vez se usará el framework Alpine.js, también presentado en el Capítulo II, conjuntamente para crear una web dinámica con las propiedades que brinda el lenguaje de programación JavaScript.

**2.3.1.4. Selección del Motor de la Base de Datos.** La información generada por el aplicativo será almacenada en una base de datos (BD) relacional. Para Suárez (2018), estos tipos de BD pueden establecer relaciones entre varios datos representados en tablas. Para manipular la información de estas bases de datos es necesario usar el estándar SQL (Structured Query

Languaje, lenguaje de consulta estructurado). Este tipo de lenguaje es usado por los motores de bases de datos encargados de gestionar la BD.

***2.3.1.4.1. MySQL.*** Es un sistema gestor de bases de datos (SGBD) usado en diferentes aplicaciones por su facilidad de uso, rendimiento y rápida implementación. Su distribución es GPL (Licencia Pública General) relacionada con el software libre y cuenta con varias versiones estables (Santillán et al, 2014, p. 5)

***2.3.1.4.2. PostgreSQL.*** Chávez (2020) define a PostgreSQL como:

> PostgreSQL es un gestor de bases de datos relacionales organizado en diferentes niveles. El nivel más externo es el clúster. En un servidor se pueden instalar varios clústeres del gestor, siempre y cuando cada uno de ellos utilice un puerto diferente. El puerto por defecto es 5432.
>
> Un clúster contiene bases de datos. Ésta, es una colección de esquemas (schema), que, a su vez, contiene los objetos, que se conocen el contexto de técnicas de almacenamiento de datos, como tablas (table), vistas (view), secuencias (sequence) y así seguido. (p. 6)

***2.3.1.4.3. Comparación entre MySQL y PostgreSQL.*** Los dos motores de bases de datos cuentan con diferentes versiones que han sido lanzadas desde su nacimiento, estas cuentan con diferentes características, la siguiente tabla muestra algunas características disponibles en PostgreSQL 8.0 y MySQL 4.1 y 5.0

**Tabla 4**

*Comparación entre PostgreSQL y MySQL 4.1 y 5.0*

| | **PostgreSQL 8.0** | **MySQL 4.1** | **MySQL 5.0** |
|---|---|---|---|
| **Sistemas Operativos** | Windows, Linux, todos los BSDs, HP-UX, AIX, OS X, Unixware, Netware... | Linux, Windows, FreeBSD, MacOS X, Solaris, HP UX, AIX, y otros | Linux, Windows, FreeBSD, MacOS X, Solaris, HP UX, AIX, y otros |
| **Compilación ANSI SQL** | ANSI-SQL 92/99 | Posible | Posible |
| **Sub-selects** | Si | Si | Si |
| **Transacciones** | Si | Si<br>Tablas de InnoDB | Si |
| **Replicación de la base de datos** | Si | Si | Si |
| **Soporte de llaves foráneas** | Si | Si<br>Tablas de InnoDB | Si |
| **Vistas** | Si | No | Si |
| ***Stored procedures*** | Si (pl/SQL) | No | Si |
| ***Triggers*** | Si | No | Si |
| ***Unions*** | Si | Si | Si |
| ***Full joins*** | Si | No | No |
| ***Constraints*** | Si | No | No |
| ***Cursors*** | Si | No | Parcial |
| **Lenguajes de procedimiento** | Si | No | Si |
| **Vacuum** | Si | Si | Si |
| **Diferentes tipos de tablas** | No | Si | Si |
| **ODBC** | Si | Si | SI |
| **JDBC** | Si | Si | Si |
| **Otras API** | Casi todos los lenguajes | Casi todos los lenguajes | Casi todos los lenguajes |
| **IPv6** | Si | No | No |

*Nota*. Klimek y Skublewska-Paszkowska (2021, p. 54)

Al analizar las posibilidades de ambos motores de bases de datos, se ha escogido a MySQL para ser utilizado en el ambiente del aplicativo web.

# Capítulo III: Desarrollo de la Investigación

## 3.1. Product Backlog

El *product backlog* se definirá junto con los requerimientos del Capítulo II

**Tabla 5**

*Product backlog del aplicativo web para la gestión de reservas de un restaurante*

| Id. | Requerimiento | Responsable |
|---|---|---|
| **1** | Interfaz Responsive | Erick Rea |
| **2** | Versión Completa | Erick Rea |
| **3** | Documentación | Erick Rea |
| **4** | Gestión de Restaurante | Erick Rea |
| **5** | Gestión de Sucursales | Erick Rea |
| **6** | Gestión de Mesas | Erick Rea |
| **7** | Control de Acceso | Erick Rea |
| **8** | Gestión de Roles | Erick Rea |
| **9** | Gestión de Usuarios Administradores | Erick Rea |
| **10** | Gestión de Usuarios Clientes | Erick Rea |
| **11** | Reservas | Erick Rea |
| **12** | Horario | Erick Rea |
| **13** | Gestión de Reservas para Usuarios Administradores | Erick Rea |
| **14** | Gestión de Reservas para Usuarios Clientes | Erick Rea |
| **15** | Reportería | Erick Rea |
| **16** | Multiusuario | Erick Rea |
| **17** | Concurrencia | Erick Rea |
| **18** | Escalabilidad | Erick Rea |
| **19** | Seguridad | Erick Rea |
| **20** | Implementación en plataforma online | Erick Rea |

## 3.2. Sprint I

### *3.2.1. Sprint Planning*

El alcance de este Sprint abarca la fase inicial del proyecto. Se inicializará el aplicativo realizado con el framework Laravel y se empezará a poblar las tablas necesarias que serán utilizadas en la base de datos para obtener el primer prototipo, además de los requerimientos que serán tomados para el *sprint backlog*.

### *3.2.2. Sprint Backlog*

Este sprint deberá lograr un primer acercamiento de las siguientes historias de usuario definidas en el *product backlog.*

**Tabla 6**

*Backlog del Sprint I*

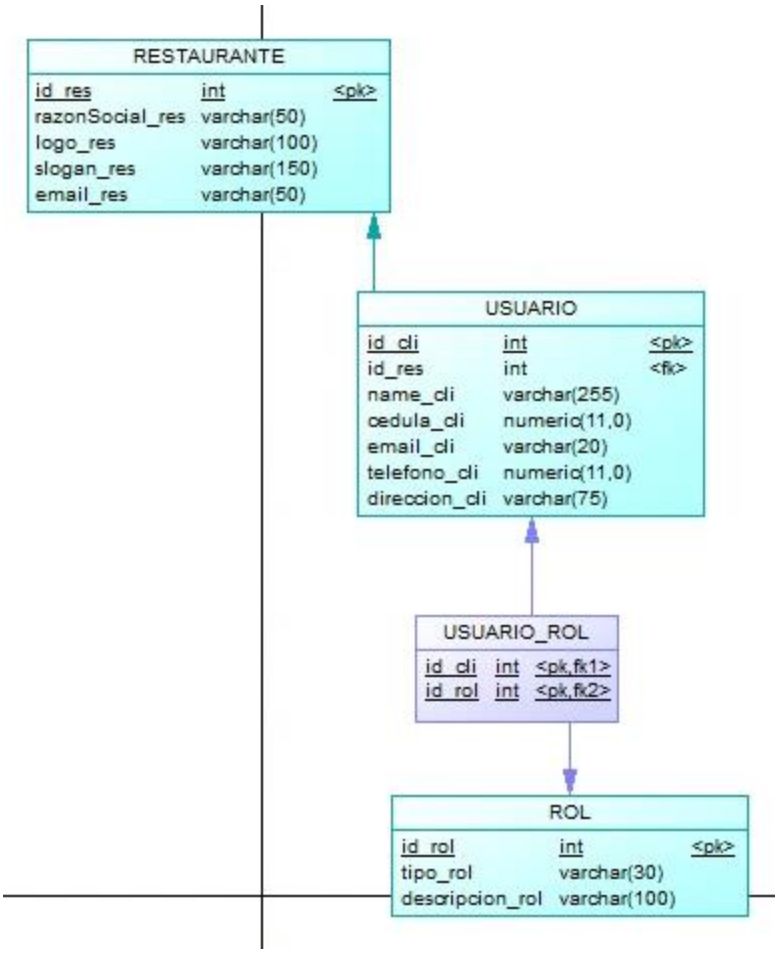
| Id. | Requerimiento | Responsable |
|---|---|---|
| **4** | Gestión de Restaurante | Erick Rea |
| **7** | Control de Acceso | Erick Rea |
| **8** | Gestión de Roles | Erick Rea |

### *3.2.3. Diseño*

La base de datos tendrá el siguiente modelo inicial.

**Figura 16**

*Primera versión de la base de datos*




Esta versión de la base de datos toma en cuenta que un usuario puede estar asignado a un restaurante y que a su vez este tendrá un rol. El rol será monitoreado por el aplicativo para poder restringir las vistas del sistema a los roles existentes.

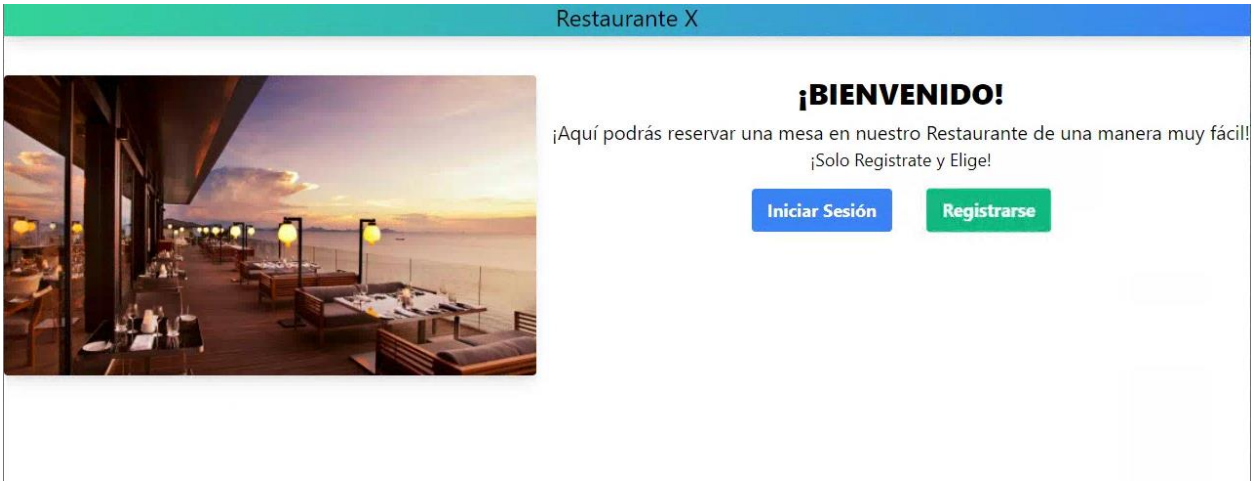
### *3.2.4. Prototipo*

#### 3.2.4.1. Control de Acceso

***3.2.4.1.1. Landing Page.*** Como primer apartado se creó una *landing page* para que los usuarios puedan escoger entre iniciar sesión o registrarse.

**Figura 17**

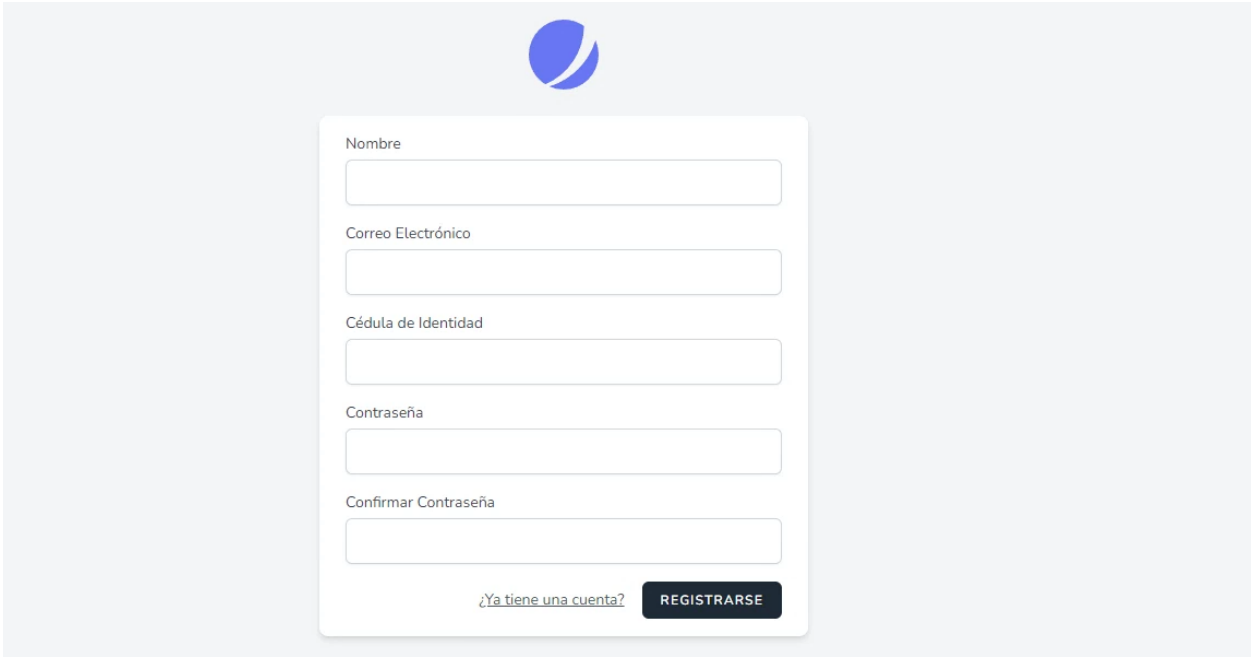
*Landing page inicial*




***3.2.4.1.2. Página de Registro.*** Página inicial donde el usuario podrá registrarse. Para geste apartado se usó la tecnología Jetstream propia de Laravel para generar esta vista

**Figura 18**

*Registro de usuarios clientes*

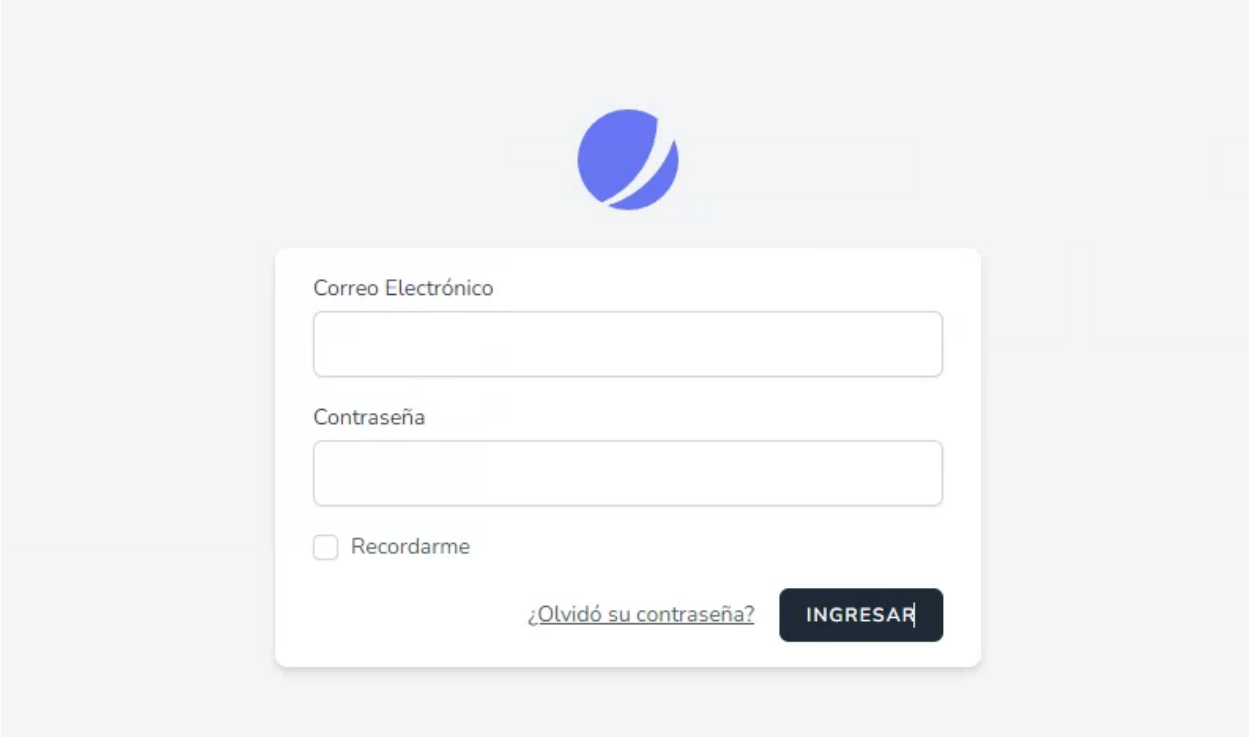



El usuario debe ingresar su número de cédula de identidad para que el registro sea válido.

***3.2.4.1.3. Página de Inicio de Sesión.*** En este apartado el usuario puede ingresar las credenciales con las que se registró y poder visualizar información propia de su tipo rol.

**Figura 19**

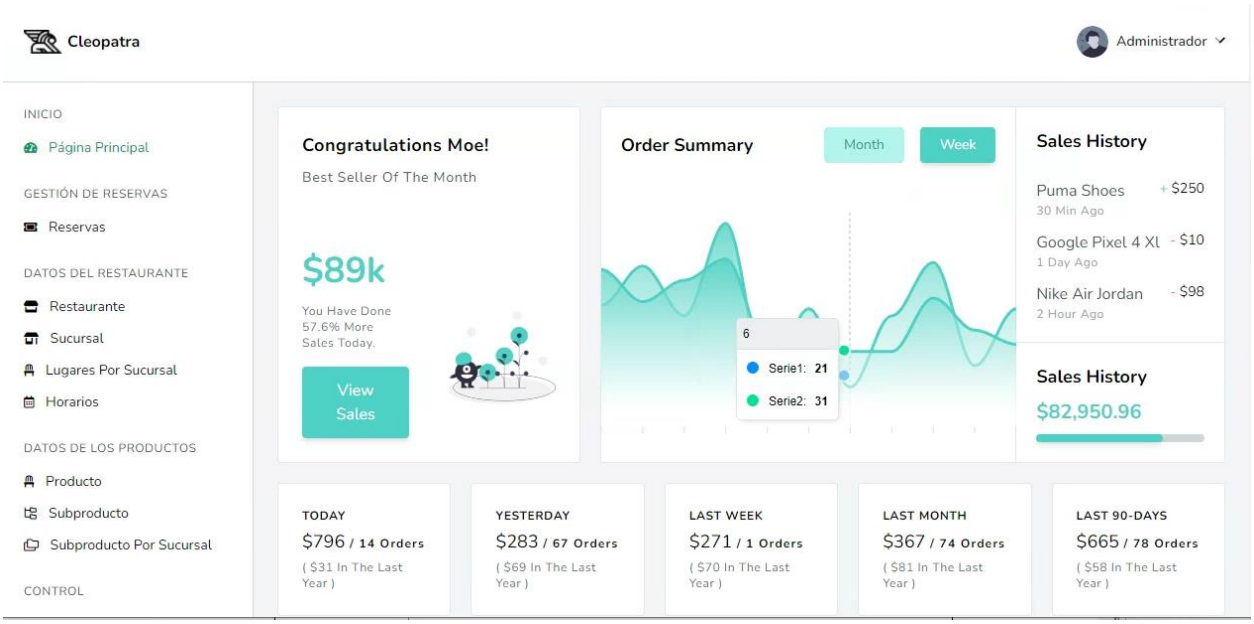
*Página de Inicio de Sesión*




***3.2.4.1.4. Página Principal – Usuario Administrador.*** Esta es la primera página visible al iniciar sesión como usuario administrador.

**Figura 20**

*Página principal de usuario administrador*

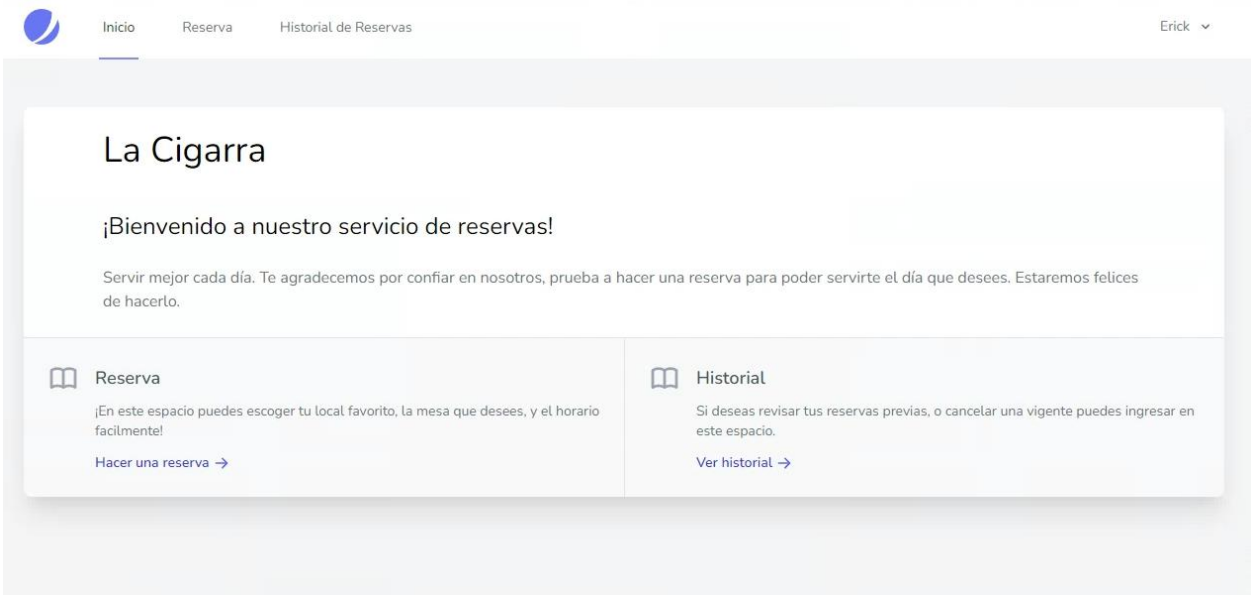



Para la creación de las vistas del usuario administrador se utilizó el tempate Cleopatra con licencia MIT.

***3.2.4.1.5. Página Principal – Usuario Cliente.*** Esta es la primera página visible al iniciar sesión como usuario cliente.

**Figura 21**

*Página principal al iniciar sesión como usuario cliente*




**3.2.4.2.1. Gestión de Roles.** Los roles podrán crearse solamente siendo un usuario administrador. Las opciones existentes son la de crear un nuevo rol, editar un rol y eliminar un rol.

**Figura 22**

*Página principal al seleccionar la opción rol como administrador*

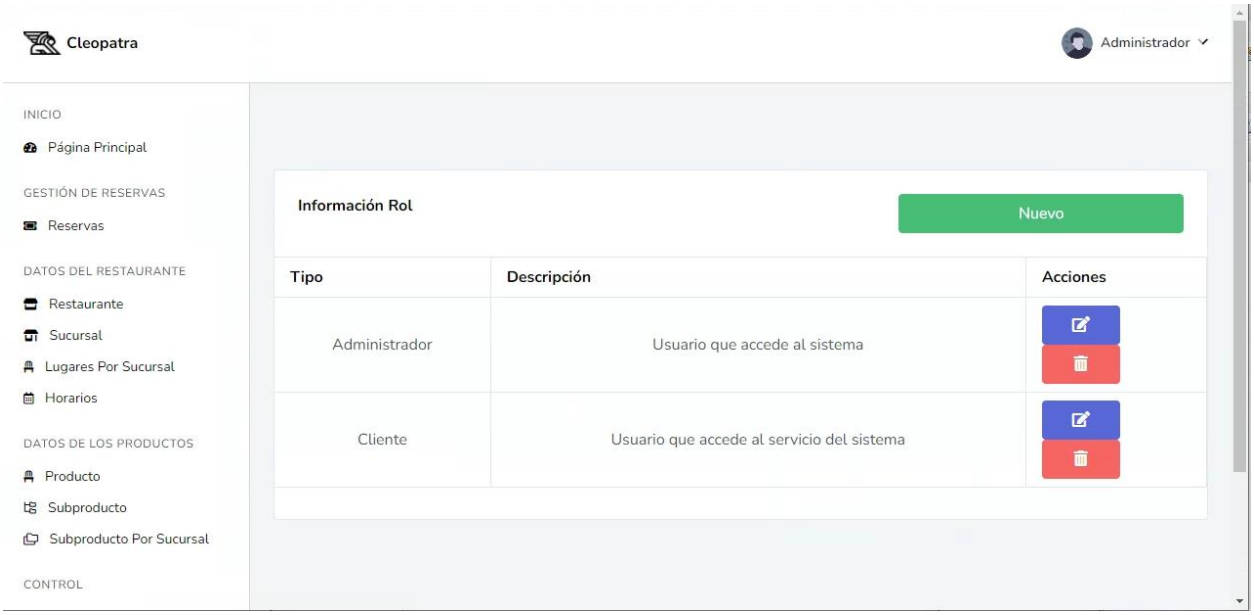


**Figura 23**

*Modal mostrado al dar clic en la opción de nuevo rol*

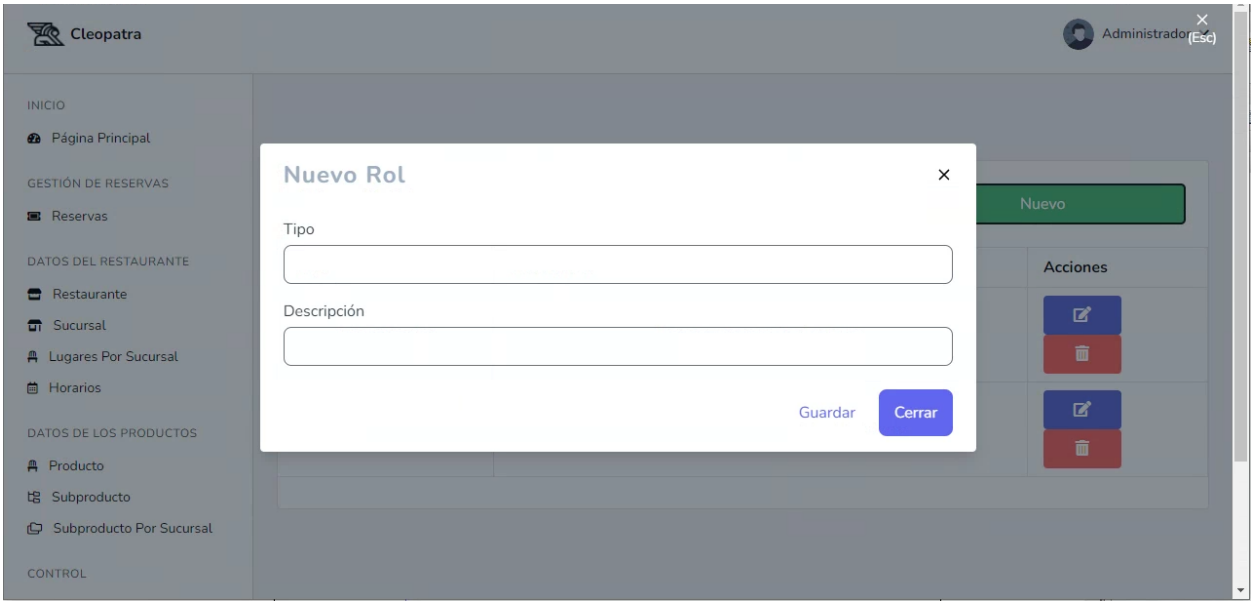



**Figura 24**

*Modal mostrado al dar clic en la opción de editar un rol*

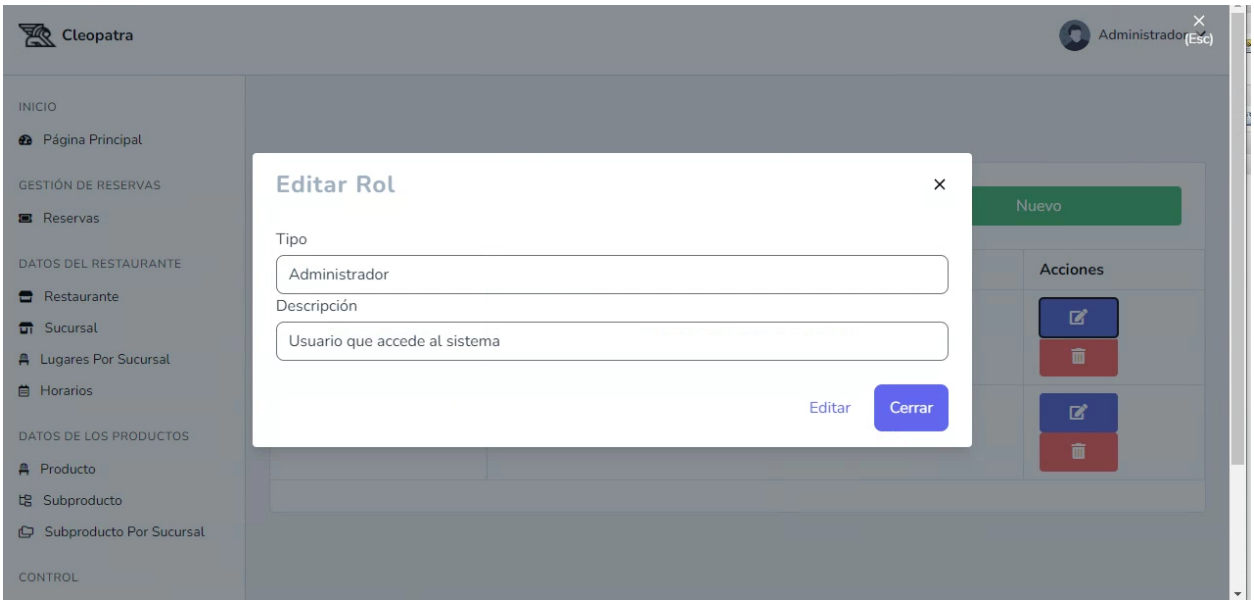
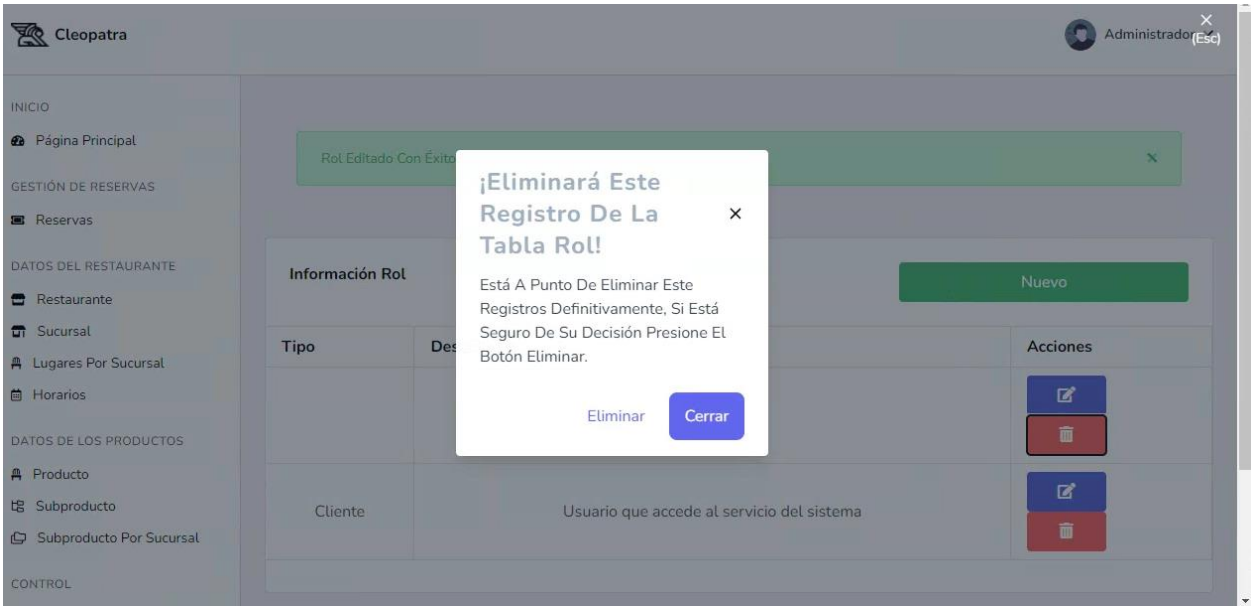


**Figura 25**

*Modal mostrado al dar clic sobre la opción de eliminar rol*




**3.2.4.2.2. Gestión de Restaurantes.** Para la gestión de restaurantes se tomó de base la vista de gestionar roles. A esta vista se le ha agregado validaciones a la hora de crear y editar un restaurante.

**Figura 26**

*Página principal al seleccionar la opción rol como administrador*

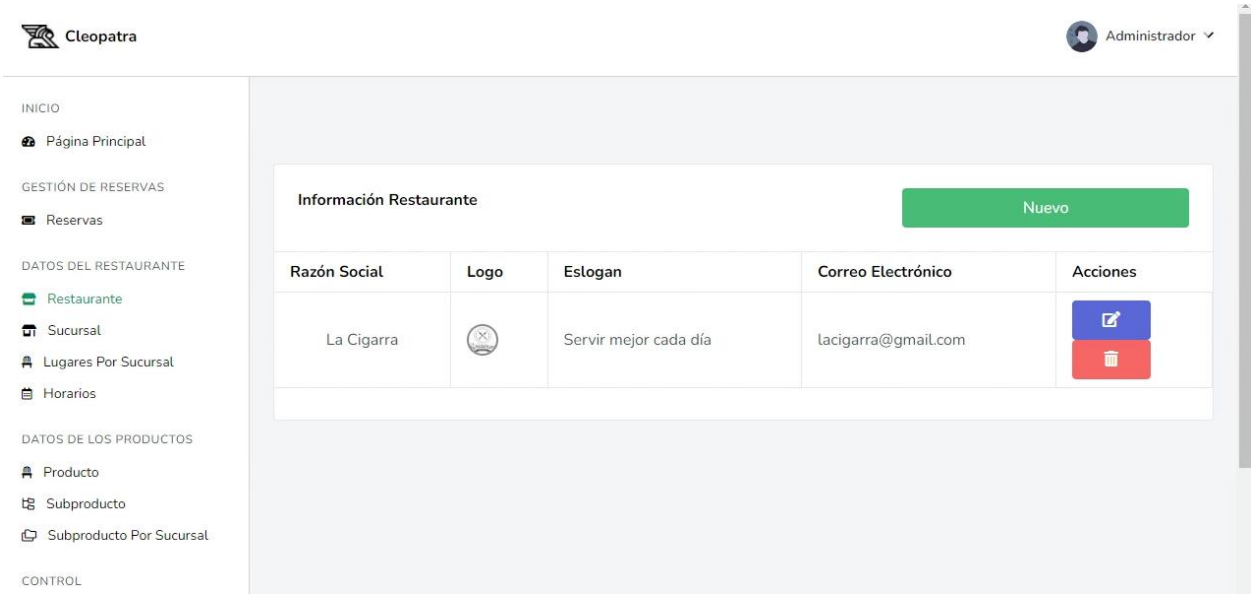



**Figura 27**

*Modal mostrado al dar clic en la opción de nuevo restaurante*

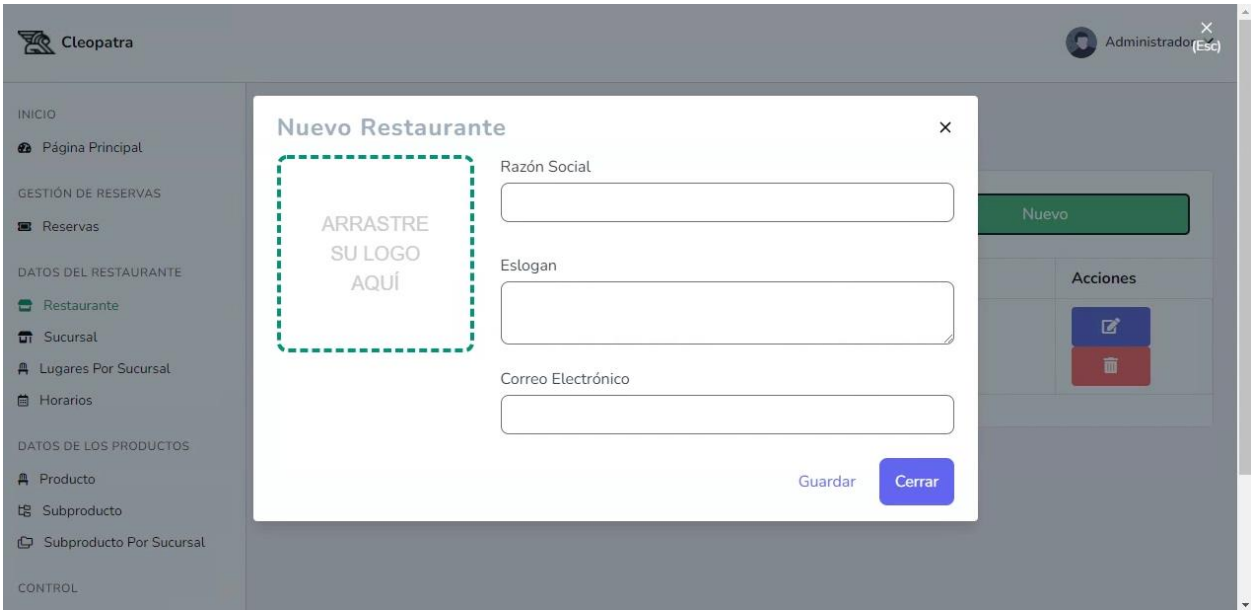



**Figura 28**

*Validación mostrada al intentar guardar un restaurante*

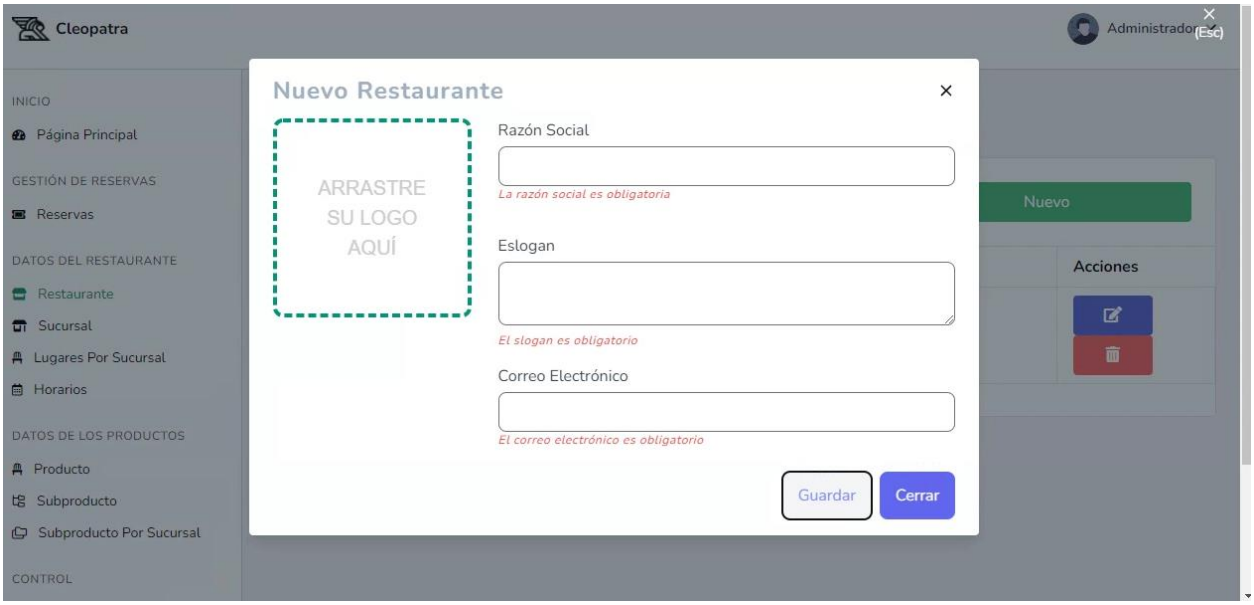



**Figura 29**

*Modal mostrado al momento de dar clic sobre editar un restaurante*

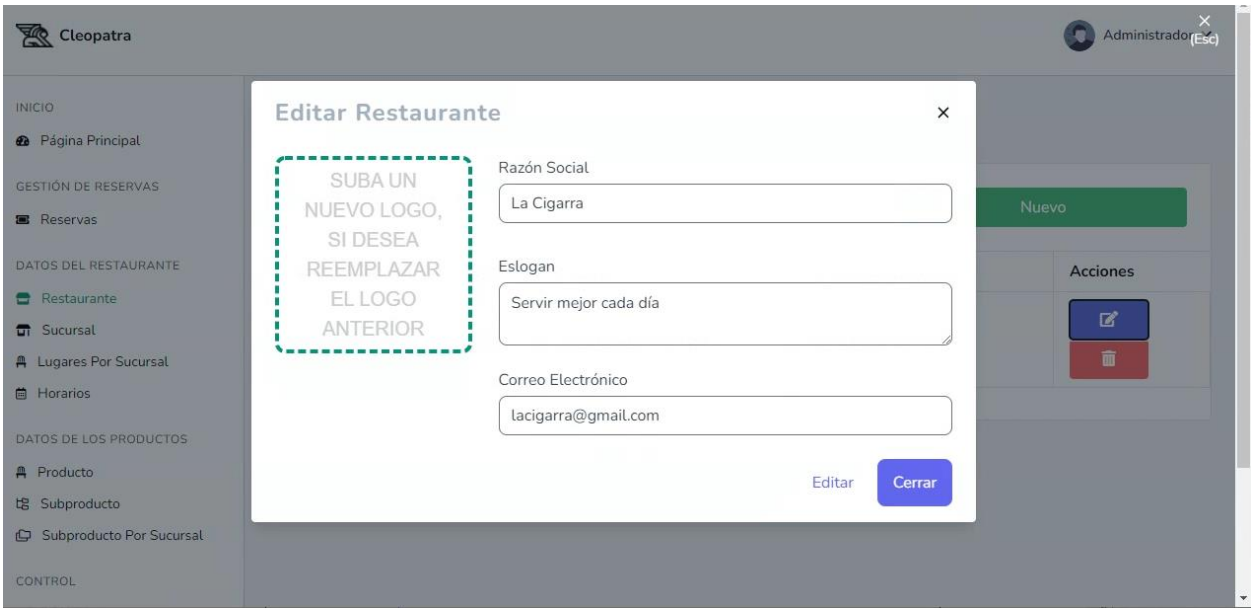



**Figura 30**

*Validación mostrada al momento de intentar actualizar el restaurante sin información*

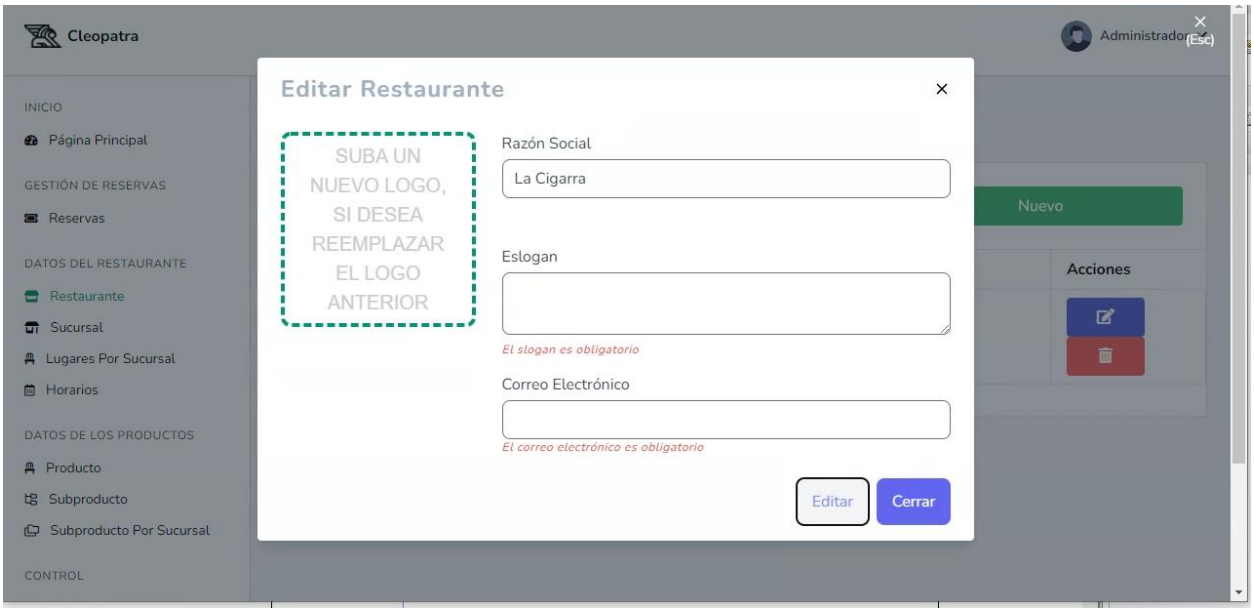
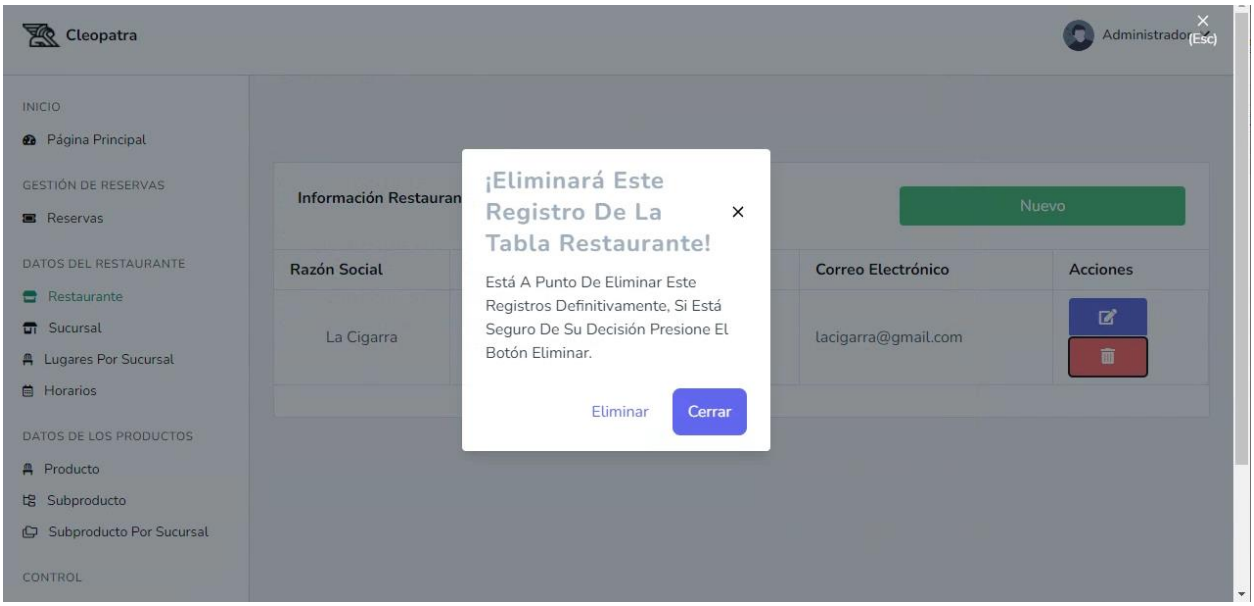



**Figura 31**

*Modal mostrado al momento de intentar eliminar un restaurante*




### *3.2.5. Sprint Review*

Se ha cumplido la totalidad de los avances planteados durante el Sprint I. Aunque, existen las siguientes observaciones:

- Modificar la página principal del sitio web, esta cuenta con un diseño temporal.
- Modificar las vistas de registro y inicio de sesión, ya que estas están hechas con la vista por defecto generada por Laravel.
- Modificar la página principal de usuario cliente, debido a que esta es visible con la vista por defecto generada por Laravel.
- Verificar las validaciones al momento de gestionar roles.

#### *3.2.6. Sprint Retrospective*

Este es el último evento del *sprint* donde de la mano de un profesional se da el visto bueno al aplicativo.

**Tabla 7**

*Observaciones del experto en el tema*

| **Nombre del Experto** | **Correo Electrónico** | **Observaciones** |
|---|---|---|
| Damián Nicolalde | danicolalde@puce.edu.ec | Validar todos los campos al momento de guardar un campo. |
| | | Validar todos los campos al momento de editar un campo. |
| | | Modificar las vistas por defecto de las plantillas del aplicativo. |

### 3.3. Sprint II

#### *3.3.1. Sprint Planning*

Este *sprint* toma en cuenta los resultados del anterior *sprint* que son: la gestión información de los roles y del restaurante. Las interfaces que se usaron para gestionar dicha

formación serán reusadas para poder administrar información de las sucursales que pueda tener un restaurante y las mesas de estas. Por último, se realizará una interfaz para gestionar información de los usuarios que serán administradores para el restaurante.

### *3.3.2. Sprint Backlog*

De acuerdo con el *sprint planning* los requerimientos que cumplen lo estimado se muestran a continuación:

**Tabla 8**

*Backlog del Sprint II*

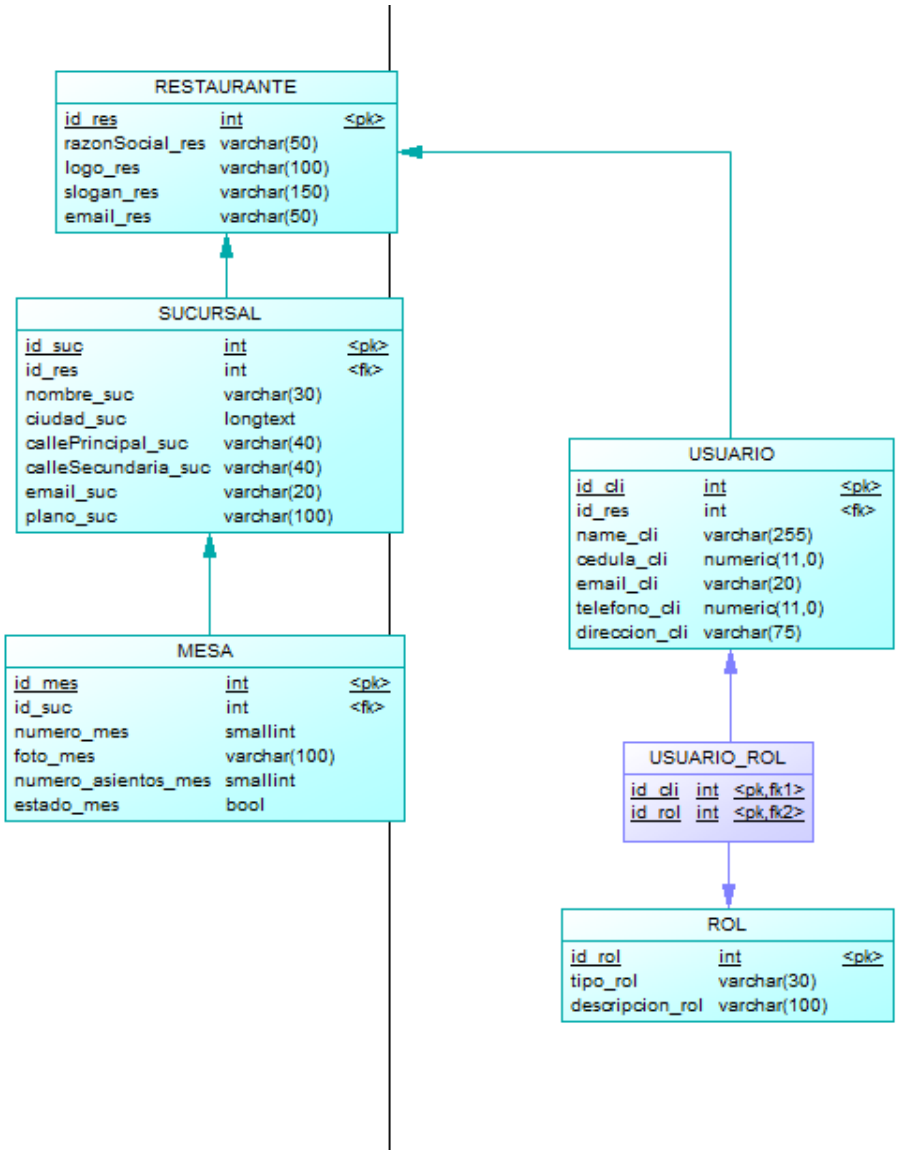
| Id. | Requerimiento | Responsable |
|---|---|---|
| **5** | Gestión de Sucursales | Erick Rea |
| **6** | Control de Mesas | Erick Rea |
| **9** | Gestión de Usuarios Administradores | Erick Rea |

### *3.3.3. Diseño*

En este Sprint II, la base de datos fue modificada para que un restaurante pueda tener varias sucursales y que estos locales puedan registrar información de las localizaciones (mesas) con las que cuentan.

**Figura 32**

*Segunda versión de la base de datos*

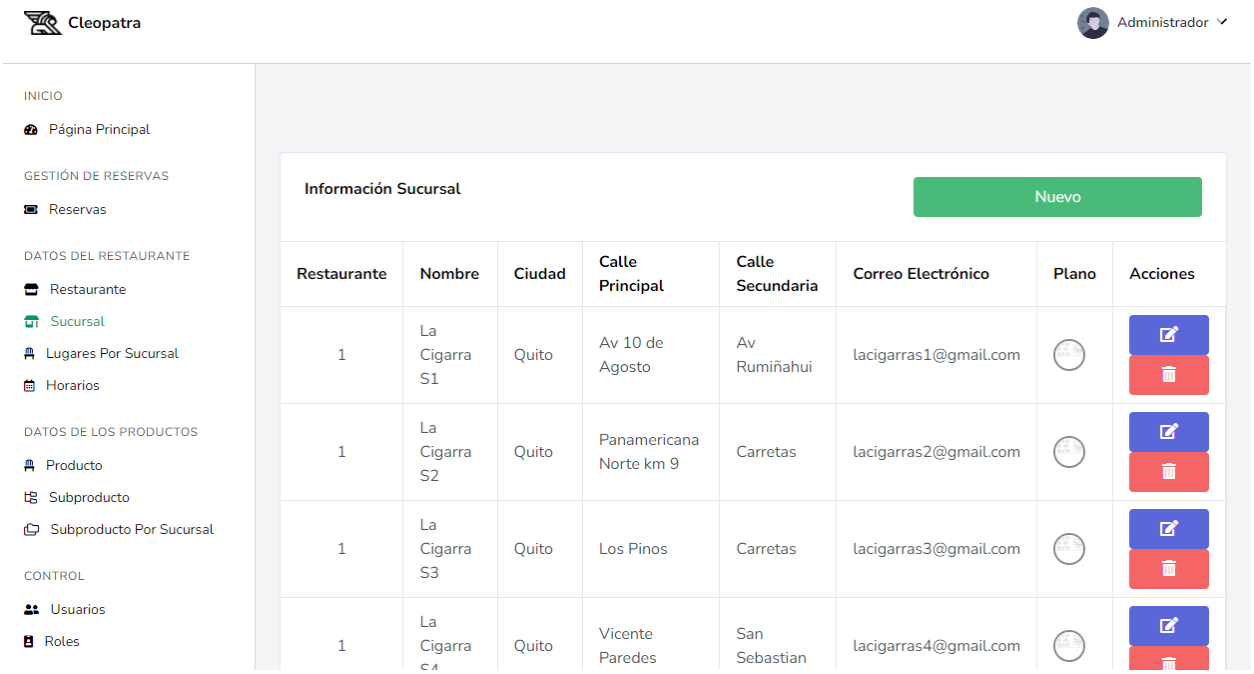



### *3.3.4. Prototipo*

**3.3.4.1. Gestión de Sucursales.** Para realizar el CRUD de las sucursales se reutilizó la misma interfaz gráfica que fue usada para gestionar roles y restaurantes. Se puede crear, ver, editar y eliminar información de las sucursales.

**Figura 33**

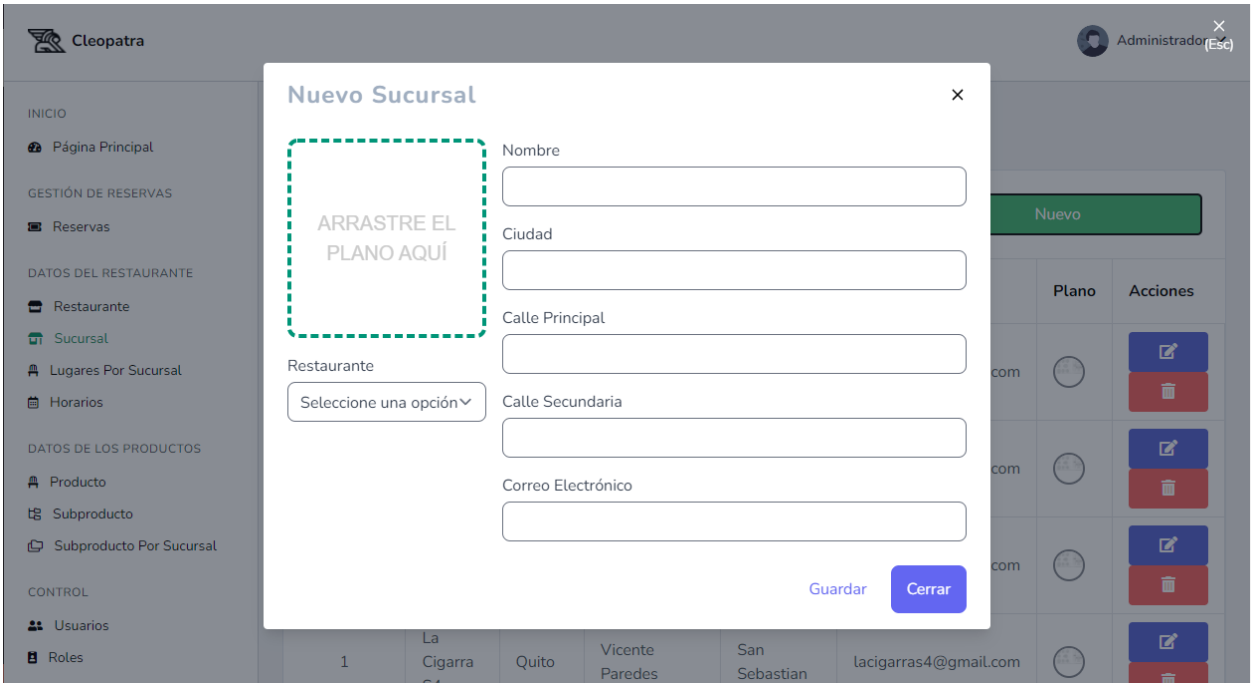
*Página principal de la opción Sucursal como administrador*




*Nota*. Esta interfaz es mostrada al dar clic sobre el enlace Sucursal en el menú lateral.

**Figura 34**

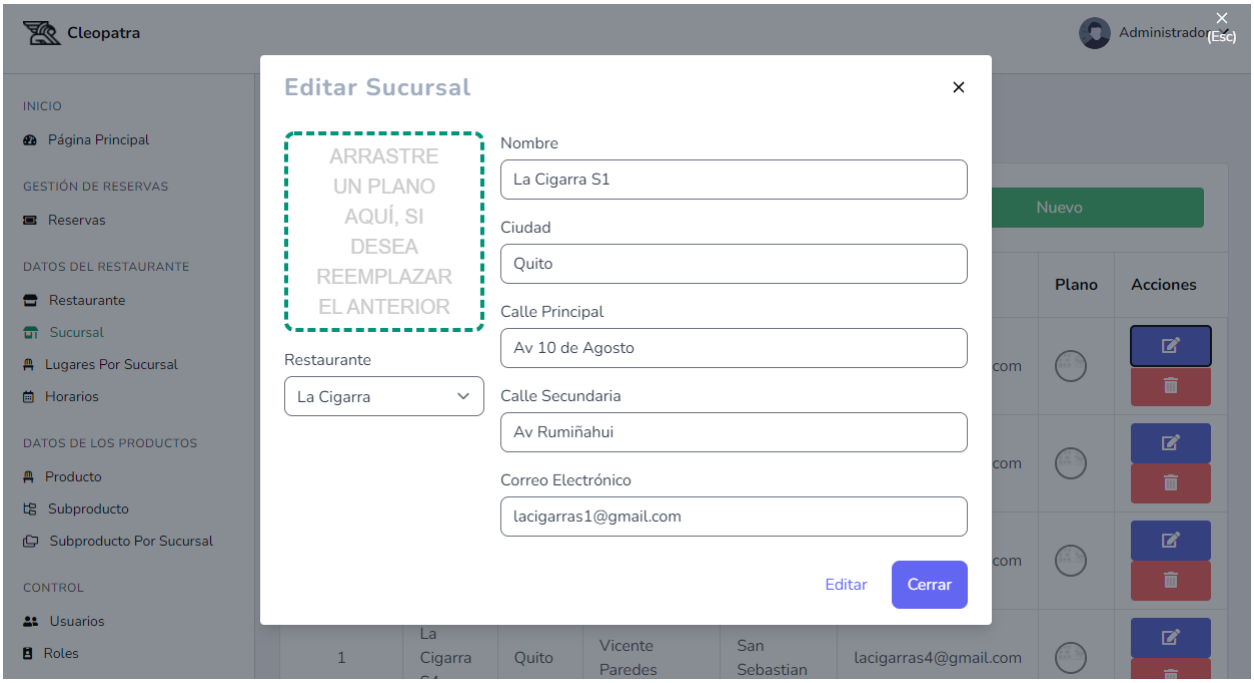
*Modal mostrado al dar clic en la opción de nueva sucursal*




*Nota*. La ventana emergente mostrada permite ingresar diferente tipo de información de la sucursal, además que permite seleccionar a que restaurante pertenece la misma. También permite subir una imagen del local.

**Figura 35**

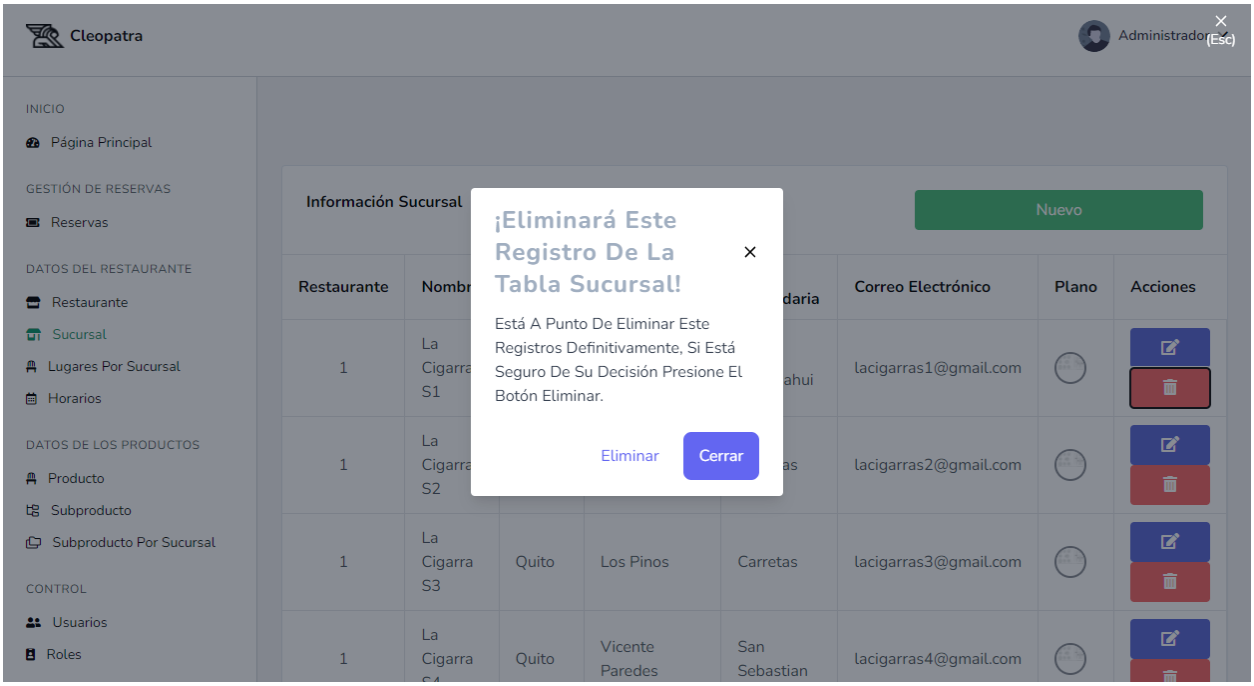
*Modal mostrado al momento de dar clic sobre editar una sucursal*




*Nota*. La ventana emergente muestra la información previamente cargada de la sucursal a la que se le dio clic.

**Figura 36**

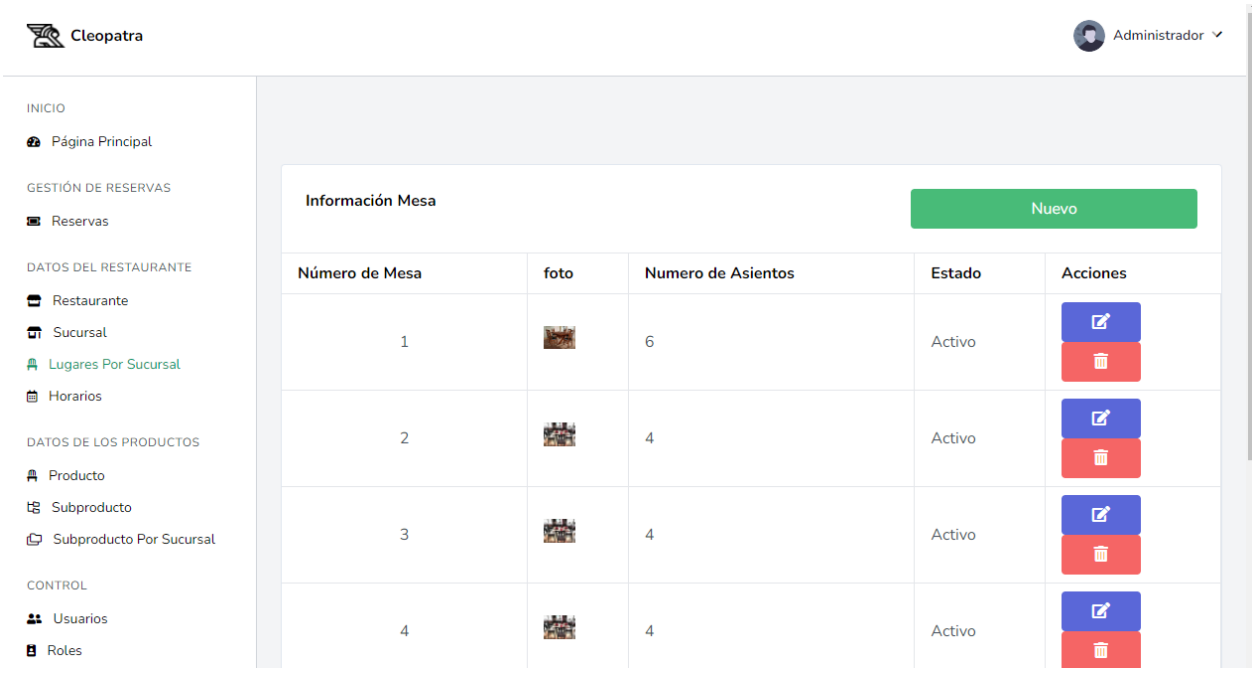
*Modal mostrado al momento de intentar eliminar una sucursal*




**3.3.4.2. Gestión de Mesas.** Al igual que los CRUD anteriores se reutilizó la interfaz gráfica para poder añadir, ver, editar y eliminar información de las mesas de las sucursales.

**Figura 37**

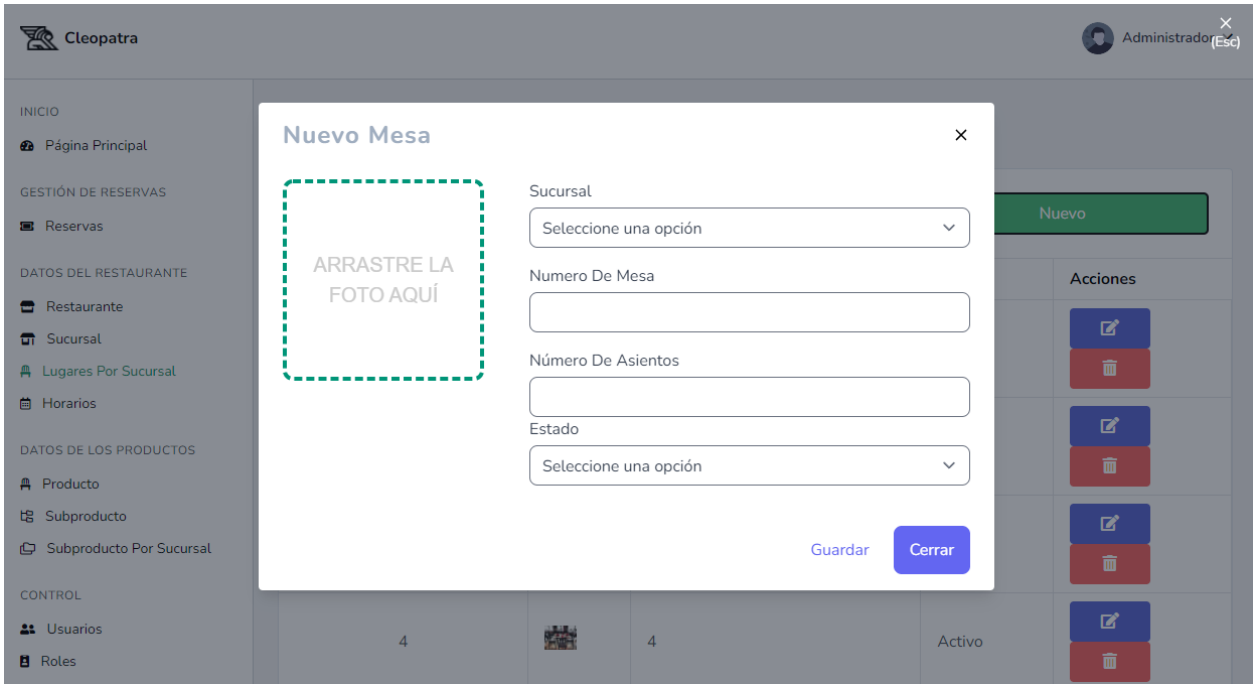
*Página principal de la opción "Lugares Por Sucursal" como administrador*




*Nota*. Esta interfaz es mostrada al dar clic sobre el enlace Lugares por Sucursal en el menú lateral.

**Figura 38**

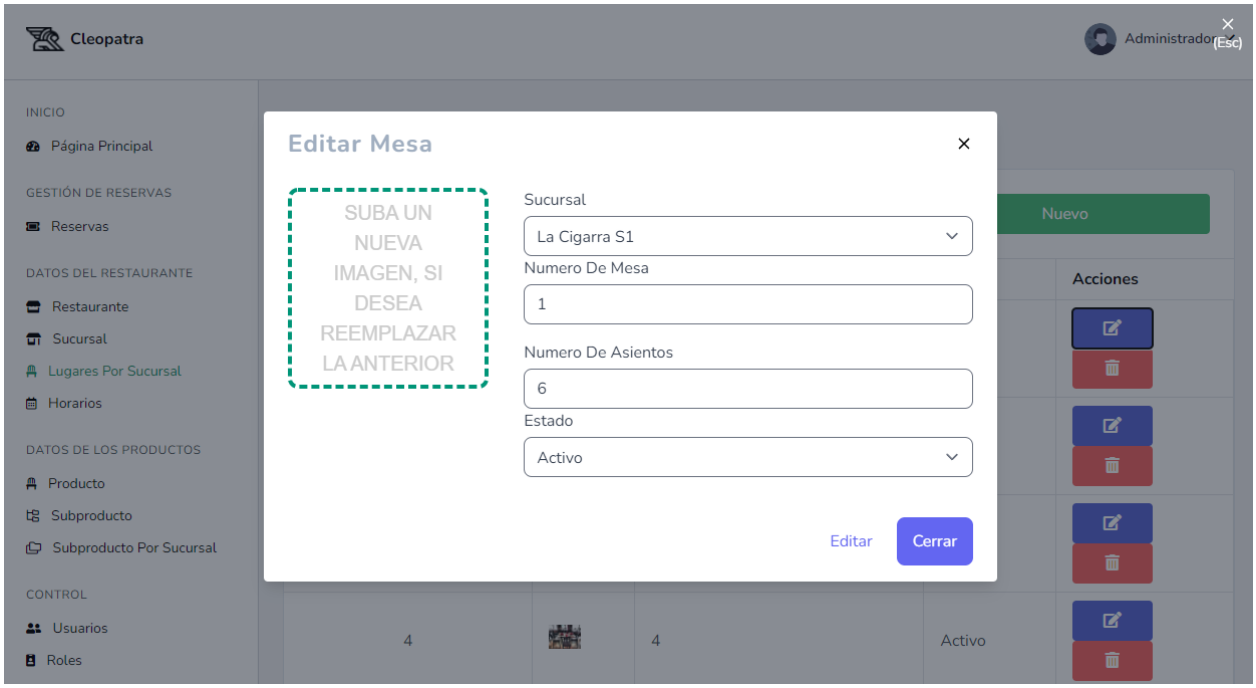
*Modal mostrado al dar clic en la opción de Nuevo*




*Nota*. La ventana emergente mostrada permite ingresar diferente tipo de información de la mesa como la sucursal a la que pertenece, el número que se le ha asignado, su número de asientos y si está activa o no. También permite subir una imagen de la mesa.

**Figura 39**

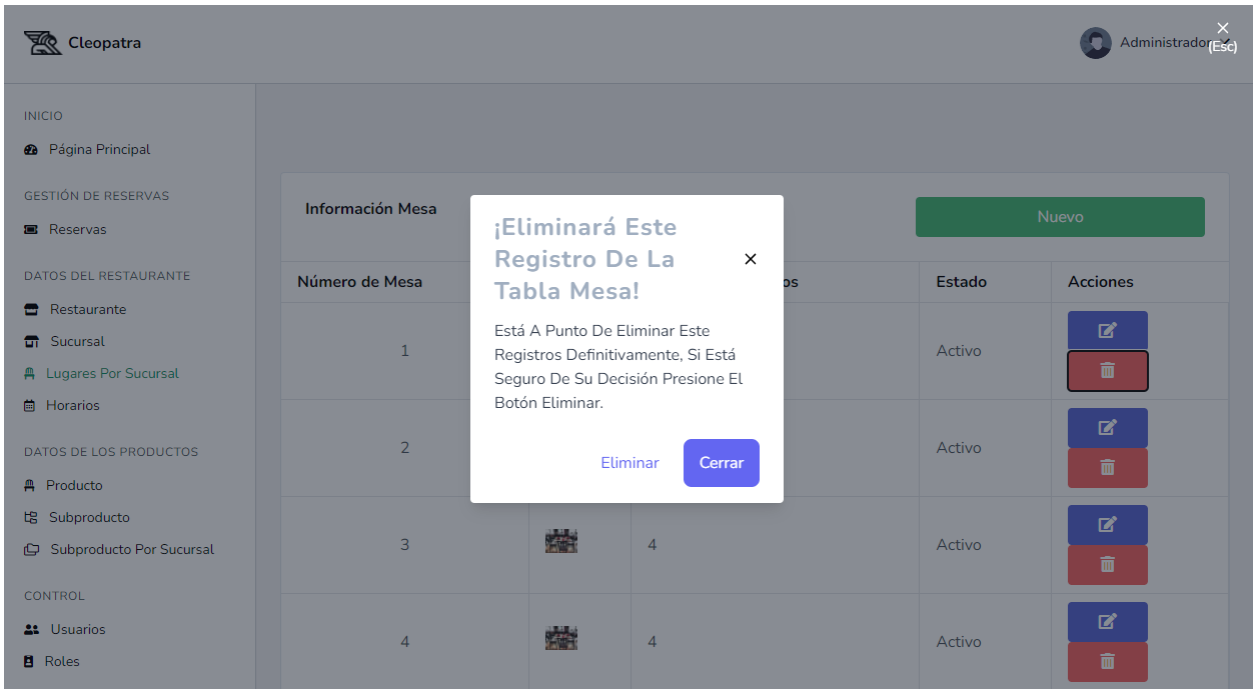
*Modal mostrado al momento de dar clic sobre editar una mesa*




*Nota*. La ventana emergente muestra la información previamente cargada de la mesa a la que se le dio clic.

**Figura 40**

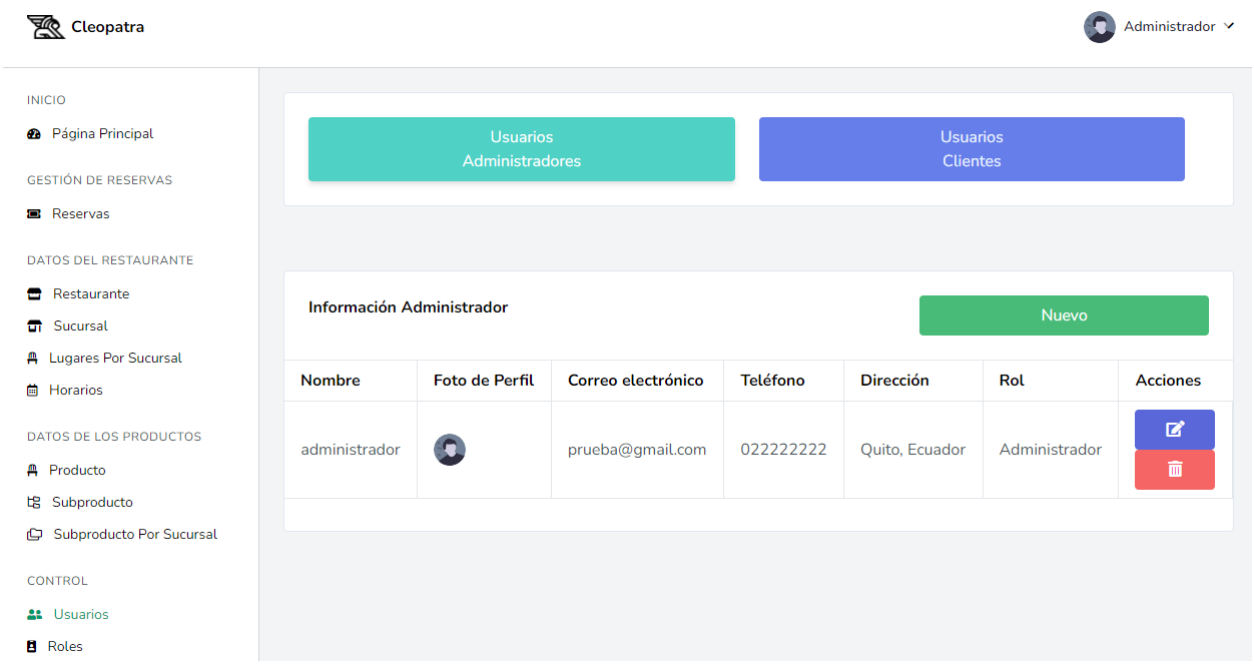
*Modal mostrado al momento de intentar eliminar una mesa*




**3.3.4.3. Gestión de Usuarios Administradores.** Este apartado reutiliza las interfaces anteriores para su CRUD, pero tiene un añadido que es separar los usuarios administradores de los usuarios clientes.

**Figura 41**

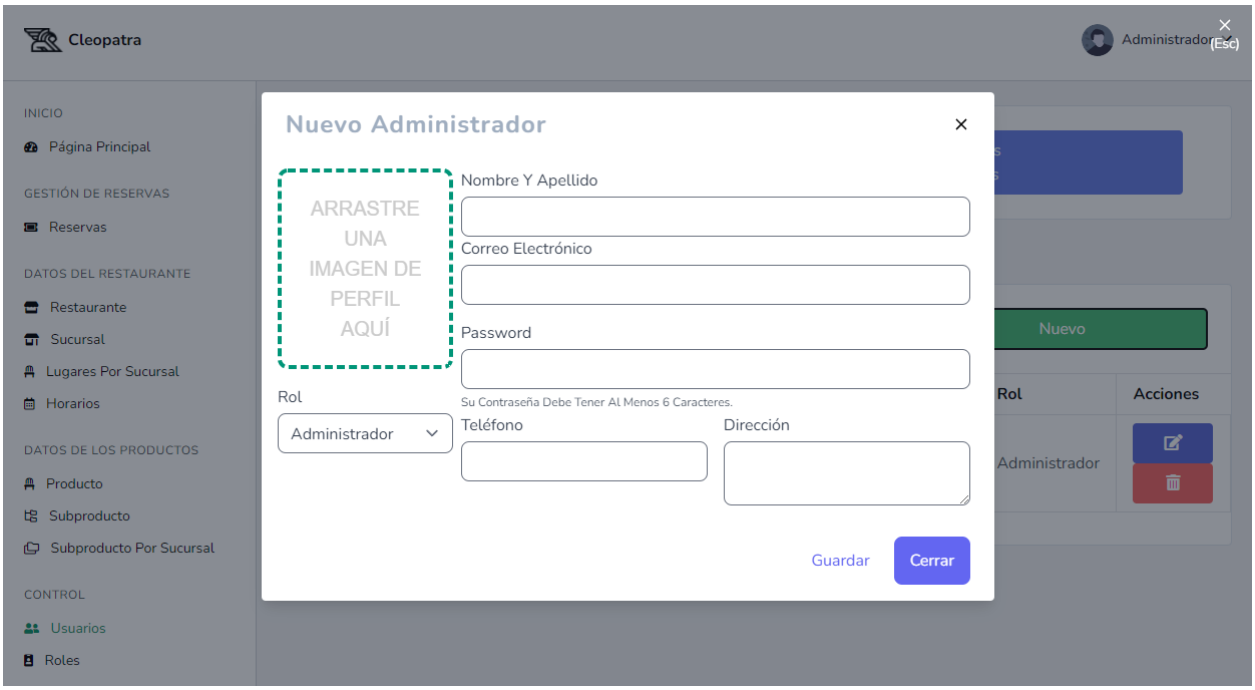
*Página principal de la opción "Usuarios" como administrador*




*Nota*. Esta interfaz es mostrada al dar clic sobre el enlace Usuarios en el menú lateral.

**Figura 42**

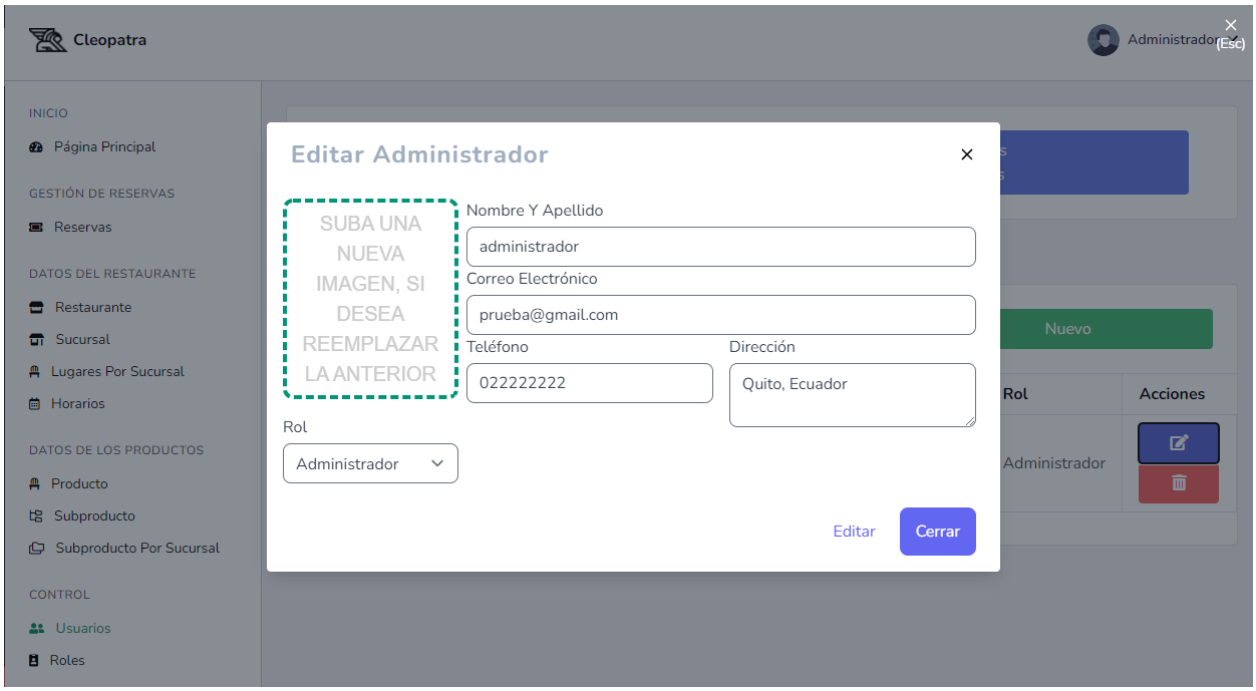
*Modal mostrado al dar clic en la opción de Nuevo*



*Nota*. La ventana emergente mostrada permite ingresar diferente tipo de información del usuario como su nombres y apellidos, teléfono, dirección y el rol asignado. Además de estas opciones se puede agregar las credenciales con las que el usuario entrará al sistema. Al igual que otros CRUD permite ingresar una imagen, pero está servirá como imagen de perfil del usuario.

**Figura 43**

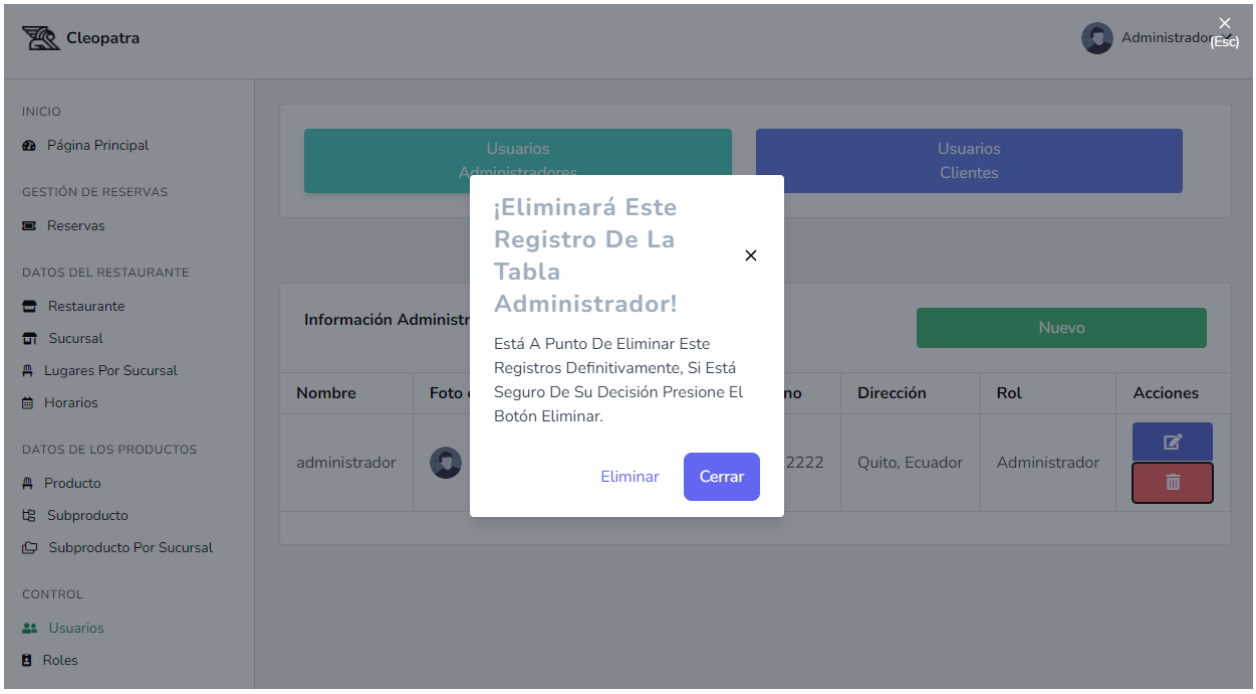
*Modal mostrado al momento de dar clic sobre editar un usuario administrador*




*Nota*. La ventana emergente muestra la información previamente cargada del usuario al que se le dio clic

**Figura 44**

*Modal mostrado al momento de intentar eliminar un usuario administrador*




### *3.3.5. Sprint Review*

Se ha cumplido la totalidad de los avances planteados durante el Sprint II. Aunque, existen las siguientes observaciones:

- Aumentar una barra con buscador en el menú horizontal de administrador.
- Aumentar un pie de página a la vista de administradores y de cliente.
- Revisar el apartado de usabilidad de la página.
- Realizar pruebas de carga de datos en el aplicativo.

### *3.3.6. Sprint Retrospective*

Este es el último evento del *sprint* donde de la mano de un profesional se da el visto bueno al aplicativo.

**Tabla 9**

*Observaciones del experto en el tema*

| Nombre del Experto | Correo Electrónico | Observaciones |
|---|---|---|
| Damián Nicolalde | danicolalde@puce.edu.ec | Tomar en cuenta los principios de usabilidad. Por ejemplo, aumentar *footer* en la página de administrador. |
| | | Aumentar barra de buscador en el apartado de administrador |
| | | Realizar pruebas del sistema. Por ejemplo, cargar cientos de datos para comprobar el funcionamiento del aplicativo. |

## 3.4. Sprint III

### *3.4.1. Sprint Planning*

Durante este *sprint* se tomará en cuenta las observaciones del *sprint* anterior y además se completarán nuevos requerimientos de acuerdo con la planificación.

### *3.4.2. Sprint Backlog*

Este sprint deberá completar las siguientes historias de usuario definidas en el *product backlog.*

**Tabla 10**

*Backlog del Sprint III*

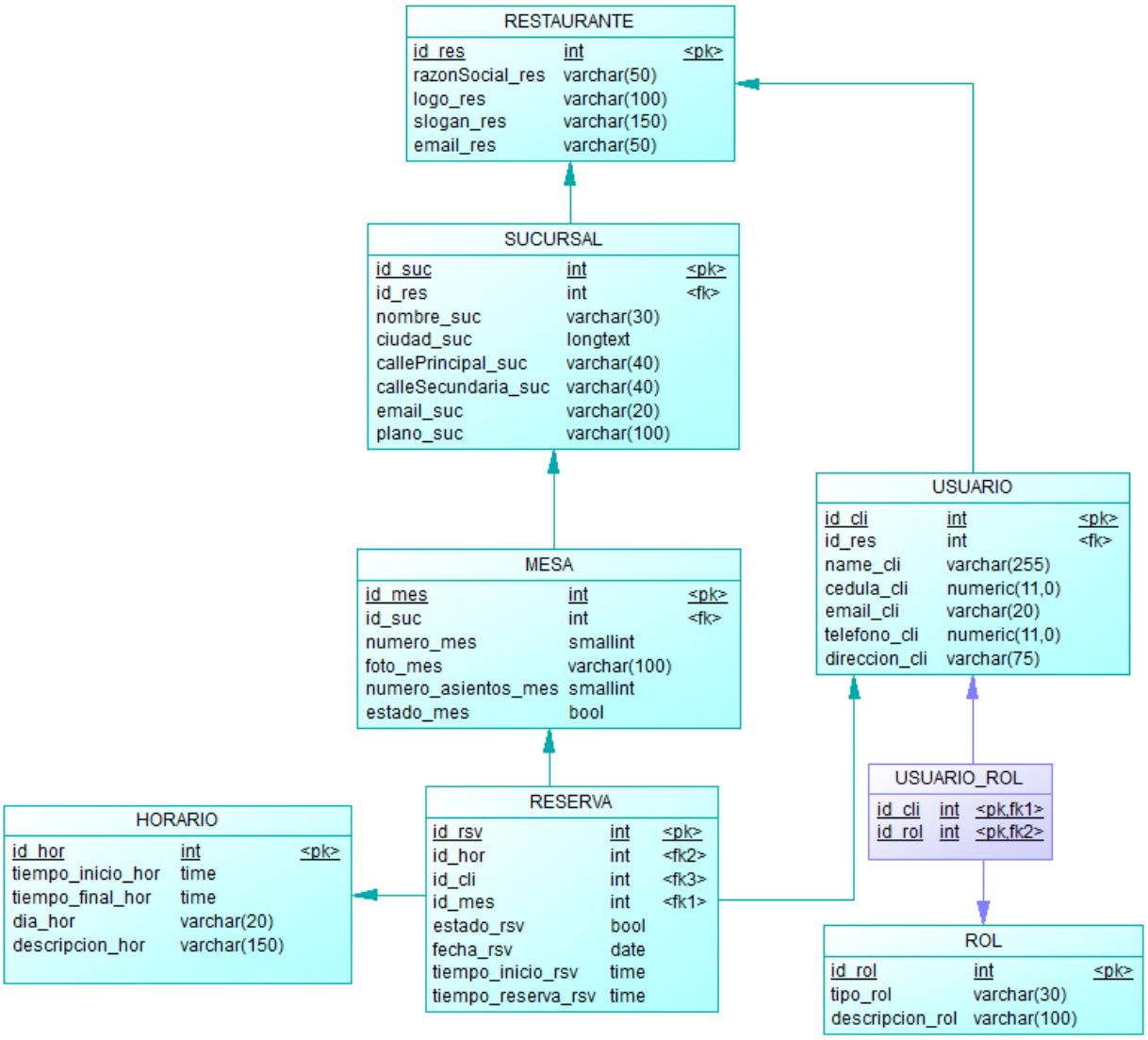
| Id. | Requerimiento | Responsable |
|---|---|---|
| **10** | Gestión de Usuarios Clientes | Erick Rea |
| **11** | Reservas | Erick Rea |
| **12** | Horario | Erick Rea |

### *3.4.3. Diseño*

Para cumplir los requerimientos de este *sprint* a la base de datos se le añadirá dos nuevas tablas. Estas tablas son “Horario” y “Reserva”. La primera tabla servirá para almacenar los días y las horas en las que se podrá realizar una reserva. La segunda tabla servirá para que un usuario pueda hacer la reserva de una mesa o más mesas en el día, hora y tiempo desee, esta reserva pasará por diferentes estados como se definió en los requerimientos.

**Figura 45**

*Tercera versión de la base de datos*



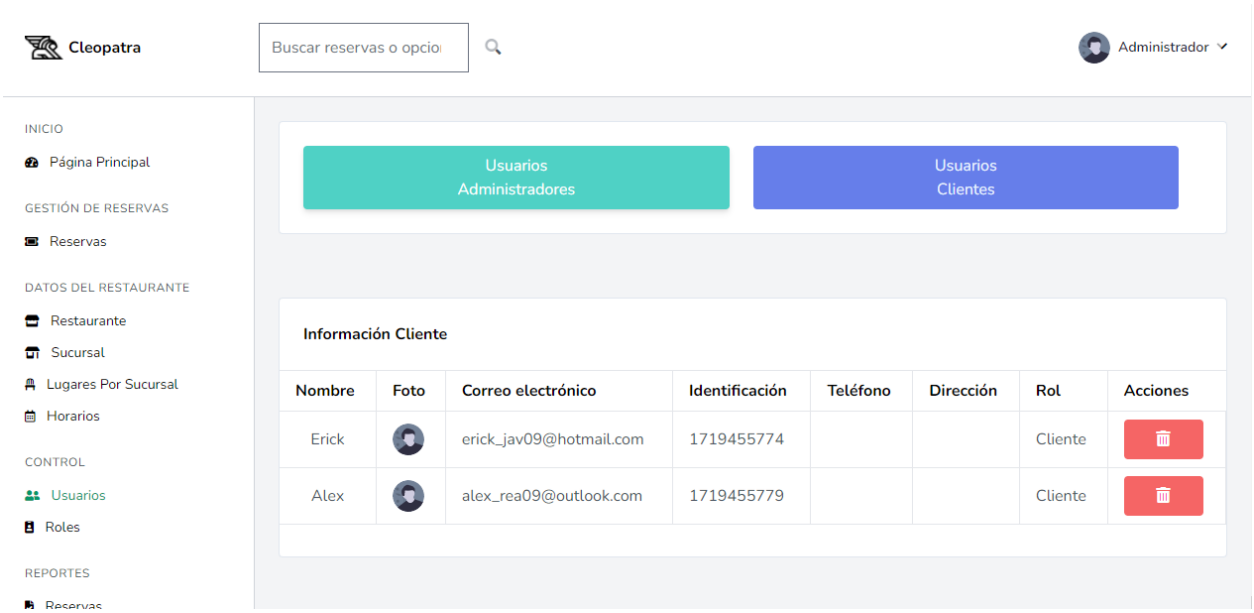

*Nota*. En comparación a la segunda versión, se ha aumentado las tablas “Horario” y “Reserva”.

### *3.4.4. Prototipo*

**3.4.4.1. Gestión de Usuarios Clientes.** La gestión de usuarios clientes es vista por los administradores, en este caso, solamente se podrá observar este tipo de usuarios y eliminarlos. No se podrá editar la información de los mismos, ya que se toma en cuenta que este tipo de información es personal y solamente el mismo usuario podrá editar su propia información.

**Figura 46**

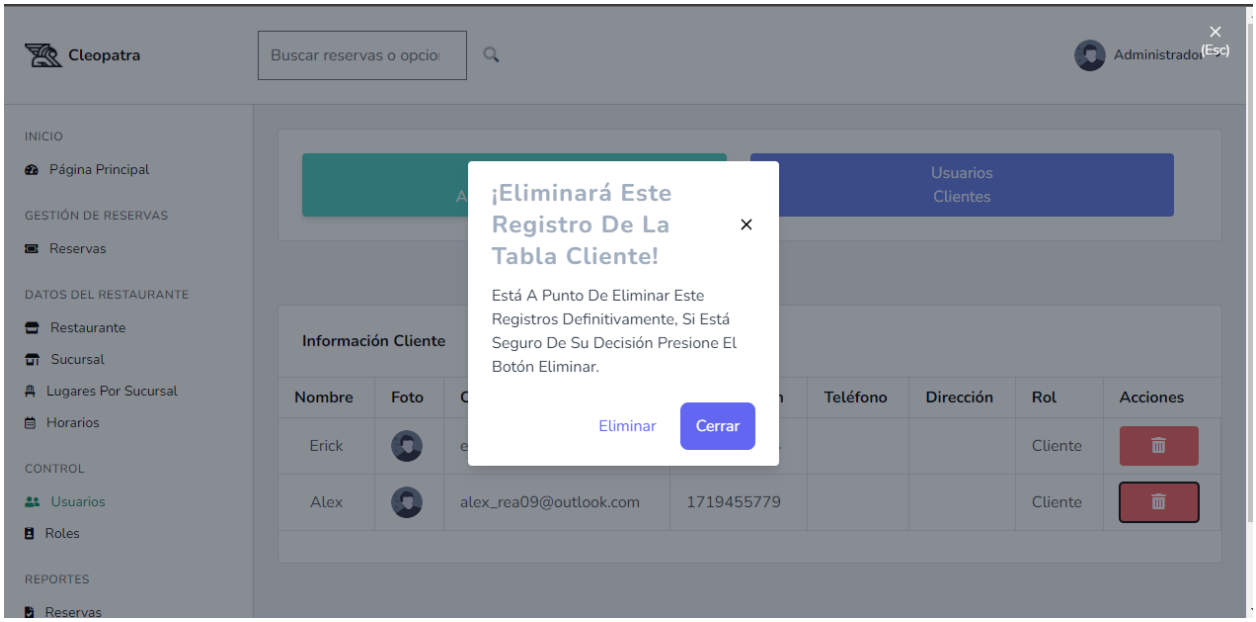
*Página principal de la opción "Usuarios Clientes" como administrador*




*Nota.* Se puede acceder a esta vista mediante el botón "Usuarios Clientes" dentro de la opción "Usuarios" del menú lateral.

**Figura 47**

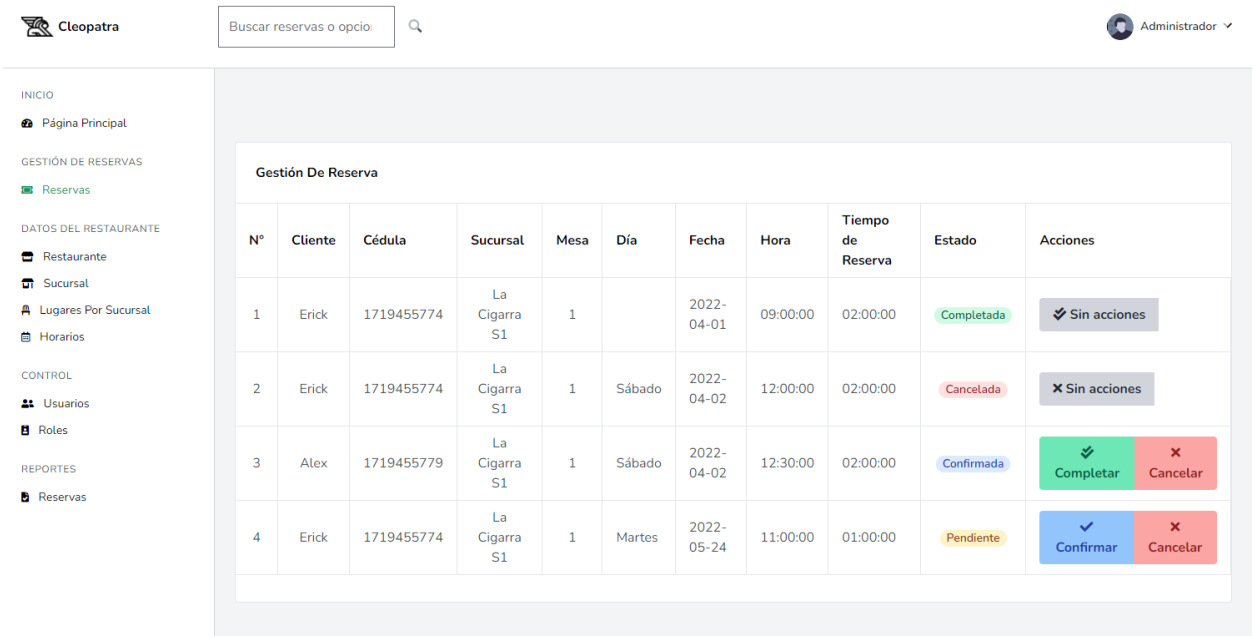
*Modal mostrado al momento de intentar eliminar un usuario cliente*




**3.4.4.2. Reservas.** Tomando en cuenta el alcance de este sprint, se creó la vista de la lista de reservas del sistema para un restaurante y sus sucursales. En este caso, se ingresaron casos de prueba de reservas con estados Completada, Cancelada, Confirmada y Pendiente.

**Figura 48**

*Página principal de la opción "Reservas" como administrador*




*Nota*. Los estados mostrados fueron creados en base a los requerimientos del aplicativo. Estos serán gestionados en el siguiente *sprint.*

**3.4.4.3. Horario.** Para realizar el CRUD de los horarios se reutilizó la misma interfaz gráfica que fue usada previamente. Se puede crear, ver, editar y eliminar información de los horarios.

**Figura 49**

*Página principal de la opción "Horarios" como administrador*

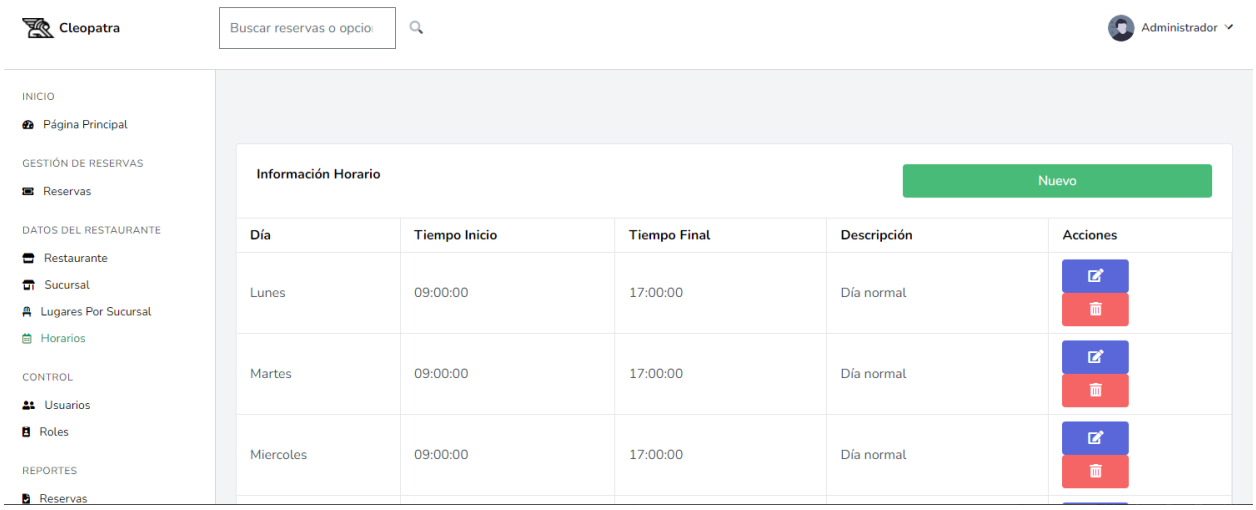



**Figura 50**

*Modal mostrado al dar clic en la opción de Nuevo*

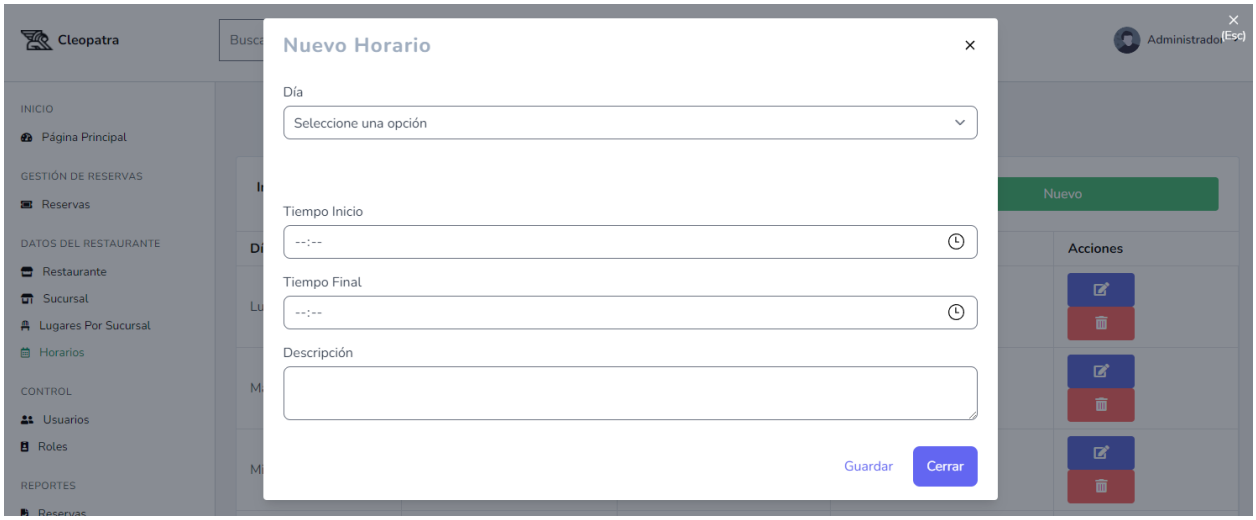
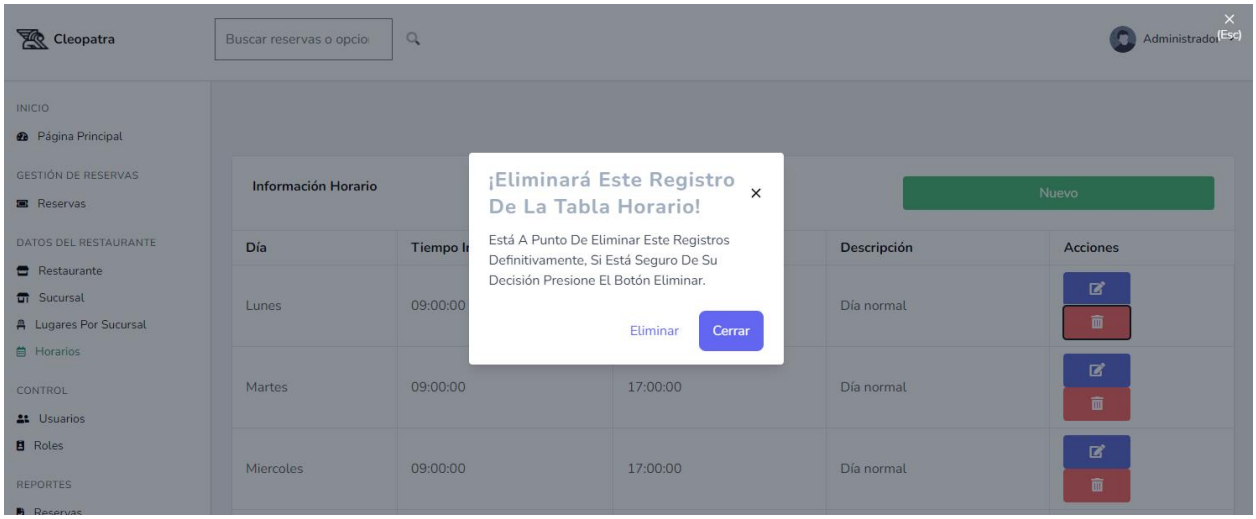



**Figura 51**

*Modal mostrado al momento de dar clic sobre editar un horario*


**Figura 52**

*Modal mostrado al momento de intentar eliminar un horario*




### *3.4.5. Sprint Review*

Se ha cumplido la totalidad de los avances planteados durante el Sprint III. Se tiene las siguientes observaciones:

- Aumentar un buscador por cada tabla. Por ejemplo, la tabla de reservas debería poder buscar una reserva específica por su identificador, el día o por información del cliente.

#### *3.4.6. Sprint Retrospective*

Este es el último evento del *sprint* donde de la mano de un profesional se da el visto bueno al aplicativo.

**Tabla 11**

*Observaciones del experto en el tema*

| Nombre del Experto | Correo Electrónico | Observaciones |
|---|---|---|
| Damián Nicolalde | danicolalde@puce.edu.ec | Sin observaciones. |

### 3.5. Sprint IV

#### *3.5.1. Sprint Planning*

Para completar este *sprint* es necesario que se haya completado el requerimiento “REQ11 Reservas”, ya que las características que se desarrollarán durante este ciclo gestionarán las reservas según el tipo de usuario que las manipule. A su vez, las reservas realizadas tendrán un apartado en cuál se exportará y observará información de estas.

#### *3.5.2. Sprint Backlog*

Durante este ciclo se desarrollarán los siguientes requerimientos definidos en el *product backlog.*

**Tabla 12**

*Backlog del Sprint IV*

| Id. | Requerimiento | Responsable |
|---|---|---|
| **13** | Gestión de Reservas para Usuarios Administradores | Erick Rea |
| **14** | Gestión de Reservas para Usuarios Clientes | Erick Rea |
| **15** | Reportería | Erick Rea |

### *3.5.3. Diseño*

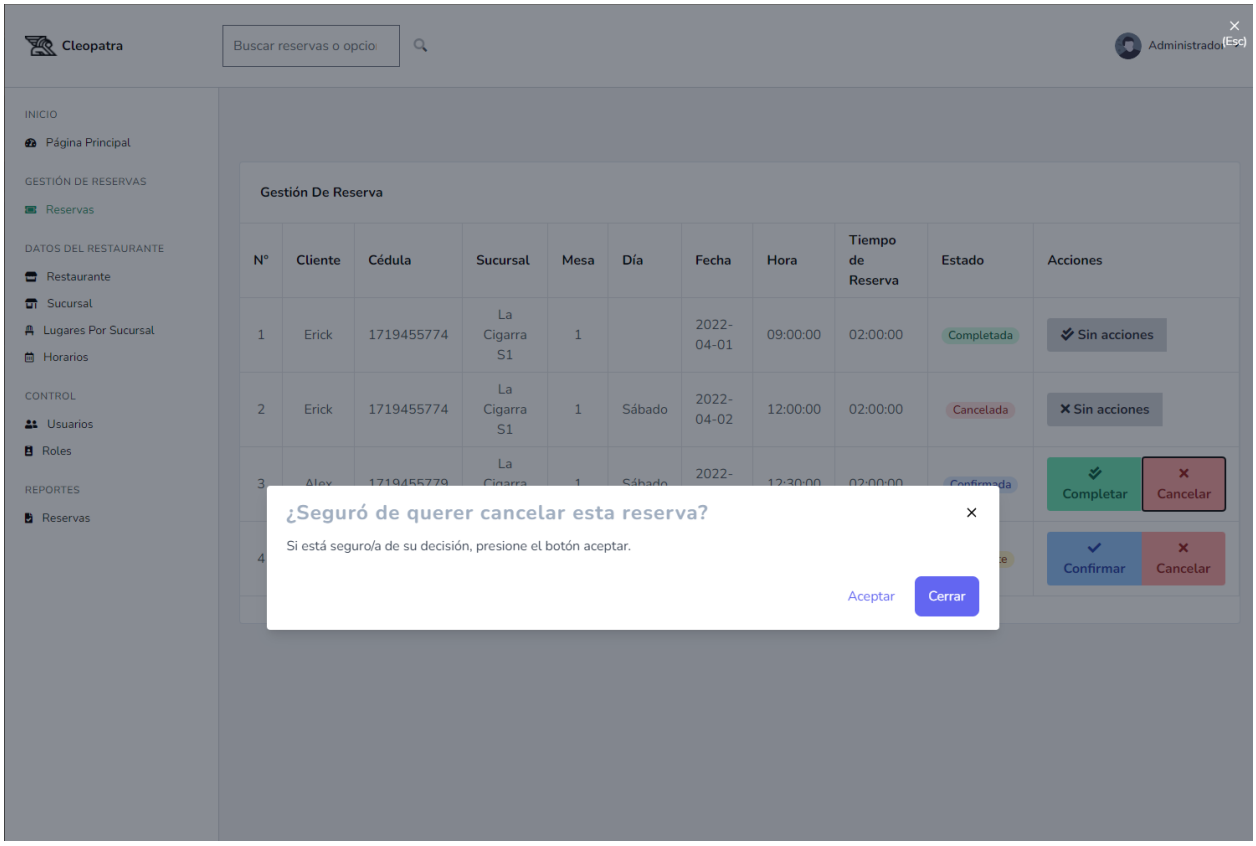
El diseño de la base de datos realizado en el *Sprint* III será preservado, dado que cumple con lo necesario para poder realizar lo planeado durante este ciclo.

### *3.5.4. Prototipo*

**3.5.4.1. Gestión de Reservas para Usuarios Administradores.** Este requerimiento es un complemento al REQ11, ya que el usuario administrador será capaz de poder cambiar los estados de las reservas.

**Figura 53**

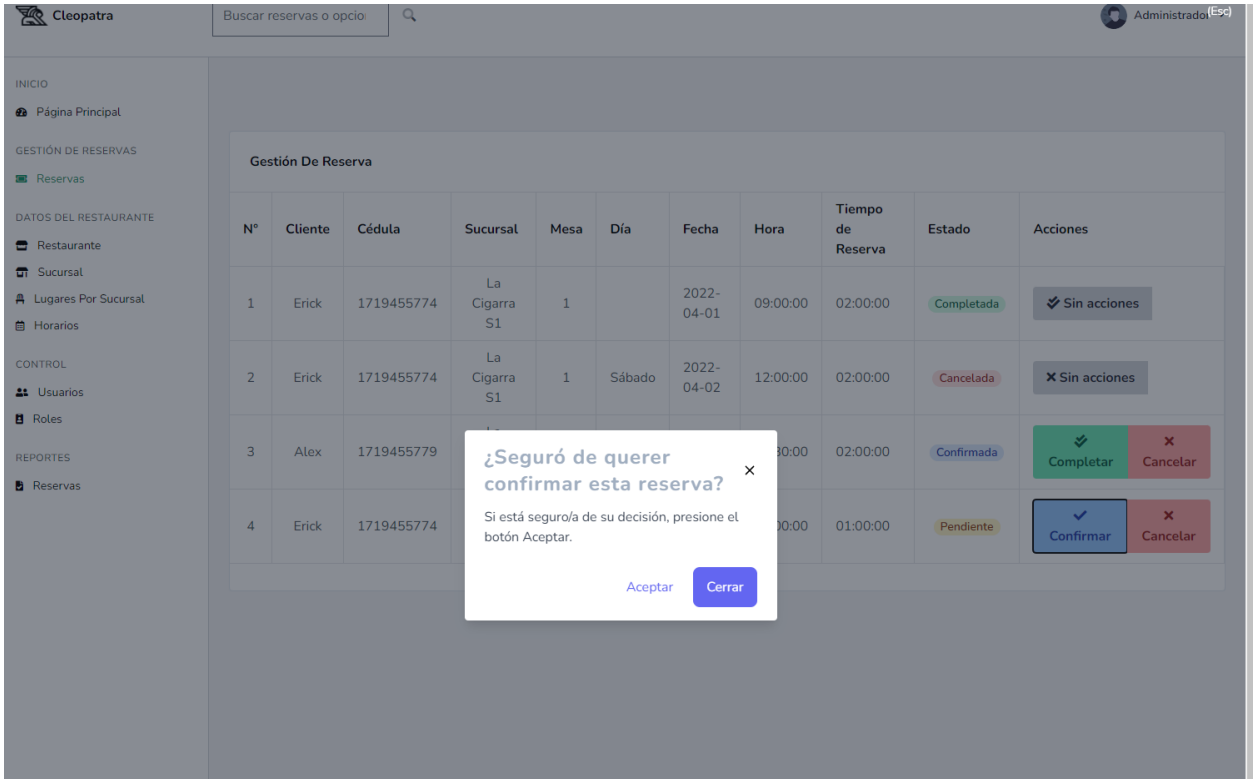
*Modal mostrado al intentar cancelar una reserva*




*Nota*. La venta emergente aparece al dar clic en la opción de Cancelar. Esta opción aparece en reservas que tengan el estado de Confirmada o Pendiente.

**Figura 54**

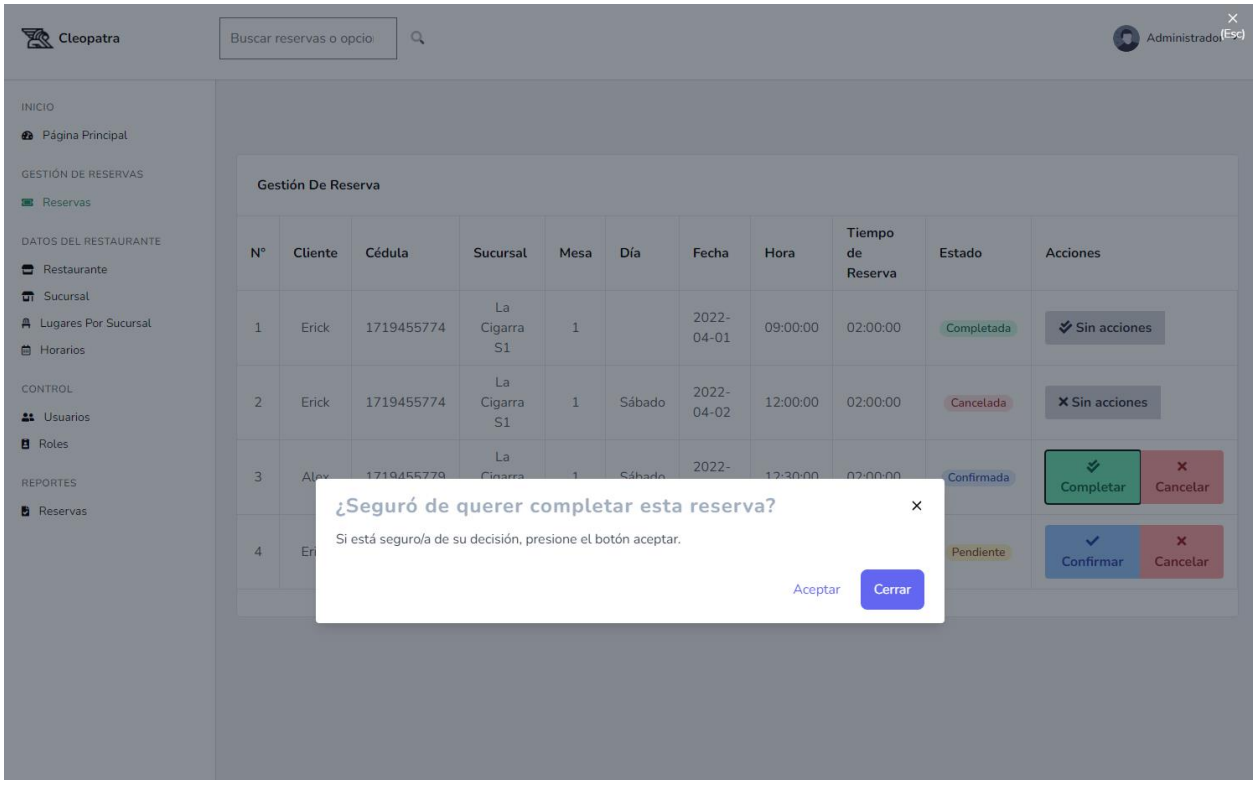
*Modal mostrado al intentar confirmar una reserva*




*Nota*. La opción de Confirmar solo aparece en reservas Pendientes.

**Figura 55**

*Modal mostrado al intentar completar una reserva*



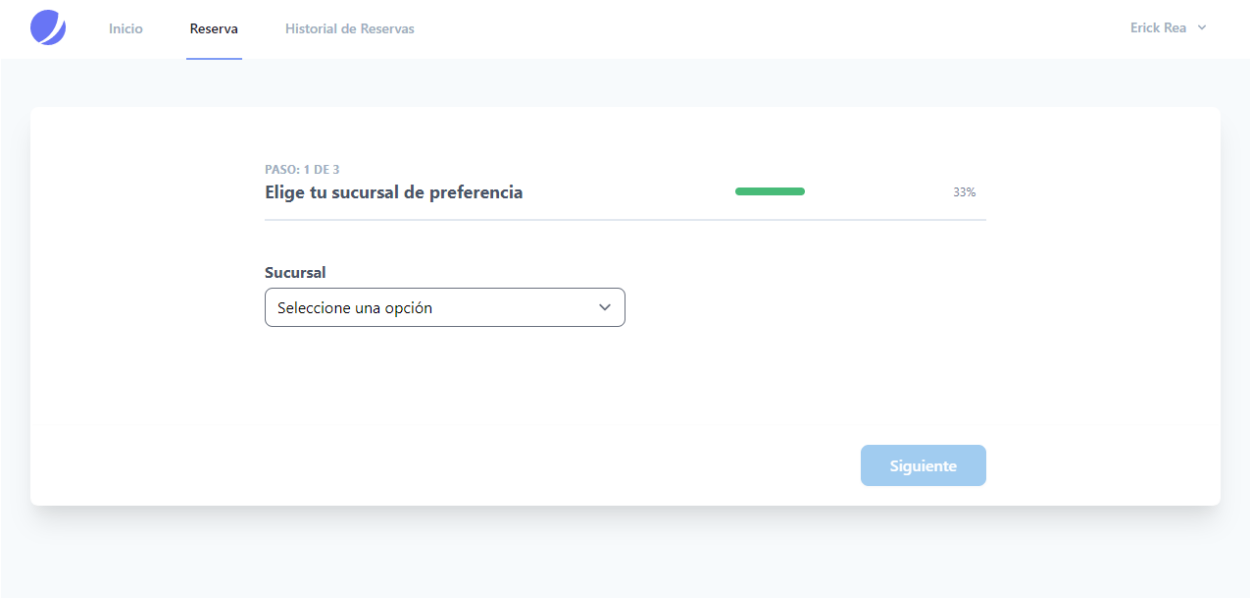

*Nota.* La opción Completar aparece solamente en reservas con estado Confirmada.

En caso de que las reservas estén completas o canceladas, ya no habrá más opciones para modificar el estado de estas.

**3.5.4.2. Gestión de Reservas para Usuarios Clientes.** Los usuarios clientes son capaces de crear una o más reservas, según la sucursal que prefieran. La reserva se realiza en tres pasos mostrados en la interfaz de este tipo de usuario. También, el cliente puede gestionar sus propias reservas, dependiendo del estado de estas.

**Figura 56**

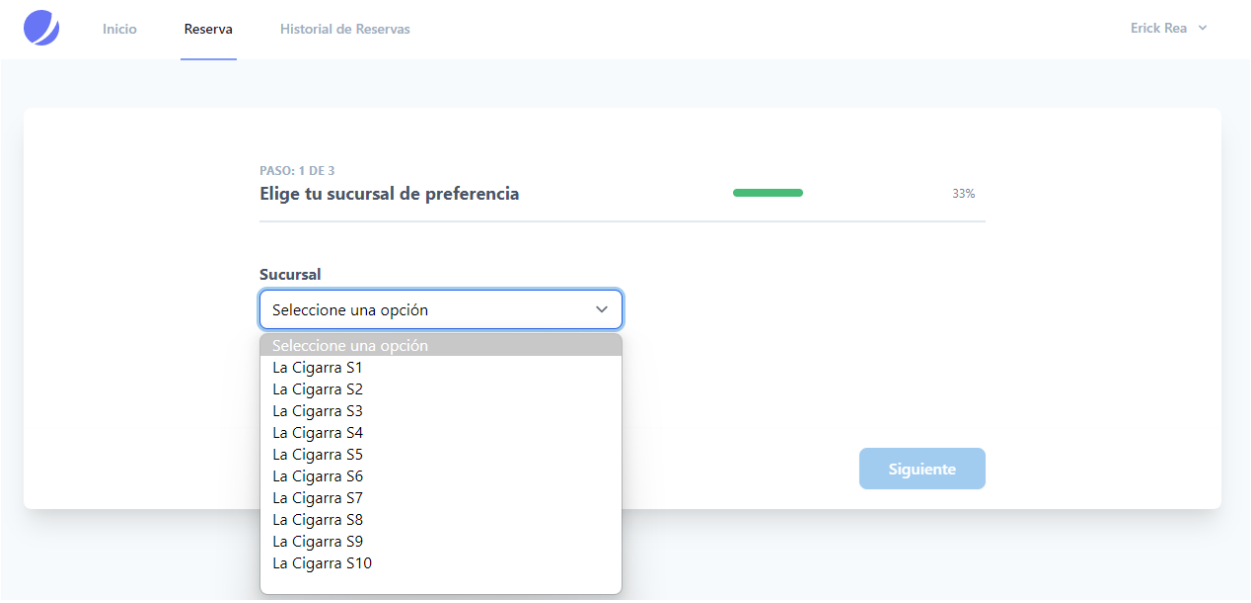
*Paso 1 de 3 para realizar una reserva*




*Nota.* La vista mostrada aparece al escoger la opción “Reserva” en el menú de clientes.

**Figura 57**

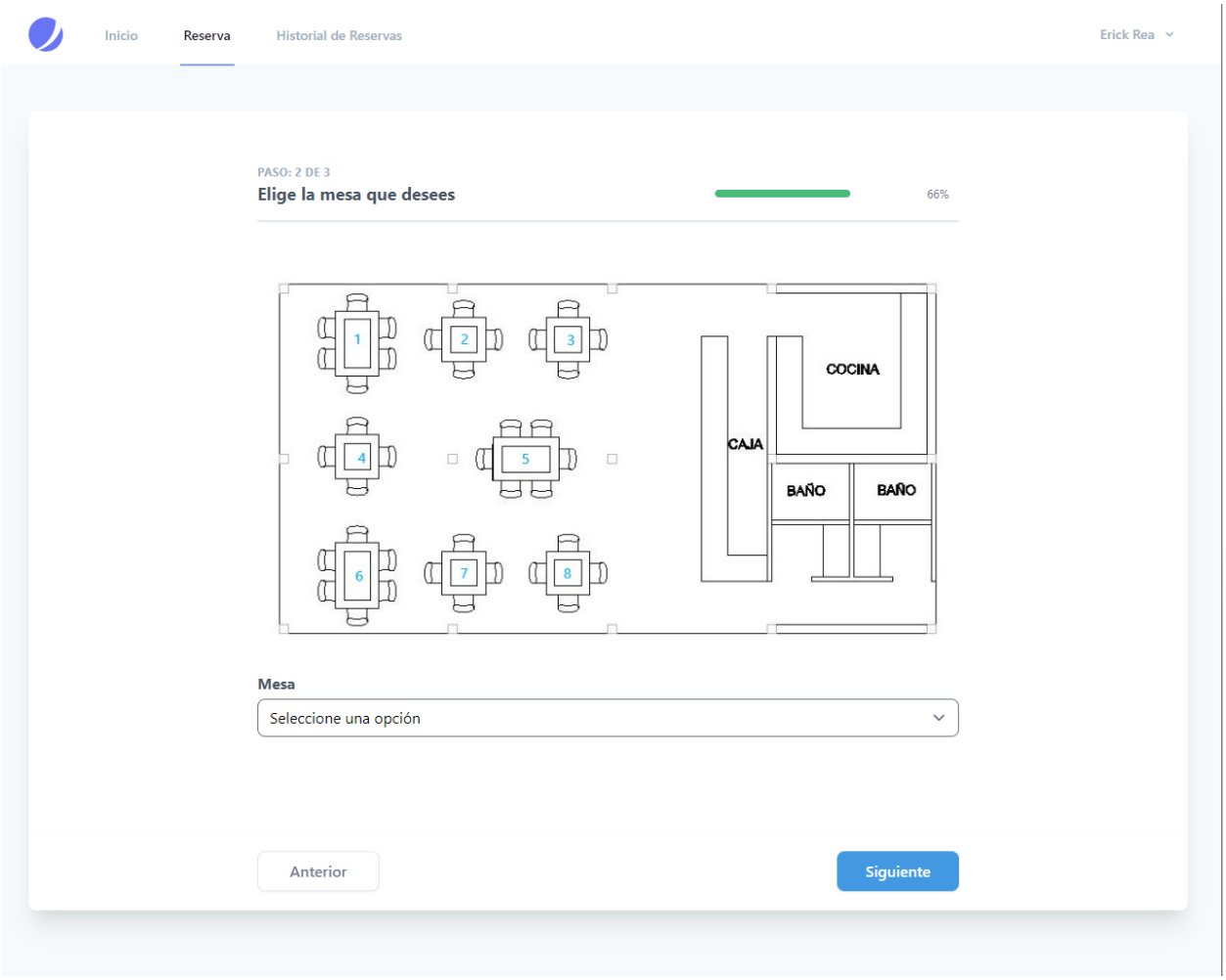
*Selección de una sucursal*




*Nota*. El usuario puede escoger una de las sucursales que el restaurante ha añadido.

**Figura 58**

*Elegir la mesa de la sucursal*




*Nota*. Este el siguiente paso después de haber escogido una sucursal y dar clic en el botón de siguiente. La imagen mostrada es el plano añadido de la sucursal.

**Figura 59**

*Escoger la mesa de la sucursal*

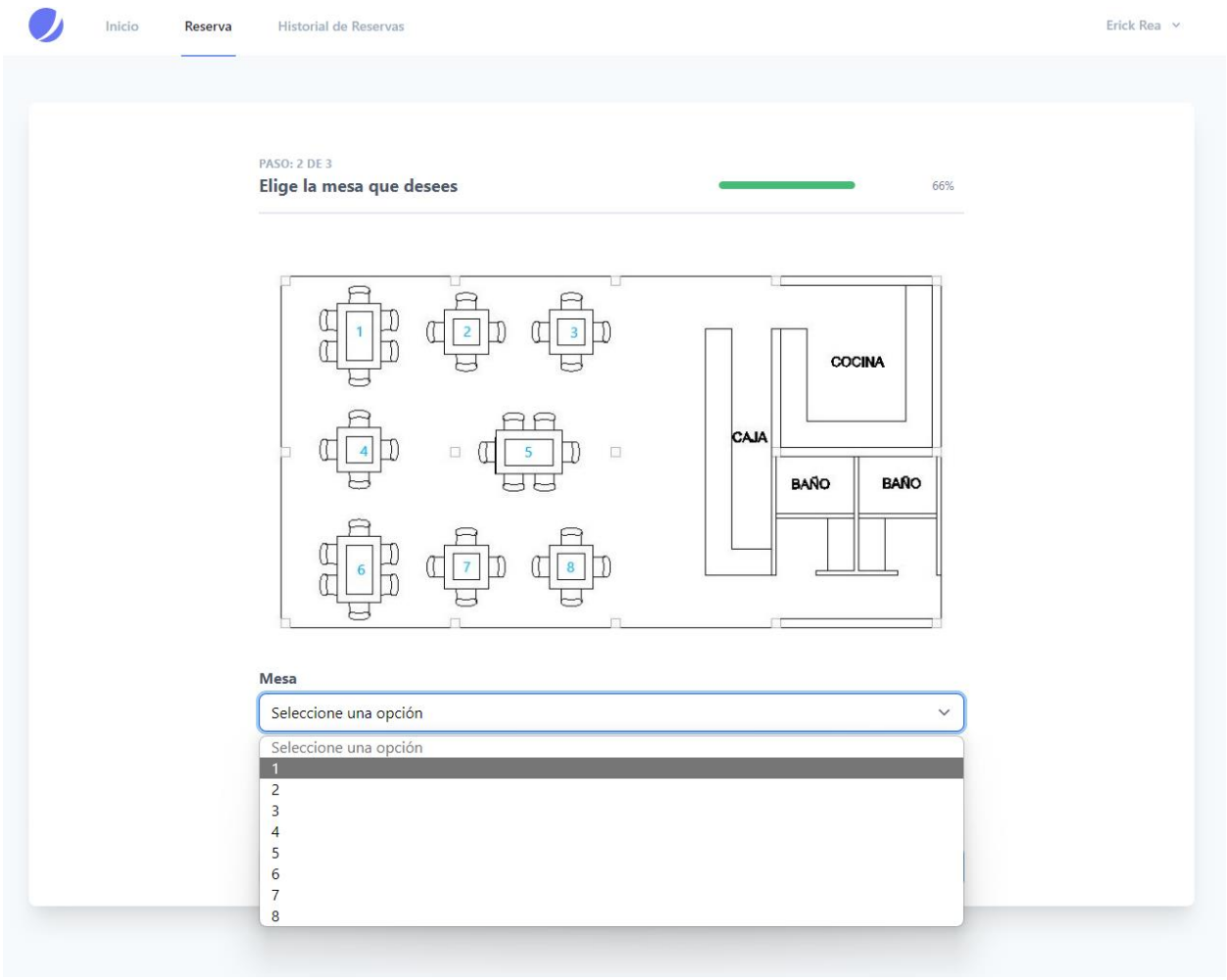
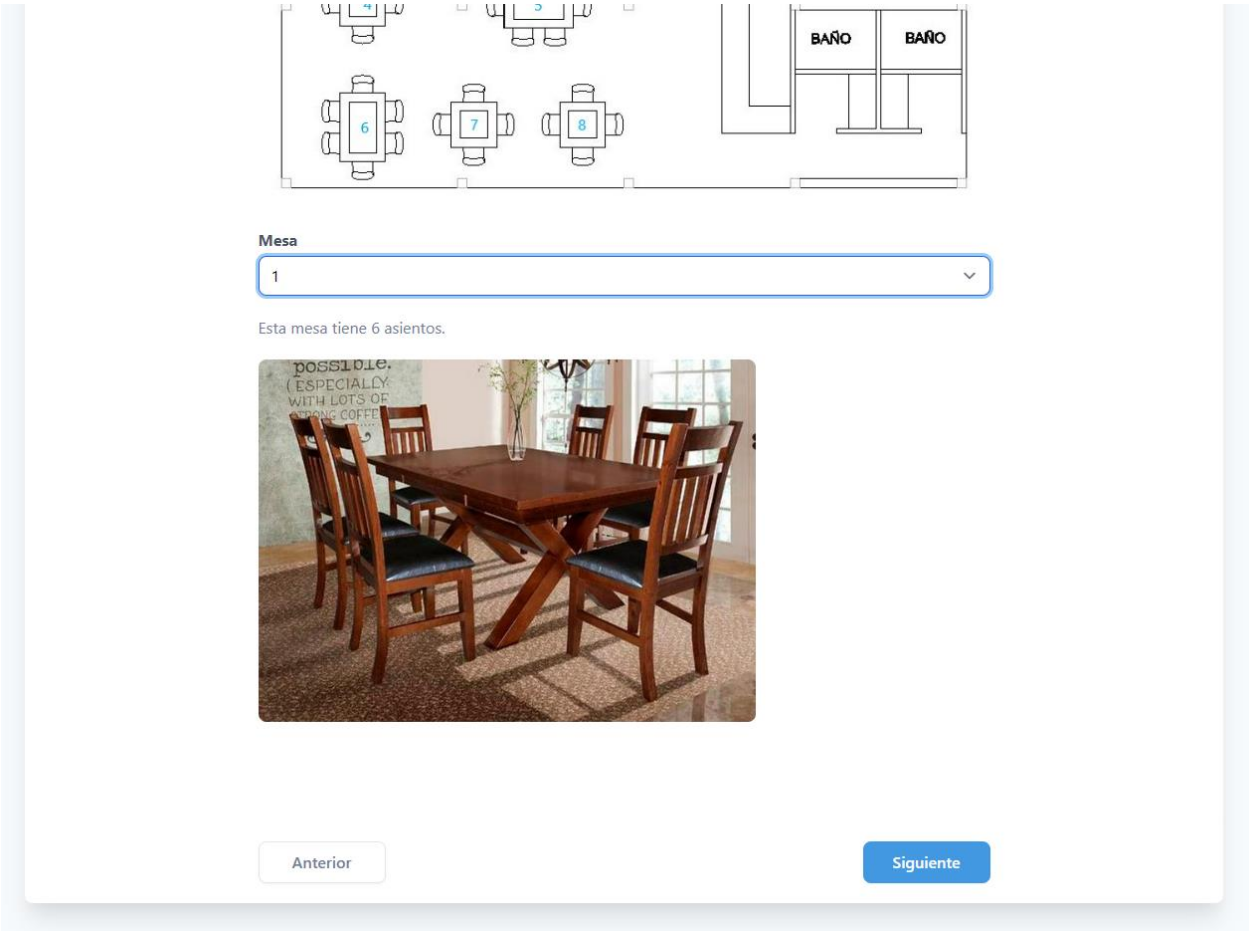



**Figura 60**

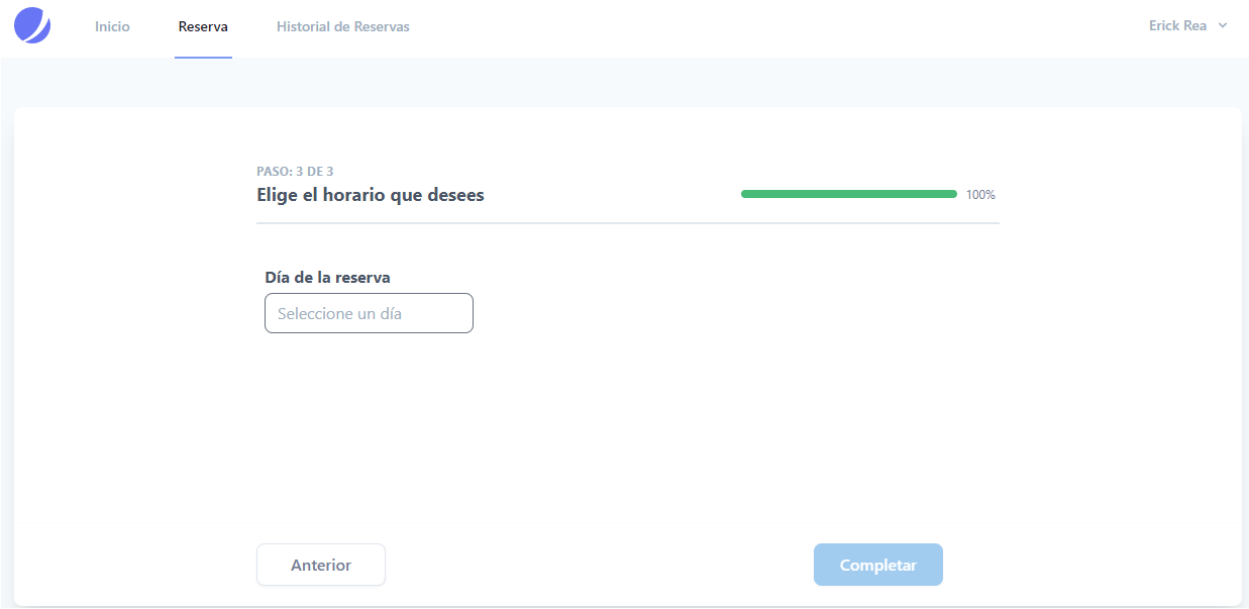
*Imagen mostrada después de escoger una mesa*




*Nota*. La imagen mostrada es parte de la información que ha sido guarda para la mesa de la sucursal.

**Figura 61**

*Escoger horario de reservación*

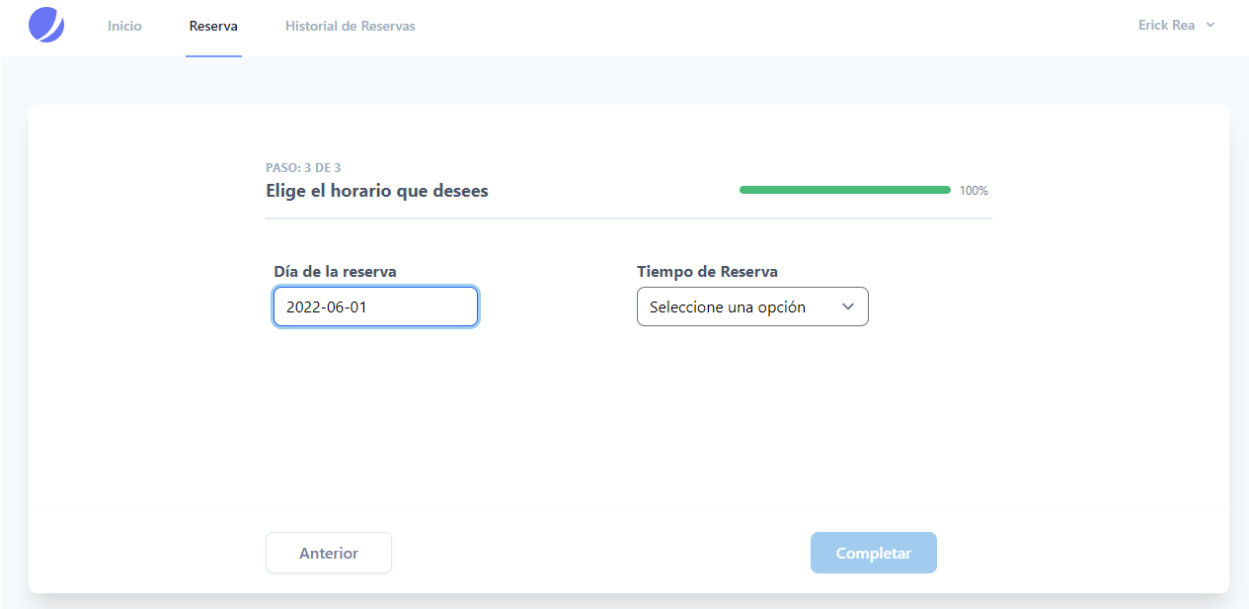



*Nota*. Al completar el paso anterior se muestra la vista para poder escoger el día de la reserva.

Una reserva se puede realizar con un día de antelación.

**Figura 62**

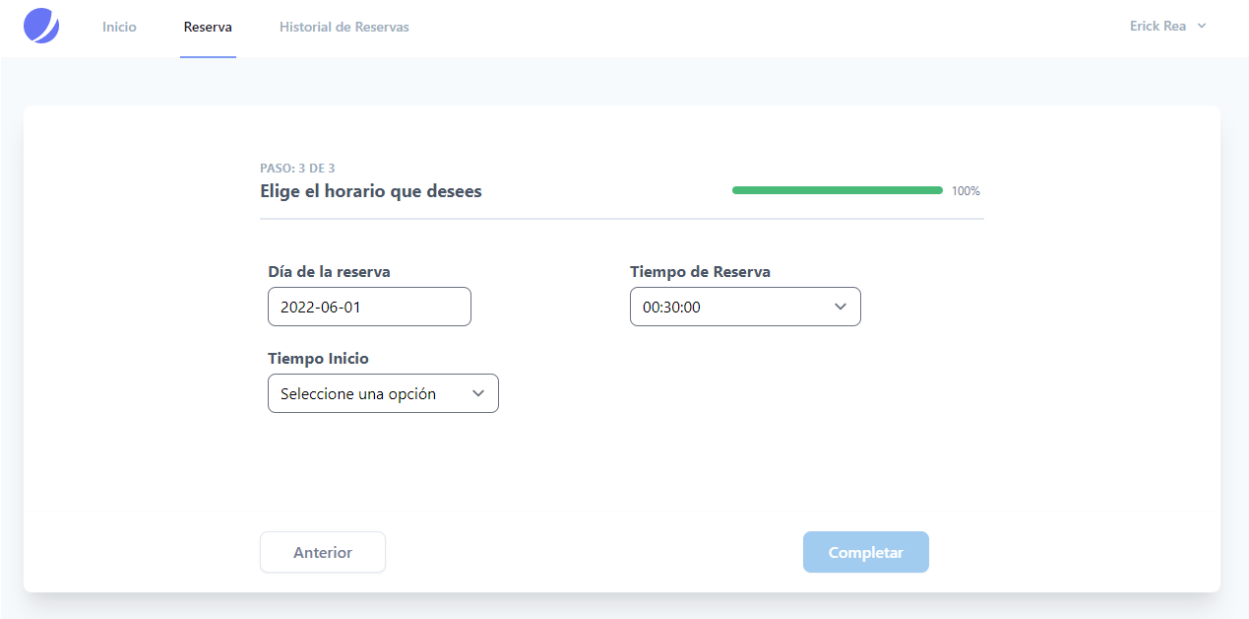
*Escoger tiempo de reserva*




*Nota*. Al escoger un día, se despliega la opción de tiempo de reserva.

**Figura 63**

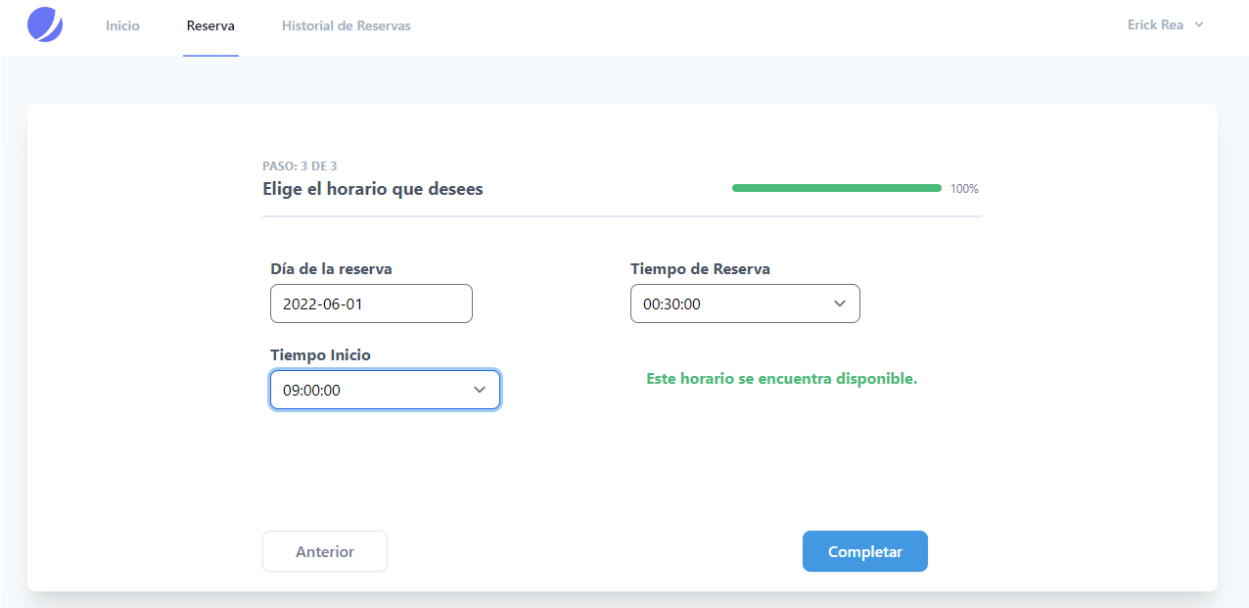
*Escoger el tiempo de inicio de la reserva*




*Nota*. Después que el cliente ha escogido cuanto tiempo durará su reserva, podrá escoger en que horario se acercará. Los tiempos mostrados van acorde al horario que se haya ingresado en el sistema.

**Figura 64**

*Horario disponible para reserva*




*Nota*. En caso de que la reserva no haya sido tomada, se mostrará un mensaje señalando que el horario está disponible y el botón de completar se activará.

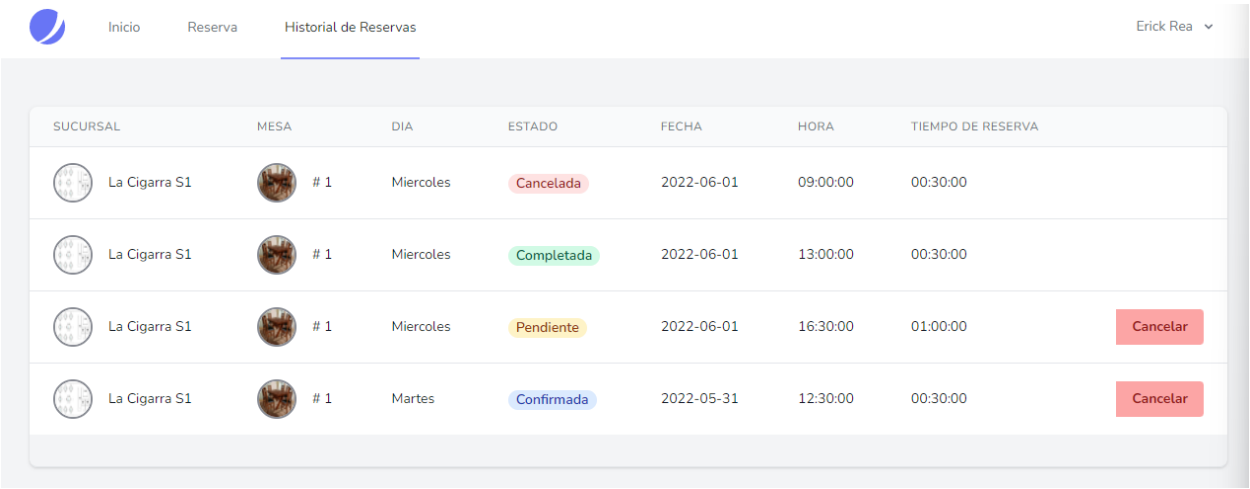
**Figura 65**

*Reserva exitosa*


*Nota*. Mensaje mostrado si la reserva no ha tenido errores durante su creación.

**Figura 66**

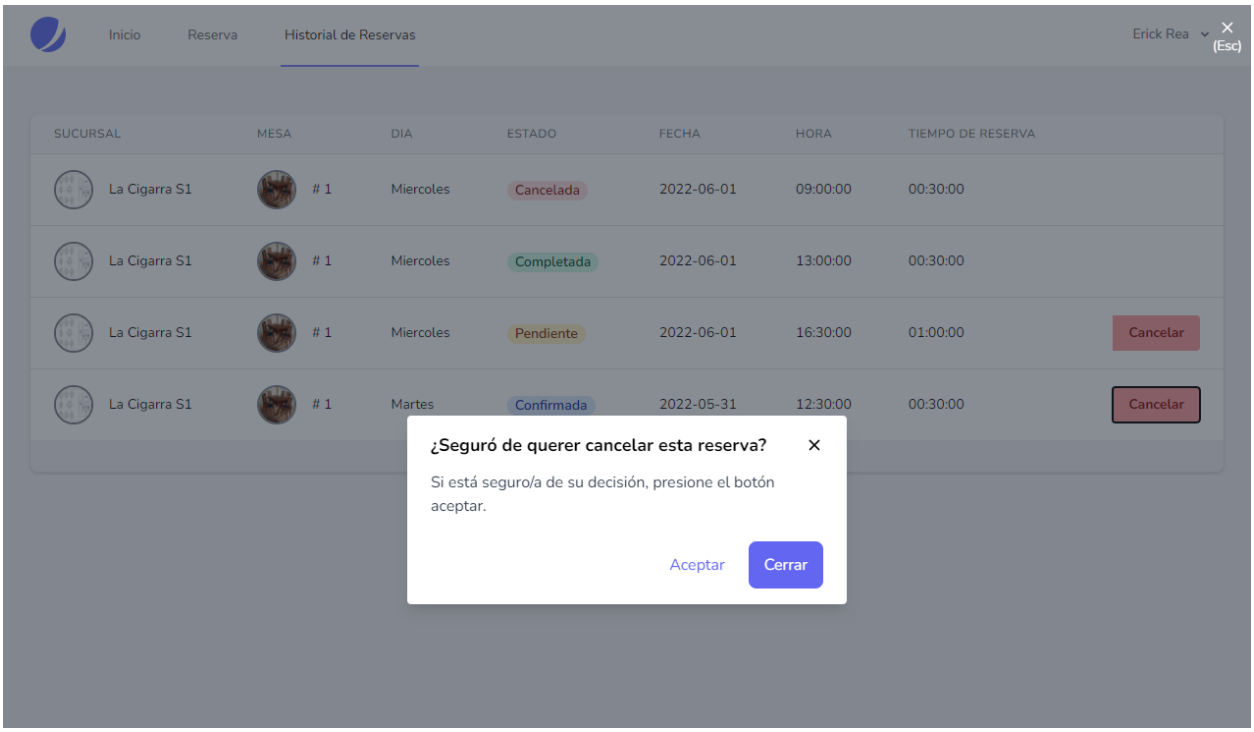
*Historial de reservas del cliente*




*Nota*. Historial de reservas que ha tenido un cliente. Los usuarios clientes solo podrán cancelar las reservas pendientes y/o confirmadas.

**Figura 67**

*Modal mostrado para cancelar reserva*




*Nota*. Ventana emergente mostrada al dar clic sobre el botón “Cancelar” de una reserva.

**3.5.4.3. Reportería.** Este apartado está divido en dos partes que son observadas por un usuario administrador. La primera parte abarca la creación de un archivo hoja de cálculo en un rango de fechas con las reservas realizadas. La otra mitad es un gráfico de barras, con los números de reservas que ha tenido el sistema con sus diferentes estados.

**Figura 68**

*Vista de reportería, exportar hoja de cálculo*

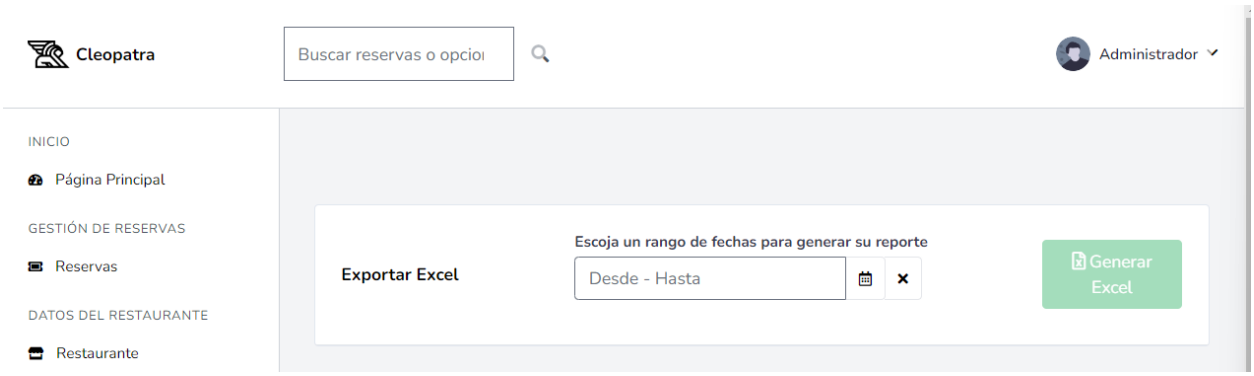



**Figura 69**

*Elección de rango de fechas para generar reporte*

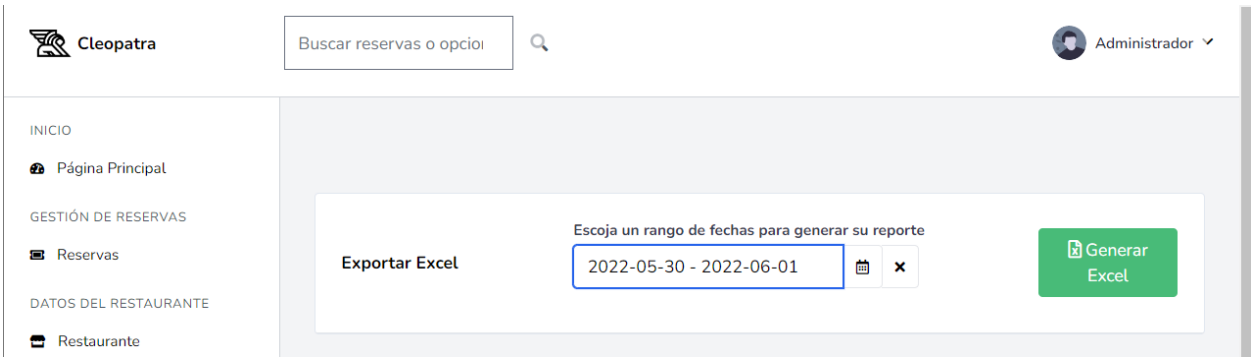


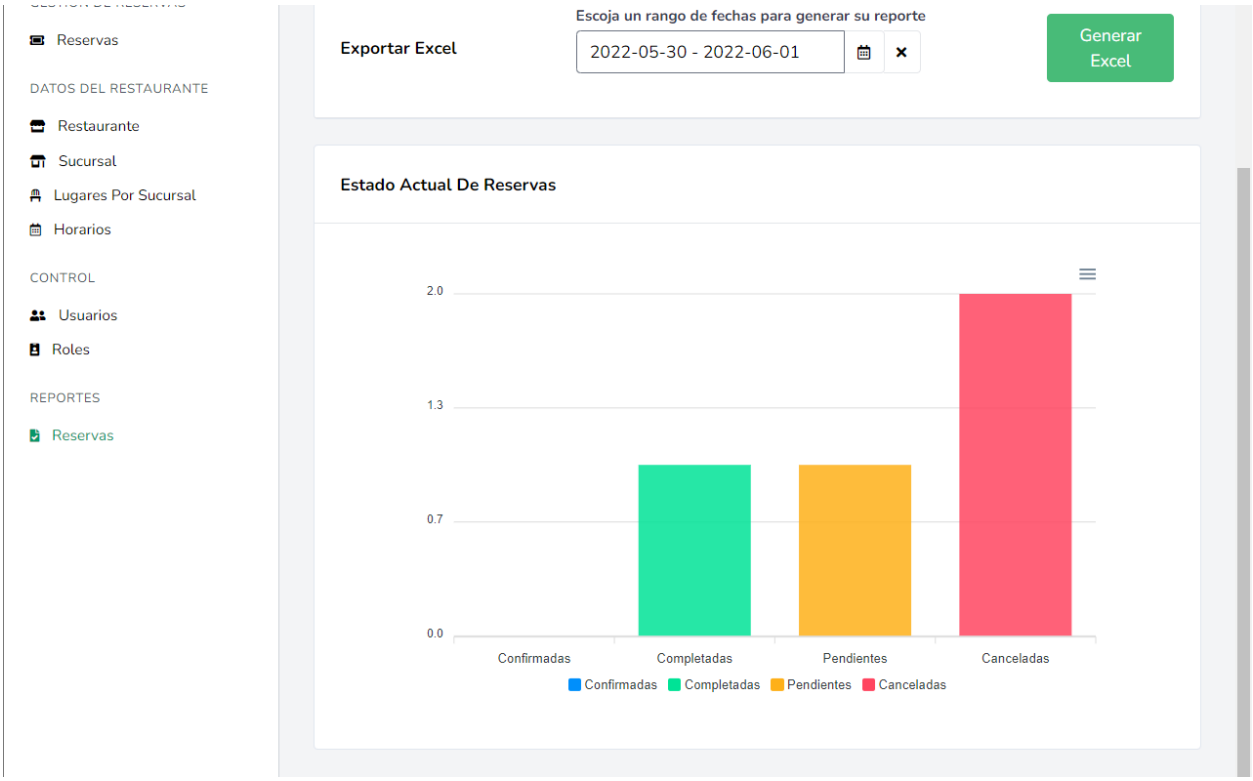

**Figura 70**

*Ejemplo de hoja de cálculo generada*

| | A | B | C | D | E | F | G | H | I | J | K |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | **Id** | **Sucursal** | **Mesa** | **Día** | **Cliente** | **Cédula de Cliente** | **Estado** | **Fecha** | **Hora de Inicio** | **Duración** | |
| 2 | 1 | La Cigarra S1 | 1 | Miercoles | Erick Rea | 1719455774 | Cancelada | 2022-06-01 | 09:00:00 | 00:30:00 | |
| 3 | 2 | La Cigarra S1 | 1 | Miercoles | Erick Rea | 1719455774 | Completada | 2022-06-01 | 13:00:00 | 00:30:00 | |
| 4 | 3 | La Cigarra S1 | 1 | Miercoles | Erick Rea | 1719455774 | Pendiente | 2022-06-01 | 16:30:00 | 01:00:00 | |
| 5 | 4 | La Cigarra S1 | 1 | Martes | Erick Rea | 1719455774 | Cancelada | 2022-05-31 | 12:30:00 | 00:30:00 | |
| 6 | | | | | | | | | | | |

**Figura 71**

*Gráfico de barras con el número total de reservas según su estado*




### *3.5.5. Sprint Review*

Se han completado los requerimientos estimados durante este ciclo. Se tienen las siguientes observaciones

- Crear una interfaz que siga los principios de usabilidad para las vistas que el usuario cliente observa al momento de realizar reservas y visualizar su historial.

### *3.5.6. Sprint Retrospective*

Este es el último evento del *sprint* donde de la mano de un profesional se da el visto bueno al aplicativo.

**Tabla 13**

*Observaciones del experto en el tema*

| Nombre del Experto | Correo Electrónico | Observaciones |
|---|---|---|
| Damián Nicolalde | danicolalde@puce.edu.ec | Sin observaciones. |

## 3.6. Sprint V

### *3.6.1. Sprint Planning*

El objetivo principal de este *sprint* será el tener una versión funcional que pueda ser usada en una página web mediante una conexión de internet. Además, se mejorará el estado del aplicativo para que funcione de manera óptima según los requerimientos.

### *3.6.2. Sprint Backlog*

Durante este ciclo se desarrollarán los siguientes requerimientos definidos en el *product backlog.*

**Tabla 14**

*Backlog del Sprint V*

| Id. | Requerimiento | Responsable |
|---|---|---|
| **01** | Interfaz Responsive | Erick Rea |
| **03** | Documentación | Erick Rea |
| **16** | Multiusuario | Erick Rea |
| **17** | Plataforma de alojamiento | Erick Rea |
| **18** | Escalabilidad | Erick Rea |
| **19** | Concurrencia | Erick Rea |
| **20** | Seguridad | Erick Rea |

### *3.6.3. Diseño*

Al igual que el *sprint* IV, el diseño de la base de datos se ha terminado en función a los requerimientos.

### *3.6.4. Prototipo*

**3.6.4.1. Interfaz Responsive.** La interfaz responsive ha sido probada a lo largo de los *sprints*. Tanto en la vista de administrador como en la vista de cliente se ha agregado este apartado para que pueda visualizarse de acuerdo con el tamaño de la pantalla de quien ingrese al sistema.

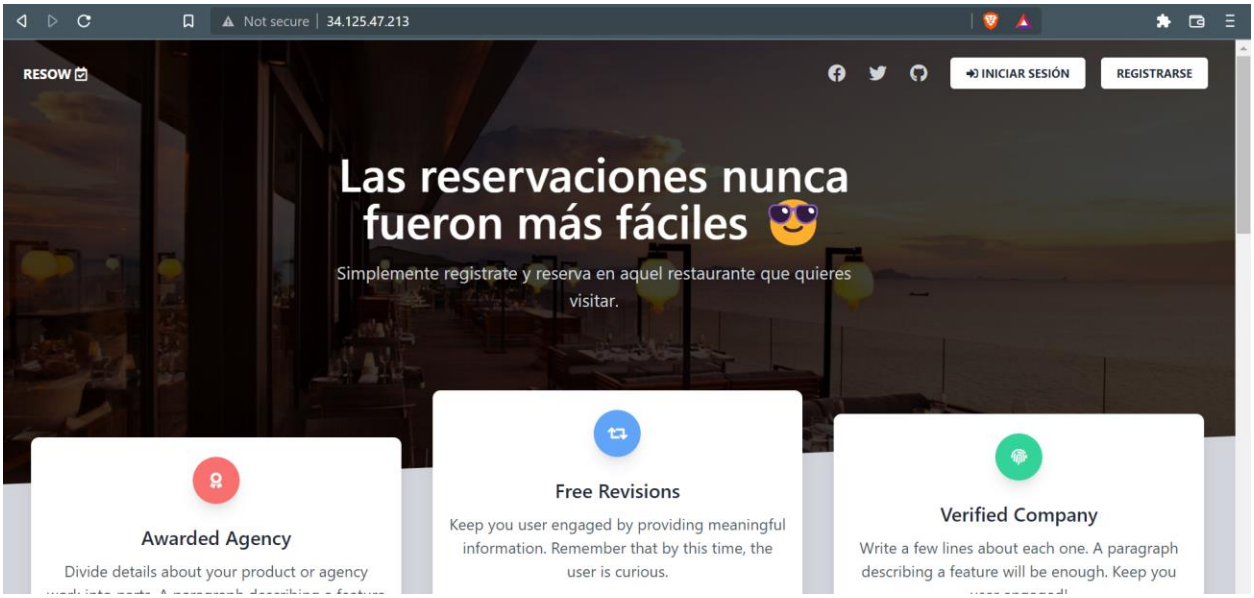
**3.6.4.2. Documentación.** La documentación del sistema se encuentra dentro del código de sus clases, vistas y modelos.

**3.6.4.3. Multiusuario.** Dado que el aplicativo usa Laravel Jetstream para la gestión de sesiones de usuarios, el sistema puede ser usado por múltiples usuarios que estén registrados en el mismo.

**3.6.4.4. Plataforma de alojamiento.** El aplicativo web fue desplegado en una máquina virtual configurada en “Google Cloud Platform”. Se puede acceder al aplicativo mediante la IP “34.125.47.213”.

**Figura 72**

*Página principal mostrada al ingresar al aplicativo mediante su dirección IP*




**3.6.4.5. Escalabilidad.** El aplicativo cuenta con las características necesarias para que pueda expandirse en sus funcionalidades. Por ejemplo, se puede adaptar para que permita reservar en múltiples restaurantes y sus sucursales.

**3.6.4.6. Concurrencia.** El aplicativo cuenta con los controles para que la base de datos pueda ser modificada por varias peticiones al mismo tiempo sin que se vea afectada la integridad de los datos.

**3.6.4.7. Seguridad.** El desarrollo de la aplicación web cuenta con las siguientes características para ser seguro:

- La autenticación de usuarios añadida en un *sprint* previo es un apartado obligatorio para asegurar la integridad del sistema. Se cuenta con un *token* por usuario activo, permitiendo que solo se permita realizar cambios si el usuario ha sido identificado.

- Los usuarios cuentan con roles que delimitan las acciones que pueden realizar dentro del aplicativo.
- El código del aplicativo para realizar transacciones a la base de datos está controlado para que no exista inyección de SQL malicioso.

### *3.6.5. Sprint Review*

Este ha sido el *sprint* final del desarrollo, se han cumplido todos los requerimientos según lo planeado mediante la metodología SCRUM.

### *3.6.6. Sprint Retrospective*

Este es el último evento del *sprint* donde de la mano de un profesional se da el visto bueno al aplicativo.

**Tabla 15**

*Observaciones del experto en el tema*

| **Nombre del Experto** | **Profesión** | **Observaciones** |
|---|---|---|
| Damián Nicolalde | danicolalde@puce.edu.ec | Sin observaciones. |

# Conclusiones y Recomendaciones

## 4.1. Conclusiones

Al finalizar el desarrollo de este aplicativo web enfocado en realizar reservas para restaurantes, se ha llegado a las siguientes conclusiones basadas en los objetivos específicos de este trabajo de titulación.

1. En conclusión, antes de empezar a desarrollar un sistema, es necesario analizar los requerimientos y los diferentes sistemas propietarios o libres que también buscan solventar una necesidad específica; en este caso la gestión de reservas en restaurantes. Este proyecto fue realizado apoyado del estándar IEEE 830, estándar enfocado en la especificación de requisitos de software. Además, se ha analizado el ámbito que sería trabajado para poder conocer que beneficios traería la creación de un aplicativo como el decidido a desarrollar.
2. Para empezar a desarrollar, también ha sido necesario tener un modelo funcional de una base de datos. Las bases de datos relacionales son tradicionales en el ámbito de creación de software. Por esta razón el proyecto se apoya en estas para poder almacenar la información que será recopilada y gestionada por el aplicativo.
3. El uso de una metodología de desarrollo de software ha sido obligatorio para poder construir una aplicación web en un rango de fechas definido. La metodología Scrum fue la escogida para construir el proyecto debido a su flexibilidad. En cada ciclo de desarrollo se obtuvo un prototipo que era incrementado en las iteraciones posteriores hasta poder cumplir con las metas y finalizar el proyecto.

## 4.2. Recomendaciones

1. Se recomienda usar metodologías ágiles de desarrollo de software cuando se necesita que el producto a realizar deba tener un avance constante en periodos de tiempo que hayan sido definidos para el proyecto.
2. Se recomienda que, al momento de desarrollar cualquier tipo de software, se tomen en consideración los principios de usabilidad y de la interacción humano computador, con el fin de que el producto final sea amigable con usuarios expertos o novatos.
3. Se recomienda investigar el ámbito sector en el que un aplicativo resolverá una problemática. Tener esta información permitiría que el equipo encargado de desarrollar un sistema tenga una idea clara del qué y por qué se construirá.

## Referencias Bibliográficas


*About Resy*. (s. f.). Resy. Recuperado el 28 de abril de 2022, de https://resy.com/about

Agut, R. M. (2001). Especificación de Requisitos Software según el estándar de IEEE 830. *Universidad Jaume I. Departamento de Informática. Paper*.

Alexandrov, A., & Lariviere, M. A. (2012). Are reservations recommended? *Manufacturing & Service Operations Management*, *14*(2), 218-230.

Alnasur, S. (2019, 14 marzo). *OpenTable vs. Resy: Reservation Systems Compared*. Eat App. https://restaurant.eatapp.co/blog/opentable-vs-resy

Arias, M. A. (2013). *Introducción a PHP*. IT Campus Academy.

Armenta, B., Rodríguez, I., & Medina, L. (2018). Aplicación del modelo de prototipos: Caso de estudio Software RedbotGamesShop Application of the prototype model: Case study RedbotGamesShop Software. *Revista de Simulación*, *2*(5), 8-13.

Borgen, P. H. (2020, 24 mayo). *Learn Alpine JS in this free interactive tutorial*. freeCodeCamp.Org. https://www.freecodecamp.org/news/learn-alpine-js-in-this-free-tutorial-course/#:%7E:text=Alpine.,as%20opposed%20to%20procedural%20code).

Canós, J. H., Letelier, P., & Penadés, M. C. (2003). Metodologías ágiles en el desarrollo de software. *Universidad Politécnica de Valencia, Valencia*, 1-8.

Chávez, J. D. (2020). Cliente PSQL de PostgreSql.

Cheyre, C., & Acquisti, A. (2018). *Online Intermediaries, Prices, and Survival: A Study of OpenTable and New York City Restaurants*.

Clements, P. C. (1996, March). A survey of architecture description languages. In *Proceedings of the 8th international workshop on software specification and design* (pp. 16-25). IEEE.

Deacon, J. (2009). Model-view-controller (MVC) architecture. http://www.jdl.co.uk/briefings/MVC.pdf.

Deemer, P., Benefield, G., Larman, C., & Vodde, B. (2009). Información básica de SCRUM. *California: Scrum Training Institute*.

Diéguez, D. S. (2015). *Patrón Modelo-Vista-Presentador (MVP)*. MoleQla: revista de Ciencias de la Universidad Pablo de Olavide, (20), 7.

Francia, J. (2017, 25 septiembre). *¿Qué es Scrum?* Scrum.Org. https://www.scrum.org/resources/blog/que-es-scrum

Galo Fariño, R. (2011). Modelo Espiral de un proyecto de desarrollo de software. *http://www.ojovisual.net/galofarino/modeloespiral.pdf*.

Gamboa, J. Z. (2018). Evolución de las Metodologías y Modelos utilizados en el Desarrollo de Software. *INNOVA Research Journal*, *3*(10), 20-33.

Gómez, O. T., López, P. P. R., & Bacalla, J. S. (2010). Criterios de selección de metodologías de desarrollo de software. *Industrial data*, *13*(2), 70-74.

González, J. (2020, 3 septiembre). *En Quito, aforo de restaurantes y centros comerciales aumenta en septiembre*. El Universo. https://www.eluniverso.com/noticias/2020/09/03/nota/7964851/quito-aforo-restaurantes-centros-comerciales-aumenta-septiembre/

Gu, M. X., & Tang, K. (2010, julio). Comparative analysis of WebForms MVC and MVP architecture. In *2010 The 2nd Conference on Environmental Science and Information Application Technology* (Vol. 2, pp. 391-394). IEEE.

Guillén, X. V., & Moldes, L. N. (2019). Arquitectura de aplicaciones web.

Johnson, R. E. (1997). Frameworks=(components+ patterns). *Communications of the ACM*, *40*(10), 39-42.

Joskowicz, J. (2008). Reglas y prácticas en eXtreme Programming. *Universidad de Vigo*, *22*.

Klimek, B., & Skublewska-Paszkowska, M. (2021). Comparison of the performance of relational databases PostgreSQL and MySQL for desktop application. *Journal of Computer Sciences Institute*, *18*, 61-66.

Klimm, M. C. (2021). Design Systems for Micro Frontends-An Investigation into the Development of Framework-Agnostic Design Systems using Svelte and Tailwind CSS.

Krar, S. F. (2009). Tecnología de las máquinas herramientas. México: AlfaOmega.

Kumar, G., & Bhatia, P. K. (2014, February). Comparative analysis of software engineering models from traditional to modern methodologies. In *2014 Fourth International Conference on Advanced Computing & Communication Technologies* (pp. 189-196). IEEE.

Lea, T. (2020, 17 julio). *What is Livewire?* DevDojo. https://devdojo.com/tnylea/what-is-livewire

Maida, E. G., & Pacienzia, J. (2015). Metodologías de desarrollo de software.

Mathieu, M. J. (2014). Introducción a la programación. Grupo Editorial Patria.

Negrea, S. (2020, 4 febrero). *Research Recap: Flexibility in reservation times can increase restaurant revenue*. Cornell SC Johnson. https://business.cornell.edu/hub/2020/02/04/research-reservation-times-increase-restaurant-revenue/

*OpenTable*. (2022, 14 marzo). Capterra. https://www.capterra.ec/software/17313/opentable-for-restaurants

Ortiz, M. (2012). *Modelo Incremental*. Modelos de Desarrollo de Software. https://isw-udistrital.blogspot.com/2012/09/ingenieria-de-software-i.html

Otwell, T. (s. f.). *Getting Started*. Laravel. Recuperado el 2 de mayo de 2022, de https://laravel.com/docs/9.x

Pater, B. J. (2015). *Modern Web Application Frameworks* [Tesis de maestría no publicada]. Universidad Masaryk.

Pater, J. (2015). Modern Web Application Frameworks. *Annals of Dunarea de Jos University. Fascicle I: Economics and Applied Informatics*, *21*(3), 82-86.3

Paz, J. A. M. (2016). Análisis del proceso de pruebas de calidad de software. *Ingeniería solidaria*, *12*(20), 163-176.

Pérez, J. E. (2019). Introducción a JavaScript.

Primicias. (2022, 18 marzo). *Presidente Lasso anuncia 100% de aforo en todas las actividades*. https://www.primicias.ec/noticias/sociedad/lasso-aforo-actividades-turismo-comerciales-deportivas-ecuador/

Reiley, L., & van Dam, A. (2021, 20 mayo). *What are Americans making for dinner? Reservations.* Washington Post. https://www.washingtonpost.com/business/2021/05/20/restaurants-dining-out-pandemic/

Resy Network. (2021, 15 abril). *Resy OS*. Capterra. https://www.capterra.ec/software/197806/resyos

Ruggieri, R., Savastano, M., Scalingi, A., Bala, D., & D'Ascenzo, F. (2018). The impact of digital platforms on business models: An empirical investigation on innovative start-ups. Management & Marketing, 13(4).

Santillán, L. A. C., Ginestà, M. G., & Mora, Ó. P. (2014). Bases de datos en MySQL. *Universitat oberta de Catalunya*.

Smith, P. (2012). Modelo Cascada [Imagen]. Wikipedia. http://en.wikipedia.org/wiki/File:Waterfall_model_%281%29.svg

Sommerville, I. (2004). *Software Engineering* (7.ª ed.). Addison-Wesley.

Suárez, E. M. (2008). *Qué es una base de datos relacional*. Universidad de murcia, Murcia, España.

Sunardi, A. (2019). MVC architecture: A comparative study between laravel framework and slim framework in freelancer project monitoring system web based. *Procedia Computer Science*, *157*, 134-141.

Velásquez, S. M., Montoya, J. D. V., Adasme, M. E. G., Zapata, E. J. R., Pino, A. A., & Marín, S. L. (2019). Una revisión comparativa de la literatura acerca de metodologías tradicionales y modernas de desarrollo de software. *Revista Cintex*, *24*(2), 13-23.

West, S., Gaiardelli, P., & Rapaccini, M. (2018). Exploring technology-driven service innovation in manufacturing firms through the lens of Service Dominant logic. Ifac-Papersonline, 51(11), 1317-1322.

World Bank Group. (2021, 18 octubre). *Ecuador, el país que venció la pesadilla de la pandemia en 100 días*. World Bank. https://www.bancomundial.org/es/news/feature/2021/10/18/ecuador-the-country-that-vanquished-the-nightmare-pandemic-in-100-days#:%7E:text=El%202020%20fue%20a%C3%B1o%20bisiesto,COVID%2D19%20en%20su%20territorio.

Zaballos, A. G., Inglesias, E., Cave, M., Elbittar, A., Guerrero, R., Mariscal, E., & Webb, W. (2020). El impacto de la infraestructura digital en las consecuencias de la COVID-19 y en la mitigación de efectos futuros. *Banco Iberoamericano de Desarrollo. Colombia, Rep. Téc*

Zofío Jiménez, J. (2013). *Aplicaciones web.* Macmillan Iberia, S.A. https://elibro.puce.elogim.com/en/lc/puce/titulos/43262