UNIVERSIDAD POLITÉCNICA SALESIANA

SEDE CUENCA

CARRERA DE INGENIERÍA ELECTRÓNICA

Trabajo de titulación previo
a la obtención del título de
Ingeniero Electrónico

.

PROYECTO TÉCNICO CON ENFOQUE INVESTIGATIVO:

"DISEÑO DEL PROTOTIPO DE UN SISTEMA PARA LA GESTIÓN DE DECISIONES A DISTANCIA EN UN RESTAURANTE UTILIZANDO UNA PLATAFORMA DIGITAL Y CÓDIGOS QR"

AUTOR:

JAVIER EDUARDO CORDERO MERCHÁN

TUTORA:

ING. ANA CECILIA VILLA PARRA, PHD.

CUENCA - ECUADOR

2021

CESIÓN DE DERECHOS DE AUTOR

Yo, Javier Eduardo Cordero Merchán con documento de identificación N° 0105497218, manifiesto mi voluntad y cedo a la Universidad Politécnica Salesiana la titularidad sobre los derechos patrimoniales en virtud de que soy autor del trabajo de titulación: **"DISEÑO DEL PROTOTIPO DE UN SISTEMA PARA LA GESTIÓN DE DECISIONES A DISTANCIA EN UN RESTAURANTE UTILIZANDO UNA PLATAFORMA DIGITAL Y CÓDIGOS QR"**, mismo que ha sido desarrollado para optar por el título de: *Ingeniero Electrónico*, en la Universidad Politécnica Salesiana, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En aplicación a lo determinado en la Ley de Propiedad Intelectual, en mi condición de autor me reservo los derechos morales de la obra antes citada. En concordancia, suscribo este documento en el momento que hago entrega del trabajo final en formato digital a la Biblioteca de la Universidad Politécnica Salesiana.

Cuenca, diciembre de 2021.

Javier Eduardo Cordero Merchán

C.I. 0105497218

CERTIFICACIÓN

Yo, declaro que bajo mi tutoría fue desarrollado el trabajo de titulación: **"DISEÑO DEL PROTOTIPO DE UN SISTEMA PARA LA GESTIÓN DE DECISIONES A DISTANCIA EN UN RESTAURANTE UTILIZANDO UNA PLATAFORMA DIGITAL Y CÓDIGOS QR"**, realizado por Javier Eduardo Cordero Merchán, obteniendo el *Proyecto Técnico con enfoque investigativo* que cumple con todos los requisitos estipulados por la Universidad Politécnica Salesiana.

Cuenca, diciembre de 2021.

Ing. Ana Cecilia Villa Parra, Ph.D.

C.I. 0103874194

DECLARATORIA DE RESPONSABILIDAD

Yo, Javier Eduardo Cordero Merchán con documento de identificación N° 0105497218, autor del trabajo de titulación: **"DISEÑO DEL PROTOTIPO DE UN SISTEMA PARA LA GESTIÓN DE DECISIONES A DISTANCIA EN UN RESTAURANTE UTILIZANDO UNA PLATAFORMA DIGITAL Y CÓDIGOS QR"** certifico que el total contenido del *Proyecto Técnico con enfoque investigativo*, es de mi exclusiva responsabilidad y autoría.

Cuenca, diciembre de 2021.

Javier Eduardo Cordero Merchán

C.I. 0105497218

AGRADECIMIENTOS

Agradezco primeramente a Dios por brindarme la salud, sabiduría y valentía para poder afrontar muchas batallas en el transcurso de mi vida universitaria y permitirme alcanzar esta meta.

En segundo lugar agradezco a mis padres: Manuel Cordero y María Merchán; como también a mis hermanos: Sandra, Alonso, Pablo y Esteban; ya que han sido un pilar esencial en mi desempeño universitario, por apoyarme y alentarme a no rendirme cuando se tiene un objetivo trazado en la vida.

Quiero agradecer también a los Srs. Wilson Auquilla y Geovanny Tapia, quienes me dieron la oportunidad de laborar en sus unidades de taxis para poder sustentar económicamente mis gastos universitarios. Por otra parte agradezco a mis amigos, con quienes he compartido conocimientos y me han colaborado en este trabajo de titulación.

Por último agradezco a la Ing. Ana Cecilia Villa, quien como directora de tesis me aportó con sus conocimientos, paciencia y tiempo en cada una de las etapas del trabajo de titulación.

Javier Eduardo Cordero Merchán

DEDICATORIAS

Quiero dedicar este proyecto primeramente a Dios, he afrontado varias dificultades y gracias a él he triunfado.

También dedico este trabajo a mis padres, quienes han sido un pilar fundamental y motivo de superación en el transcurso de mi vida universitaria.

Javier Eduardo Cordero Merchán

ÍNDICE GENERAL

AGRADECIMIENTOS ..... **I**

ÍNDICE GENERAL ..... **III**

ÍNDICE DE FIGURAS ..... **V**

ÍNDICE DE TABLAS ..... **VII**

RESUMEN ..... **VIII**

INTRODUCCIÓN ..... **IX**

ANTECEDENTES O PROBLEMA DE ESTUDIO ..... **XI**

JUSTIFICACIÓN (IMPORTANCIA Y ALCANCES) ..... **XIII**

OBJETIVOS ..... **XIV**

OBJETIVO GENERAL XIV

OBJETIVOS ESPECÍFICOS XIV

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA O ESTADO DEL ARTE **1**

1.1 SARS-COV-2 O COVID-19 1

1.2 CÓDIGOS QR 2

1.3 SERVIDOR WEB 3

1.3.1 Servidores Web más utilizados 3

1.4 BASE DE DATOS 4

1.5 SISTEMAS GESTORES DE BASES DE DATOS 4

1.6 PYTHON 5

1.7 LENGUAJES DE PROGRAMACIÓN PARA LA INTERFAZ DE CÓDIGOS QR 5

1.8 LENGUAJES DE PROGRAMACIÓN PARA LA PLATAFORMA DIGITAL 6

CAPÍTULO 2: DISEÑO E IMPLEMENTACIÓN DEL PROTOTIPO **7**

2.1 ARQUITECTURA DEL PROTOTIPO 7

2.2 DESCRIPCIÓN DEL SISTEMA 8

2.3 HARDWARE 8

2.3.1 CELULAR 9

2.3.2 COMPUTADORA 9

2.3.3 ENRUTADOR 9

2.4 SOFTWARE 10

2.4.1 INTERFAZ GRÁFICA PARA GENERAR CÓDIGOS QR 10

2.4.2 PLATAFORMA DIGITAL 14

2.4.3 ESTRUCTURA DEL PROYECTO WEB 16

2.4.4 INTERFAZ PARA ADMINISTRACIÓN 30

2.4.5 SISTEMA GESTOR DE BASE DE DATOS 37

2.5 PRUEBAS 40
2.5.1 PRUEBAS TÉCNICAS 40
2.5.2 PROTOCOLO DE PRUEBAS 41

CAPÍTULO 3: RESULTADOS Y ANÁLISIS DE RESULTADOS **42**

3.1 INTERFAZ PARA GENERAR CODIGOS QR 42
3.2 PLATAFORMA DIGITAL 42
3.3 PRUEBAS A LOS SERVIDORES 47
3.4 ANÁLISIS DE COSTO DEL PROTOTIPO 49

CAPÍTULO 4: CONCLUSIONES Y RECOMENDACIONES **52**

REFERENCIAS BIBLIOGRÁFICAS **54**

ANEXOS **56**

ANEXO 1: MANUAL DE USUARIO 56
ANEXO 2: 61
MANUAL DE ADMINISTRADOR PARA LA PLATAFORMA DIGITAL 61
ANEXO 3: MANUAL DE ADMINISTRADOR PARA LA INTERFAZ DE CODIGOS QR 63

ÍNDICE DE FIGURAS

Figura 1. Código qr de tipo 2d. **2**

Figura 2. Sintaxis de lenguaje de programación python **5**

Figura 3. Arquitectura del prototipo de la plataforma digital y códigos qr **7**

Figura 4. Diagrama de bloques del funcionamiento del sistema **8**

Figura 5. Interfaz de diseño en qtdesigner **10**

Figura 6. Diagrama de flujo para la generación de un código qr **13**

Figura 7. Interfaz gráfica para generar códigos qr **14**

Figura 8. (a) maquetación del sitio web; (b)menú del restaurante aplicado un estilo responsive **15**

Figura 9. Diagrama de flujo del modo de operación de django **16**

Figura 10. Diagrama de flujo de la plataforma digital **20**

Figura 11. Modelo para los tipos de carta del restaurante **21**

Figura 12. Modelo para los platillos del restaurante **22**

Figura 13. Función lista_platos para mostrar el menú del restaurante **22**

Figura 14. Direcciones urls de la aplicación menú **23**

Figura 15. (a) formulario de registro de usuario; (b) formulario de inicio de sesión. **24**

Figura 16. Interfaz de presentación del proceso de añadir productos al carrito **25**

Figura 17. Decorador login_required para permisos de usuario **25**

Figura 18. Procesadores de contexto para la cantidad de productos y precio total **26**

Figura 19. Formulario para datos de factura, tipo de pago y vajilla adicional **27**

Figura 20. Modelo pedido perteneciente a la aplicación pedidos **28**

Figura 21. Modelo listadopedido perteneciente a la aplicación pedidos **28**

Figura 22. Pdf de la factura del cliente. **29**

Figura 23. Registro de modelos en la interfaz de administración de django **30**

Figura 24. Diagrama de flujo de la interfaz de administración de la plataforma digital **32**

Figura 25. Sitio para la administración **33**

(a) **34**

Figura 26. (a) permisos de usuario administrador por grupos. (b) usuarios registrados y permisos otorgados **34**

Figura 27. (a) sección de tipos de carta que oferta el restaurante, ... **36**

(b) sección de platos que ofrece el restaurante con sus características **36**

Figura 28. Interfaz de los pedidos gestionados por los usuarios (clientes) **36**

Figura 29. Detalle de un pedido de usuario **37**

Figura 30. Diagrama entidad relación de la base de datos. **39**

Figura 31. Escaneo de código qr por parte de un usuario¡ERROR! MARCADOR NO DEFINIDO.

Figura 32. Lectura de código qr para acceso a la plataforma digital¡ERROR! MARCADOR NO DEFINIDO.

Figura 33. Navegación en la plataforma digital para gestionar un pedido **43**

Figura 34. Configuración del servicio dns en el router **48**

ÍNDICE DE TABLAS

Tabla 1. Sistemas gestores de bases de datos..... 4

Tabla 2. Recursos para el desarrollo de una interfaz para códigos qr [23]-[24] 5

Tabla 3. Recursos para el desarrollo de una plataforma digital [22]-[29]..... 6

Tabla 4. Características de la computadora usada para el servidor web. 9

Tabla 5. Características del enrutador..... 9

Tabla 7. Funciones para el carrito de compras..... 26

Tabla 8. Propiedades para modelos del sitio de administración 31

RESUMEN

En este escrito se describe el diseño del prototipo de un sistema para realizar pedidos en un restaurante, con el cual los usuarios, haciendo uso de códigos QR, pueden acceder a una plataforma digital y gestionar pedidos de acuerdo con lo propuesto por el administrador del restaurante. Este sistema se propuso como un método para prevenir contagios causados por la enfermedad de tipo viral COVID-19.

El sistema está compuesto por una interfaz para generar códigos QR elaborado con Qtdesigner y PyQt5, que permite vincular el diseño con el lenguaje de programación Python. Además, se elaboró una plataforma digital con Django, como entorno de programación, con tecnologías complementarias CSS3, HTML5 y Bootstrap. La plataforma digital brinda una interfaz de usuario en la cual se pueden ejecutar tareas como: adición, disminución y eliminación de pedidos, selección de diferentes opciones para complementar un pedido y la solicitud de factura. También se dispone de una interfaz para el administrador, con la cual se pueden gestionar los pedidos de los usuarios (clientes), los permisos de usuarios y los reportes de ventas.

El diseño buscó que el sistema sea escalable y que pueda acoplarse a diversos entornos con pequeñas modificaciones de acuerdo con las necesidades o exigencias que demande cada proyecto, por ejemplo: servicios de hotelería y turismo, servicios de entregas, entre otros.

A fin de validar el prototipo se ejecutaron pruebas piloto haciendo uso de un servidor local, simulando diferentes pedidos y generando reportes de la factura, con envío al correo electrónico del usuario. Se establecieron permisos de usuarios administradores para gestionar los pedidos de la plataforma digital y se distribuyeron copias de un ejecutable para generar códigos QR. Una vez que los usuarios interactuaron con el sistema se pidió llenar una encuesta SUS (System Usability Scale) con el propósito de evaluar el nivel de satisfacción, obteniendo resultados satisfactorios. Se verificó por tanto el correcto funcionamiento y se cumplieron los objetivos planteados.

INTRODUCCIÓN

"A finales del año 2019 en el mes de diciembre, en la ciudad de China Wuhan, apareció la enfermedad de infección respiratoria tipo viral *SARS-CoV-2/COVID-19",* que hasta noviembre del 2021 ha causado contagios a nivel mundial con cifras que registran 259,185,236 de personas confirmadas y un total de 5,171,809 personas fallecidas [1], [2]. Las cifras indican lo grave que puede ser la enfermedad y la tasa de mortalidad que conlleva. La carga viral del COVID-19 y sus modos de transmisión impredecibles, que se presenta a través de: estornudos, superficies contaminadas, falta de higiene, aglomeraciones, entre otros factores, han exigido el uso de tecnologías que minimice el contacto en entornos como: centros comerciales, restaurantes y sitios de trabajo.

En este contexto, se planteó el diseño de un prototipo de sistema para la gestión de decisiones en un restaurante utilizando una plataforma digital y códigos QR como una propuesta para lograr el distanciamiento social. El sistema requiere la creación de una interfaz gráfica que permita la generación de códigos QR que brinden información del número de mesa en la que se encuentre el cliente y el enlace a una plataforma digital para la gestión del pedido con opciones como: menú, precios, edición y anulación de órdenes, disponibilidad de pedidos adicionales, solicitud de la factura y el tipo de pago (efectivo o tarjeta de crédito). Para que el área administrativa realice la gestión de pedidos es requerida una plataforma digital con una interfaz para la edición del menú (platos, precios) y la gestión de la información de los pedidos realizados.

En este escrito se describe el proceso de desarrollo del sistema anteriormente mencionado. En Capítulo 1 se encuentra la revisión del estado del arte con los fundamentos de los códigos QR, lenguajes de programación para desarrollo web, plataformas digitales, bases de datos y servidores web.

En el Capítulo 2 se describe la arquitectura del prototipo del sistema propuesto, los criterios considerados para el diseño de la interfaz de códigos QR y las tecnologías usadas para el desarrollo de la plataforma digital. A fin de verificar el correcto funcionamiento del prototipo, se describen las pruebas a realizar para validar el funcionamiento del sistema y el protocolo a seguir para llevar a cabo un pedido.

En el Capítulo 3 se detallan los resultados con base a las pruebas ejecutadas en los navegadores web, el desempeño de la aplicación en los dispositivos móviles, la carga de del servidor web y la opinión de los participantes con base a una encuesta de usabilidad.

En el Capítulo 4 se describe las conclusiones y las recomendaciones con base a los objetivos planteados para el desarrollo del prototipo. Se plantean las recomendaciones para trabajos futuros con base a los comentarios de los participantes en las pruebas y al administrador de la plataforma.

ANTECEDENTES O PROBLEMA DE ESTUDIO

La COVID-19 es una enfermedad de infección respiratoria, diferente a otras sepas causadas por coronavirus, que se propaga rápidamente y con brotes que aumentan de manera exponencial. “Las cifras a nivel mundial anuncian mas de 5 millones de fallecidos y 259,185,236 casos confirmados [1], de este último grupo el 5% padece de síndrome de distrés respiratorio agudo (SDRA) y fracaso multiorgánico”. Estos casos exigen que el paciente sea ingresado a “Unidad de Cuidados Intensivos (UCI) para una posterior intubación orotraqueal y ventilación mecánica invasiva (VMI)” [3].

Ante esta emergencia, la humanidad tuvo que tomar medidas necesarias de higiene y protección. Para esto fue necesario adoptar conductas como lavarse correctamente y frecuentemente las manos, usar mascarillas que cumplan con las normas de protección, mantener un distanciamiento de al menos dos metros con personas que no conforman el entorno familiar, evitar aglomeraciones, mantener una correcta limpieza y desinfección de lugares que se tocan con frecuencia y aislamiento en el caso de estar contagiado/a [4].

En enero del 2021 empezó la vacunación y hasta la fecha se tiene la aprobación de más de 10 vacunas desarrolladas en diferentes laboratorios [5]. Sin embargo, a nivel mundial existen 501,000 secuencias o variantes genómicas que pueden causar ineficacia de las vacunas y reinfecciones [6].

Con base a esta emergencia sanitaria, las tecnologías de la información y las comunicaciones (TIC), ha aportado significativamente a los procesos de automatización y gestión de actividades en diferentes campos, enfocándose en las tecnologías activadas sin contacto directo. Ante el aumento de contagios, se están empleando en áreas como la salud (telemedicina), en el ámbito comercial e industrial tecnologías de soporte para evitar el contacto directo como son: comandos de voz, reconocimiento facial, sensores de gestos, el uso de teléfonos móviles con apps.

Los comandos de voz en la actualidad se emplean en gran medida, como por ejemplo los servicios de Siri (Apple), Bixby (Samsung) o Cortana (Microsoft). La utilidad de estas herramientas no tiene limitaciones debido a que se pueden ejecutar mediante móviles, tablets y ordenadores. El empleo de esta tecnología es notorio en el entorno

doméstico, espacios públicos, servicios financieros entre otros [7], sin embargo, pueden presentar limitaciones en ambientes muy ruidosos.

El uso de sensores de gestos es útil para controles on-off y evitar la manipulación de pantallas o botones. Su utilidad es muy extendida para el mando de luminaria, control para el acceso en las puertas, regulación de temperatura, control de ascensores, etc. Sin embargo, éstas podrían presentar latencias altas en el reconocimiento de los movimientos y un tiempo de adaptación considerable por parte de las personas [7].

Con gran demanda se registra actualmente el uso de apps y móviles, gracias a lo cual ha sido posible automatizar gestiones como: pagos en línea, identificación de personas, transferencias bancarias, atenciones médicas personalizadas, etc. El empleo de las tecnologías mencionadas ha permitido agilitar procesos e incentivar al marketing digital para instituciones educativas, sector empresarial y áreas de la salud [8]. Sin embargo, en nuestro medio, son pocos los sistemas basados en apps que estén orientados a la gestión de información en ambientes públicos que particularmente exigen interacción directa y presencial entre un cliente y un proveedor. Considerando la recomendación de evitar la manipulación de superficies o botones, es necesario aprovechar la masificación del uso de apps para ofrecer sistemas que permitan gestionar información sin contacto directo entre personas o superficies. Acciones como manipular documentos, dar a conocer decisiones, enviar órdenes, etc., se desarrollan en ambientes en donde es necesario el contacto táctil con superficies o contacto directo con interlocutores. En las condiciones actuales, esta interacción representa un riesgo para la salud y el incorporar tecnología podría ayudar a automatizar procesos y garantizar las condiciones de distanciamiento recomendadas. Es claro entonces que, debido al gran número de personas que tienen acceso a un teléfono inteligente y al incremento de las plataformas para el diseño de apps es posible ampliar los campos de acción en las cuales esta tecnología pueda ayudar a realizar actividades con el menor contacto posible y servir de soporte a las medidas de distanciamiento social.

JUSTIFICACIÓN (IMPORTANCIA Y ALCANCES)

Debido a las necesidades de distanciamiento social y la necesidad de evitar contacto con superficies por el COVID-19 para reducir los contagios, es necesario la implementación de tecnologías que pueda ayudar a realizar actividades con el menor contacto posible, como es el caso de los restaurantes que exigen un estricto control. En locales de este tipo, los clientes están obligados a entablar una conversación para realizar pedidos o manipular la carta del menú. Estas pueden ser acciones que exponen a un posible contagio de afección respiratoria entre las personas que intervienen (mesero-cliente-cajero).

Tecnologías como plataformas digitales, son de gran ayuda para el comercio; ya que se pueden gestionar órdenes mediante un dispositivo móvil inteligente como celulares y tablets. Por otro lado, en los últimos años, el uso de códigos QR y de la cámara de un dispositivo móvil se ha masificado con el objetivo de permitir el acceso a información y enlaces a páginas web, películas, revistas, aplicaciones, productos, etc.

Con el objetivo de aportar con recursos para afrontar la pandemia que se derivo por la COVID-19, se propuso el desarrollo de un sistema que emplee esta tecnología para que pueda ser utilizada en espacios en los cuales es ampliamente requerido el distanciamiento social. La primera versión del sistema se ha enfocado en la gestión de pedidos en un restaurante, sin embargo, el sistema puede ser aplicado con pequeñas modificaciones para otras aplicaciones en hoteles, aeropuertos y universidades.

OBJETIVOS

OBJETIVO GENERAL

Diseñar el prototipo de un sistema para la gestión de decisiones a distancia en un restaurante utilizando una plataforma digital y códigos QR.

OBJETIVOS ESPECÍFICOS

1. Realizar una revisión sobre el uso de plataformas digitales, generación de códigos QR y servidores web.
2. Diseñar una interfaz para la generación de códigos QR que permita la lectura de información contenida en una plataforma digital mediante un dispositivo móvil.
3. Diseñar una plataforma digital para visualizar la carta de un restaurante y gestionar pedidos mediante la interacción del cliente con un dispositivo móvil.
4. Diseñar una interfaz para el proveedor que permita la gestión de los pedidos con la base de datos.
5. Realizar pruebas para evaluar el funcionamiento del prototipo.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA O ESTADO DEL ARTE

1.1 **SARS-COV-2 O COVID-19**

"A finales del año 2019 en la ciudad de Wuhan se reportaron los primeros casos de *SARS-CoV-2/COVID-19"* o más conocido como coronavirus. En enero del 2020, el gobierno de China conjunto al personal médico identificaron al nuevo agente causante de la enfermedad, un virus de la familia *Corovinaviridae, denominado SARS-CoV-2,* o Síndrome Respiratorio Agudo Grave" [2], [9]. Este virus empezó a propagarse de manera rápida alrededor del mundo y la "Organización Mundial de la Salud (OMS) el 11 de marzo del 2020, declaró una Pandemia Mundial" [9]. Esto obligó a la mayoría de países del mundo a realizar cuarentena obligatoria y tomar medidas de bioseguridad para poder controlar la propagación del virus [9].

Oficialmente "la OMS, ha definido al SARS-COV-2/Covid-19 como una enfermedad infecciosa causada por un coronavirus recientemente descubierto, que afecta de distintas maneras a la funcionalidad de los órganos internos de las persona" [10]. "Este patógeno, perteneciente a la familia de los beta-coronavirus", posee una alta capacidad zoonótica que la hace altamente transmisible [10]. Actualmente, las investigaciones relacionan a este virus con los coronavirus encontrados en los murciélagos [11]. Los coronavirus que producen afecciones al ser humano pueden causar sintomatología común de un resfriado estacional o complicaciones mayores como el Síndrome Respiratorio Grave Agudo (SARS) [12].

Con la finalidad de reducir la propagación del virus COVID-19 y salvaguardar la vida de las personas, los centros de salud en conjunto con la OMS, recomiendan acatarse a medidas de seguridad como son [4]:

- El lavado de manos constante con jabón si la persona está expuesta a lugares de mucha contaminación por un tiempo mínimo de 20 segundos.
- Utilizar alcohol o gel antibacterial con un concentrado mínimo de 60%.
- Evitar llevarse las manos al rostro e ingerir alimentos.

- Cuidar la higiene respiratoria con el uso correcto de mascarillas que cubran la nariz y boca completamente, las mismas que deberían ser remplazadas en un periodo de 3 a 7 días como máximo dependiendo de la frecuencia de uso.
- Usar el ángulo interno del codo para estornudar o toser en caso de no poseer mascarilla.
- Mantener la distancia mayor a dos metros cuando se encuentre en lugares públicos, debido a que hay personas que no presentan síntomas pero pueden ser un foco de contagio.
- Limpiar y desinfectar las superficies y objetos con alto índice de manipulación como: manijas de puertas, barandas de seguridad, pulsantes de ascensores, interruptores de luz, mesas, lavamanos, inodoros, cartas de restaurantes, etc
- Evitar aglomeraciones y limitar frecuentar lugares cerrados como gimnasios, cines, bares, discotecas, mercados, supermercados entre otros.
- Respetar el aforo permitido y contar con una ventilación adecuada.

La tecnología está aportando en gran medida para cumplir con estas recomendaciones y es importante contar con la disciplina y recursos para seguir afrontando la situación de pandemia.

1.2 CÓDIGOS QR

Con la creación de dispositivos móviles inteligentes y la migración al internet, el código QR o Código Abierto (ver Figura 1), se emplea cada vez más para facilitar el acceso a información [13].

Figura 1. Código QR de tipo 2D.

Fuente: [14]

El código QR “es un código de barras de matriz bidimensional”, que tiene capacidad de almacenar información, corrección de errores y representación de hasta 7089 caracteres que pueden ser interpretados por cualquier dispositivo electrónico que

disponga de una cámara y cuente con un software para la decodificación [13], [14]. Un Código QR puede ser identificado fácilmente, ya que tiene forma cuadrada y como estándar disponer de cuadrados oscuros y claros en tres esquinas del símbolo [14].

Los Códigos QR, gracias a su diseño que radica en la transmisión de información, tanto en dirección vertical como horizontal, se pueden leer rápidamente (hasta 100 dígitos numéricos en 30 ms) lo que facilita la lectura independientemente de la posición en la que se presente el código al escáner. Puede ser capaz de contener miles de datos, como caracteres numéricos, alfabéticos, símbolos kanji, hiragana, kana, códigos de control y códigos binarios [15].

Los Códigos QR son utilizados en la industria para gestionar y controlar inventarios, generar publicidad, dar información detallada de algún objeto o lugar, etc [16].

1.3 SERVIDOR WEB

Un "servidor web es un software diseñado para procesar datos de una página web", de manera dinámica con todo su contenido (textos, banners, widgets, imágenes, etc.), realizando comunicaciones de entrada y salida, generando una respuesta en cualquier tipo de lenguaje, generalmente HTTP [17]. Se pueden utilizar múltiples tecnologías para aumentar la potencia del servidor y entregar páginas HTML, las misma que incluyen scripts CGI, seguridad SSL y ASP [17].

Cuando se navega por la web, "automáticamente se accede a cientos de Servidores Web, los cuales almacenan información en forma de páginas Web", dispuestas para su rápida entrega [17].

1.3.1 Servidores Web más utilizados

Según la compañía inglesa Netcraft los servidores web más utilizados son:

- **Apache:** Es multiplataforma y su estructura permite el uso de diversos lenguajes (PHP, Python y Perl), incorporando características como la compresión de datos, utilización de URL´s confiables y conexiones seguras [18].
- **Microsoft IIS:** "Cuenta con un gran número de módulos, pero también con gran hándicap de funcionar exclusivamente en Windows" [18].

- **Google Web Server:** Más conocido por sus siglas como GWS, es la versión más adaptada de Apache. “Este servidor web emplea sus servicios como Blogger o App Engine” [18].
- **Nginx:** Este servidor web es multiplataforma y “es utilizado por algunos sitios comunes como Wordpress, Hulu entre otros” [18].

1.4 BASE DE DATOS

Las bases de datos son un conjunto de datos o “información organizada por campos, registros y archivos”. Se dividen en dos grupos:

a) *Bases de datos relacionales*, que almacenan información en una tabla semejante a una hoja de cálculo.

b) *Bases de datos no relacionale*s que son usadas para procesar grandes cantidades de información en forma de estructuras horizontales.

1.5 SISTEMAS GESTORES DE BASES DE DATOS

Los “Sistemas Gestores de Bases de Datos (SGBD)” facilitan la creación y administración de un conjunto de datos que están relacionados entre sí, además de que proporciona seguridad y almacenan a las bases de datos [19]. Los SGBD pueden ser clasificados de acuerdo al modo en el que administran los datos en: relacionales (SQL) y no relacionales (NoSQL) [20].

En la Tabla 1 se muestran ejemplos de cada tipo de los SGBD.

Tabla 1. Clasificación de los Sistemas gestores de bases de datos

SGBD SQL	**SGBD NoSQL**
MySQL	Mongodb
MariaDB	Redis
SQLite	Cassandra
PostgreSQL	Apache Hbase
Microsoft SQL Server	Amazon DynamoDB
ORACLE	ORACLE NoSQL

Los SGBD SQL permiten administrar una base de datos relacionales. “Se basan en las relaciones o vínculos que pueden existir entre los datos”, almacenados en forma de tablas. MySQL es un sistema que permite administrar bases de datos de tipo relacional. El motor de base de datos MySQL es de código abierto y multiplataforma codificado en el lenguaje

de programación C y C++. Emplea el lenguaje de consultas estructurado (SQL), lo cual facilita establecer relaciones uno a uno, uno a muchos y muchos a muchos [21].

Los SGBD NoSQL o no relacionales tratan con bases de datos que no tienen estructuras fijas como las tablas que están compuestas de columnas y filas [20].

1.6 PYTHON

"Python es un lenguaje de alto nivel" para el ámbito de la programación con una sintaxis clara y sencilla, flexible, dinámica y con facilidad de aprendizaje. Python puede ser instalado y ejecutado en cualquier maquina indistintamente del sistema operativo y se utiliza en muchas áreas de ingeniería porque cuenta con librerías que permiten construir cualquier tipo de proyecto [18]. En la Figura 2 se muestra un ejemplo de la sintaxis de Python.

```
def dibujar_muneco(opcion):
    if opcion == 1:
        C.create_line(580, 150, 580, 320, width=4, fill="blue")
        C.create_oval(510, 150, 560, 200, width=2, fill='PeachPuff')
```

Figura 2. Sintaxis de lenguaje de programación Python

Fuente: [18]

1.7 LENGUAJES DE PROGRAMACIÓN PARA LA INTERFAZ DE CÓDIGOS QR

En la Tabla 2 se describen las herramientas requeridas para el desarrollo de una interfaz para generar códigos QR.

Tabla 2. Recursos para el desarrollo de una interfaz para códigos QR [23], [24]

Recurso	Características
Qt Designer	• Permite el diseño de interfaces gráficas GUI visibles para el usuario final • Permite añadir widgets para el control de acuerdo con la lógica de programación requerida.
PyQt5	• Módulo que facilita el desarrollo de aplicaciones con interfaces gráficas, para Python. • Opera con Qt Designer que permite el diseño de forma gráfica de los componentes de la interfaz a diseñarse.
Visual Studio Code	• Editor de código fuente multiplataforma gratuito compatible con Python, Java, C++, JavaScript • Permite la instalación de extensiones que ayudan al programador a completar o sugerir código

1.8 LENGUAJES DE PROGRAMACIÓN PARA LA PLATAFORMA DIGITAL

En la Tabla 3 se describen las herramientas requeridas para el desarrollo de una plataforma digital haciendo uso de tecnologías a nivel de Frontend y Backend.

Tabla 3. Recursos para el desarrollo de una plataforma digital [22]-[29].

Recurso	Características
DJANGO	• Entorno de trabajo del lado del servidor "Backend" escrito en Python. • Facilita la elaboración de aplicaciones y sitios web. • "Consigna no te repitas ((DRY, del inglés *Don't Repeat Yourself*))" • Basado en la arquitectura MVT que separa la lógica de programación por módulos (modelo, vista y plantilla). • Modelo: gestión y relaciones con la base de datos • Vista: lógica para acceder a la base de datos y mostrar la plantilla HTML • Plantilla: presentación del contenido
HTML5	• Lenguaje de marcado de hipertexto • Permite estructurar el contenido web de una página, aplicación o blog. • Utiliza etiquetas para organizar el contenido. • Permite el desarrollo multiplataforma para dispositivos móviles y PC indistintamente del sistema operativo permitiendo diseño *responsive.*
CSS3	• Hoja de estilos en cascada (coding style sheets). • Permite personalizar las etiquetas del lenguaje HTML con la finalidad de dar una mejor apariencia y visualización de cada elemento.
Bootstrap 4.6	• Entorno de trabajo que usa JavaScript y CSS para la maquetación web • Facilita la creación de sitios web *responsive.* • Estructurado por clases predefinidas que se incluye en el lenguaje HTML del sitio o aplicación web. • Facilita el desarrollo web porque utiliza menús para navegación, botones desplegables, efectos de imágenes, etc.

CAPÍTULO 2: DISEÑO E IMPLEMENTACIÓN DEL PROTOTIPO

2.1 ARQUITECTURA DEL PROTOTIPO

La arquitectura del prototipo de la plataforma digital y códigos QR del sistema se indica en la Figura 3.

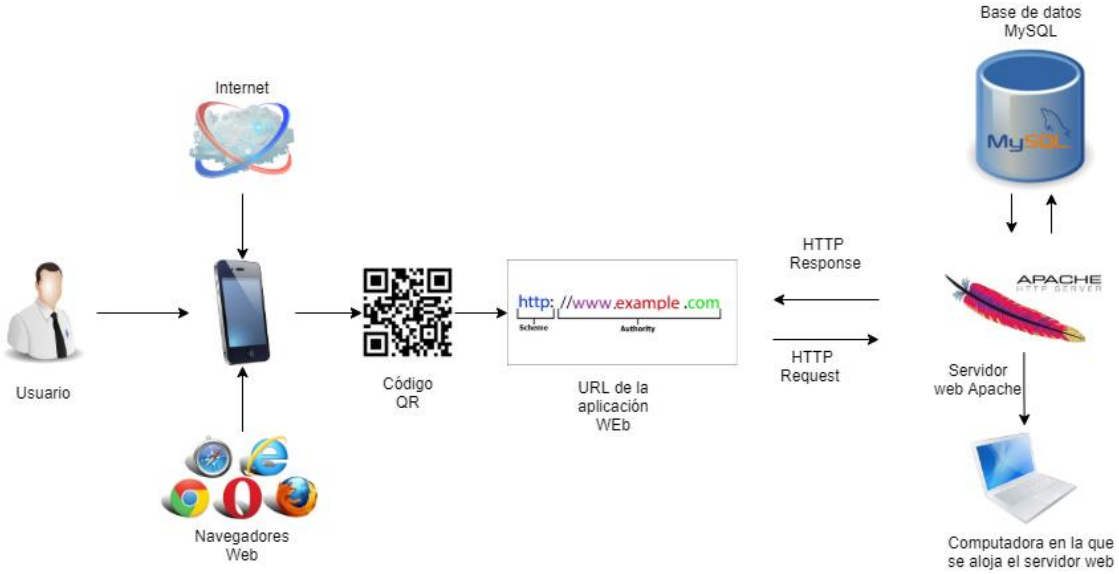


Figura 3. Arquitectura del prototipo de la plataforma digital y códigos QR.

Fuente: Autor

En primer lugar, el usuario debe escanear con su celular un código QR para navegar en la plataforma digital, en este caso, la plataforma de un restaurante.

Es necesario tener conexión a internet para realizar pedidos en el local usando un navegador web del celular.

La plataforma digital en internet emplea el servidor web de producción Apache, que esta vinculada con la base de datos creada en MySQL.

El usuario al navegar en la plataforma digital realiza peticiones HTTP Request al servidor, y este a su vez entrega la información solicitada mediante una respuesta HTTP Response.

2.2 DESCRIPCIÓN DEL SISTEMA

En la Figura 4 se muestra el diagrama de flujo en donde se exponen las diferentes etapas y procesos del sistema.

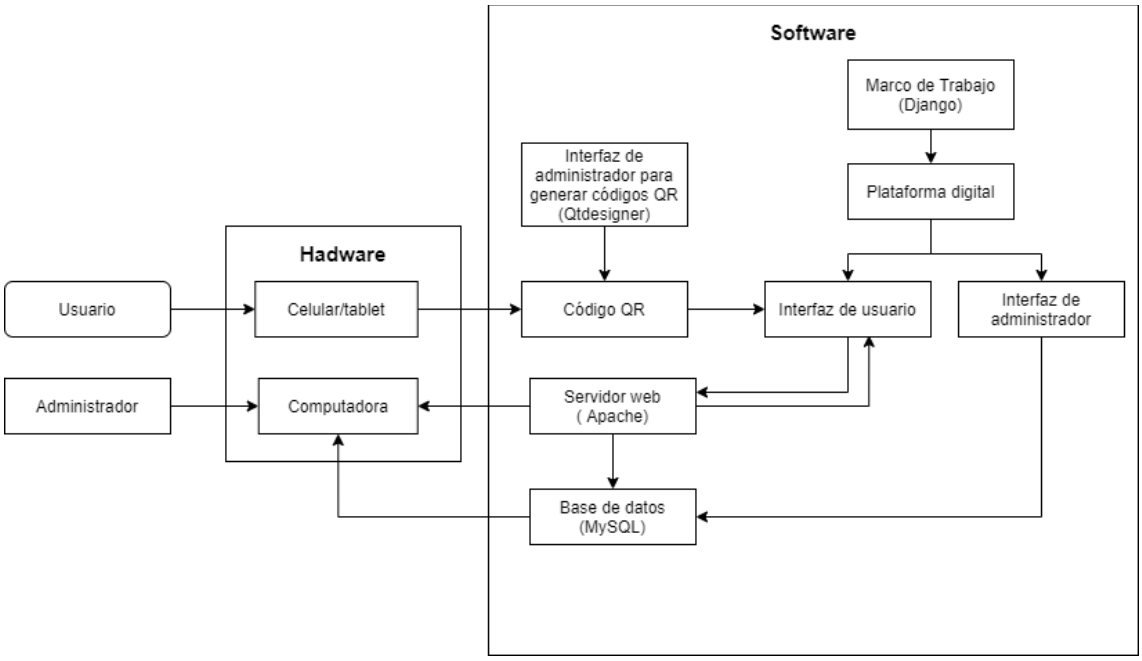


Figura 4. Diagrama de bloques del funcionamiento del sistema.

Fuente: Autor

Para la implementación y ejecución del sistema se emplea hardware y software, a los cuales tiene acceso el usuario y el administrador.

Como software se cuenta con lo siguiente: la interfaz para generar códigos QR, una plataforma digital integrada por la interfaz de usuario y administrador, el servidor web para las peticiones que se realicen, que a su vez hace consultas a la base de datos.

2.3 HARDWARE

El hardware del sistema incluye un celular inteligente o tablet del usuario para escanear el código QR, en el cual se encuentra el enlace a la interfaz de usuario, perteneciente a la plataforma digital del restaurante. Además, se cuenta con un computador para el usuario-administrador en el que se ejecuta el servidor web Apache y un router para disponer de conexión a internet.

En todos estos equipos es necesario configurar y establecer los requerimientos que se detallan a continuación:

2.3.1 CELULAR

El celular o tablet debe tener disponibilidad para usar la cámara y poder escanear el código QR, además de contar con una aplicación que permita realizar la lectura del código y conexión a internet (Red local del restaurante). Se puede trabajar con cualquier sistema operativo.

2.3.2 COMPUTADOR

El computador en el que se instala el servidor web debe contar con las características que se muestran en la Tabla 4**.**

Tabla 4. Características de la computadora usada para el servidor web.

Modelo	HP ENVY 15x360 PC
Memoria Ram	8GB
Disco duro	1 TB
Sistema operativo	Windows 10 de 64 bits
Procesador	Intel(R) Core(TM) i7-4510U CPU @2.00GHz 2.60 GHz

2.3.3 ENRUTADOR

El dispositivo enrutador para la disponibilidad de internet es una ONT (Terminal de Red Óptica) con las características que se muestra en la Tabla 5.

El ISP (Proveedor de Servicios de Internet) contratado para este proyecto es ETAPA, con 20Mbps de subida y de bajada, con una compartición 2:1.

Tabla 5. Características del enrutador

Marca	Huawei
Modelo	EchoLife EG8145V5
Fuente de alimentación de entrada	100 V a 240 V CA, 50 Hz/60 Hz
Fuente de alimentación del sistema	11 V a 14 V CC, 2 A
Puertos de usuario	1 POTS + 4GE + Wi-Fi + USB
Puertos de red	GPON
Indicadores	POWER, PON, LOS, LAN1, LAN2, LAN3, LAN4, TEL, USB, WLAN y WPS

2.4 SOFTWARE

El software consta de un ejecutable de Python que facilita la creación de códigos QR, que se desarrolló con Qtdesigner y Python. Se dispone además de una plataforma digital usando el framework Django que usa Python como lenguaje intérprete.

Como programa editor de código se emplea Visual Studio Code, disponible para el sistema operativo Windows 10. Este editor de código tiene grandes ventajas ya que permite integrar varios lenguajes de programación en el desarrollo web y librerías de acuerdo al lenguaje que se esté utilizando, además de integrar una consola para la edición y depuración de código.

2.4.1 INTERFAZ GRÁFICA PARA GENERAR CÓDIGOS QR

La interfaz gráfica para generar códigos QR se elaboró mediante Qtdesigner, que permite realizar el diseño de la interfaz sin involucrarse en el código de programación. Esto facilitó el diseño de la interfaz ya que para la construcción se arrastran a la ventana principal los widgets necesarios como son: etiquetas de texto, botones, barras de texto, ventanas, diálogos etc). En QtDesigner se pueden establecer características de los widgets como: ancho, alto, alineación vertical y horizontal dentro de la ventana principal o QMainWindow además de colores y texto (ver Figura 5).

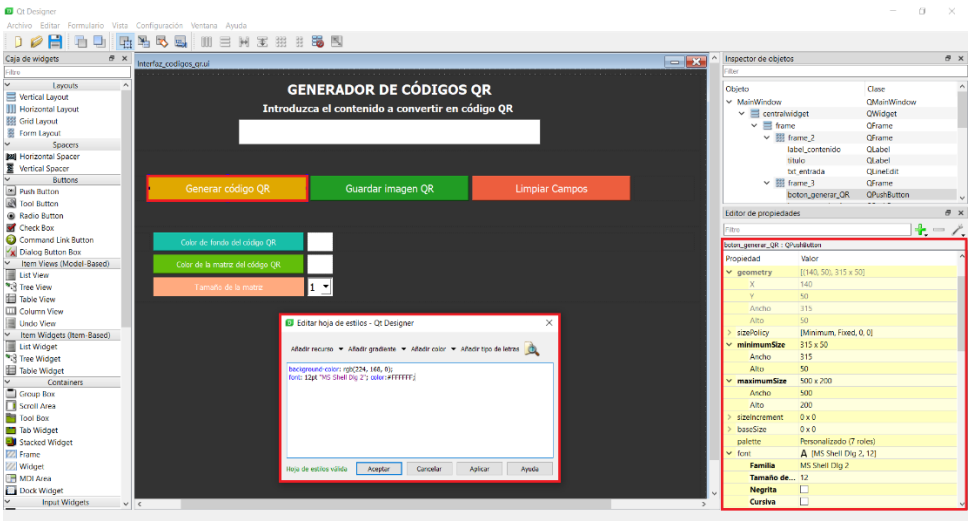


Figura 5. Interfaz de diseño en QtDesigner

Fuente: Autor

A fin de programar eventos en los widgets de la interfaz y pasar del diseño gráfico a código Python se utiliza PyQt5. "Un binding que permite integrar el diseño de la aplicación realizada en Qtdesigner (formato xml) y el lenguaje de programación Python" [24].

El binding PyQt5 al permitir migrar de un archivo *.xml* a un archivo Python, deja la tarea de establecer la lógica de programación de cada uno de los widgets y conectar a un evento de usuario. Para establecer eventos en la interfaz gráfica se establecieron métodos o funciones usando Python.

Para poder generar el código QR con base a la información que se aporte, se utilizaron las librerías: *pillow* para la representación del código QR y *qrcode* para su creación.

La librería *qrcode* posee parámetros de configuración que permite ajustarse a las necesidades y condiciones de diseño para el código QR como son:

- Version: permite establecer el tamaño del código QR y la cantidad de información que éste pueda contener, pudiendo variar su valor entre 1 a 40 como un número entero. Cada versión comprende 4 módulos adicionales por lado. La versión 1 equivale a una matriz de 21x21 módulos y la versión 40 a una matriz de 177x177.
- Error_correction: permite la corrección de errores del código QR. Esto en el caso de encontrarse en malas condiciones, manchado o expuesto en un lugar a la intemperie. Ante estas circunstancias se dispone de los siguientes cuatro niveles de corrección de errores:

  ERROR_CORRECT_L: nivel que permite corregir un 7% de errores o menos.

  ERROR_CORRECT_M: nivel que permite corregir un 15% de errores o menos.

  ERROR_CORRECT_Q: nivel que permite corregir un 25% de errores o menos.

  ERROR_CORRECT_H: nivel que permite corregir un 30% de errores o menos.

- Box_size: permite establecer el número de pixeles para la imagen del código QR.
- Border: permite establecer el ancho del borde del código QR.
- Fill_color y back_color: permiten establecer el color de la matriz de puntos del código QR y el color de fondo respectivamente.

Con base a los parámetros de configuración mencionados para generar los códigos QR se estableció un rango de versión de 1 a 5. Se escogió el error de corrección para recuperar el 25%, debido a la exposición que representará en las mesas, el valor del tamaño de caja de 10 y el border con un valor igual a 4. A fin de establecer un estilo personalizado de los códigos se implementaron métodos con una ventana de diálogo para la selección del color del código QR y el color de fondo.

Con el propósito de distribuir copias como una aplicación de escritorio, sin la necesidad de tener instalado Python se utilizó la librería *pyinstaller*, que permite crear un archivo de compilación (*.exe*) debido a que el sistema operativo en el cual fue codificado es Windows, mediante la ejecución del siguiente comando a través de consola:

```
pyinstaller  --windowed --onefile --icon=/logo.ico Codigos_QR_principal.py
```

El proceso a seguir por el usuario administrador (diagrama de flujo), para generar los códigos QR se puede observar en la Figura 6.

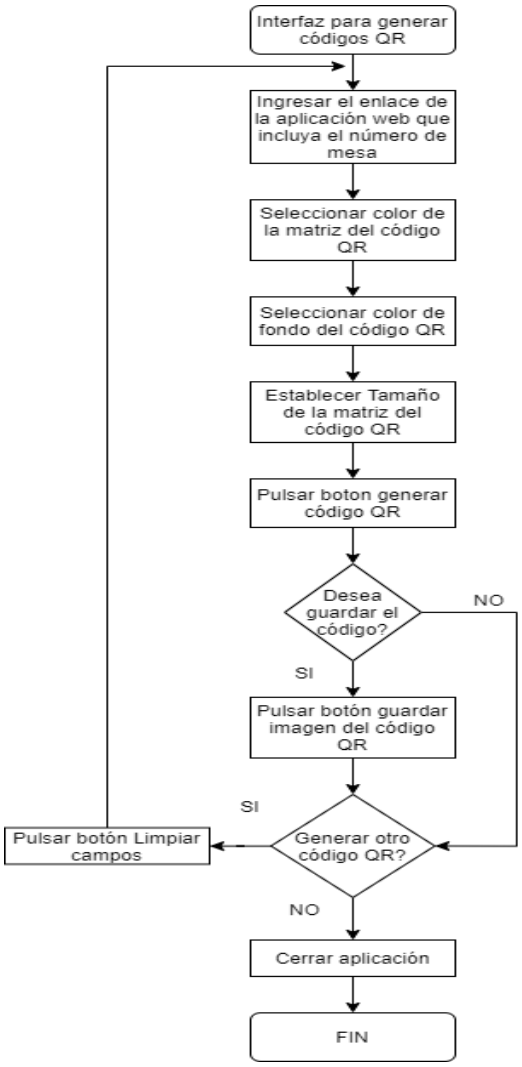


Figura 6. Diagrama de flujo para la generación de un código QR

Fuente: Autor

La interfaz gráfica para crear códigos QR se observa en la Figura 7.

Figura 7. Interfaz gráfica para generar códigos QR.

Fuente: Autor

2.4.2 PLATAFORMA DIGITAL

La plataforma digital se desarrolló con lenguajes de programación a nivel de Frontend y Backend.

El Frontend está encargado del modo de presentación de la información a los usuarios y es todo lo que hace posible la navegación web a través de la pantalla de su dispositivo inteligente.

El Backend contiene la lógica de programación que interactúa con el servidor web y las bases de datos.

FRONTEND

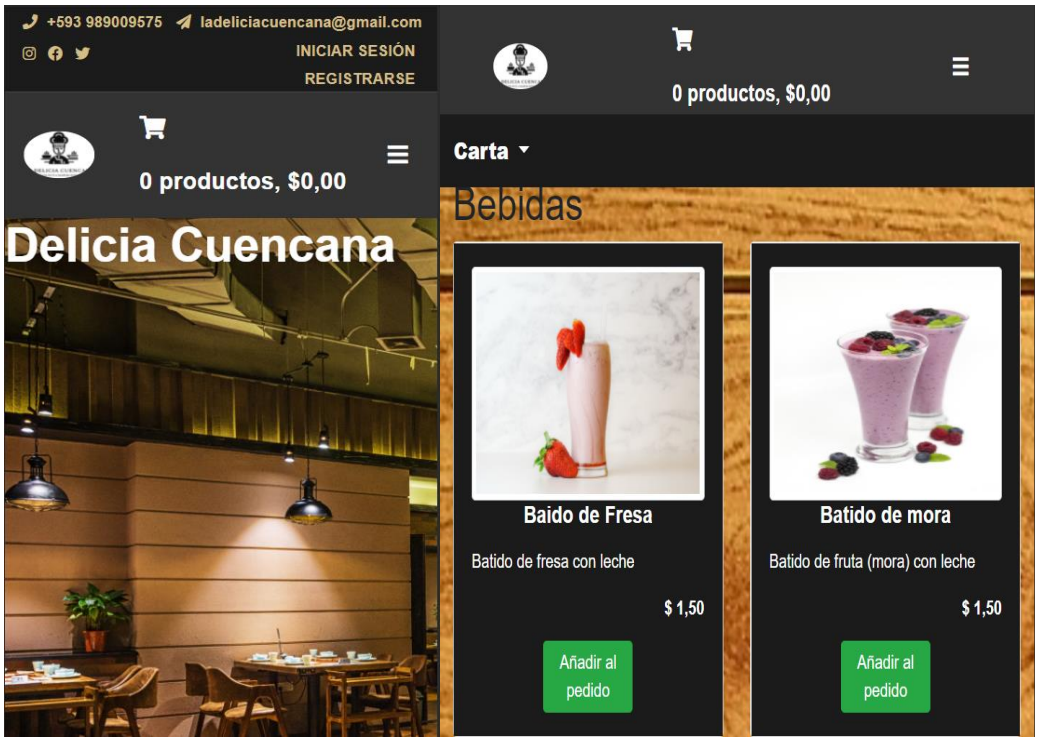
Los lenguajes que integran el desarrollo de las aplicaciones web son: HTML, CSS, Bootstrap y JavaScript. Para el desarrollo de la plataforma digital (restaurante) se consideraron tecnologías que permitan organizar y diseñar el contenido visible en la web.

En esta ocasión se utilizaron: la versión HTML5, el lenguaje de maquetación CSS3 y Bootstrap.

El lenguaje HTML permite indicar el tipo de contenido en la web mediante etiquetas, es decir, permite establecer la estructura del contenido.

Se empleó el lenguaje de estilos CSS3 que permite establecer la estética, apariencia y estilo de la información (maquetación web) contenida en la plantilla HTML (ver Figura 8 (a)).

Se trabajó con la versión de Bootstrap 4.6 que permite establecer el estilo responsive del contenido web (ver Figura 8 (b)), haciendo uso de clases predefinidas mediante CSS y JavaScript fue posible acoplarse a diferentes tamaños de pantalla de los dispositivos móviles.



(a) (b)

Figura 8. (a) Maquetación del sitio web; (b)Menú del restaurante aplicado un estilo responsive

Fuente: Autor

Para la maquetación web se emplearon íconos para el diseño de la cabecera de la interfaz de presentación, menú y carrito de compras (ver Figura 8a), haciendo uso del Script para desarrollo web Fontawesome, que está basado en lenguaje CSS.

Para la colección de archivos estáticos como imágenes y videos se utilizaron bancos de imágenes como Pixabay y Artgrid, por ser de acceso libre.

BACKEND

Los lenguajes de programación más comunes que pueden integrar el Backend son: Python, PHP, Java, Ruby entre otros. Para este fin se usó Python 3.8.3 y Framework Django, el cual permite el desarrollo de proyectos web manteniendo el código de forma estructurada para el desarrollo, edición y mantenimiento utilizando el patrón: modelo, vista, plantilla como se muestra en la Figura 10.

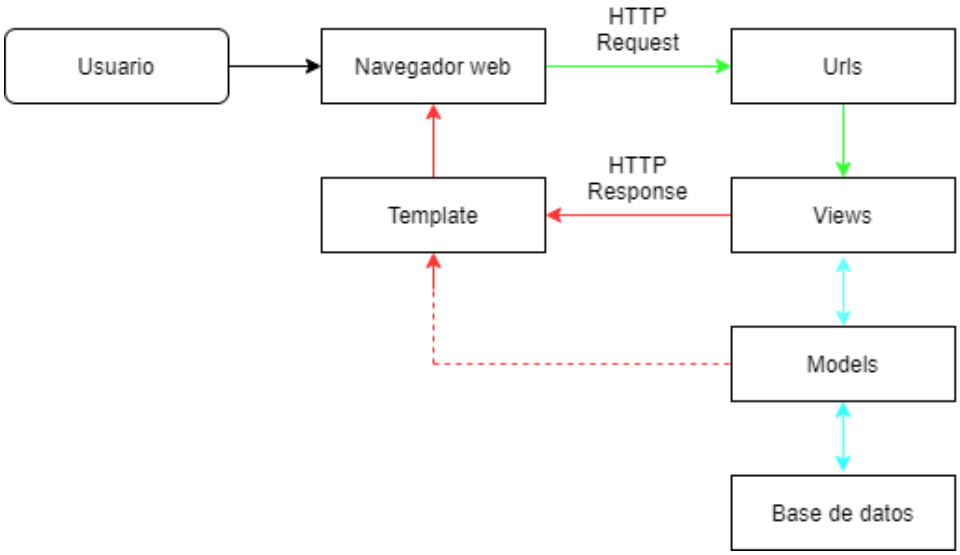


Figura 10. Diagrama de flujo del modo de operación de Django.

Fuente: Autor

Como se puede observar, el usuario realiza una petición HTTP Request al solicitar una dirección url mediante el navegador web. Posteriormente, Django realiza una búsqueda en las urls configuradas del proyecto web y si el resultado es exitoso se ejecuta la vista (función) correspondiente. La vista se relaciona con el modelo (Models) para consultar información a la base de datos y para adjuntar los datos a la plantilla (Template) que son documentos HTML. Esto permitirá visualizar el usuario mediante el navegador a través de una respuesta HTTP Response por parte del servidor.

2.4.3 ESTRUCTURA DEL PROYECTO WEB

Para la construcción del proyecto en Django, se implementó un entorno virtual de Python con el propósito de establecer un control de versiones de los programas, paquetes y librerías necesarias que fueron instaladas. El uso del entorno virtual permite la distribución de código fuente y los paquetes necesarios para ejecutar el entorno en diversas computadoras, indistintamente de la versión de sus programas.

En el proyecto se trabajaron con los siguientes archivos de extensión .py:

- `__init__.py` : para que Python interprete el directorio como un solo módulo.
- `asgi.py/wsgi.py`: son archivos de configuración necesarios para implementar el proyecto en un servidor web en producción.
- `settings.py`: contiene todas las configuraciones para el desarrollo del proyecto, entre ellas se puede citar a la base de datos, aplicaciones que conforman el proyecto, idioma, zona horaria, servidor de correos electrónicos, dirección de los archivos estáticos entre otros.
- `manage.py`: permite ejecutar funciones del proyecto para la creación de aplicaciones, ejecutar migraciones para actualizar la base de datos y el servidor local que incluye Django para su desarrollo.
- `urls.py`: este archivo contiene todas las direcciones URL del proyecto, incluidas a las de las aplicaciones web que lo constituyen.

2.4.3.1 Aplicaciones web

Las aplicaciones de Django se crearon de acuerdo con las exigencias y demanda del proyecto, permitiendo estructurar el código por partes y hacer reutilizables en más de un proyecto. Django, al permitir crear aplicaciones, puede incorporar diversas funcionalidades para cada una, esto fomenta el desarrollo rápido, permite estructurar el código y a realizar posteriores modificaciones.

Las aplicaciones web en Django están basadas en los siguientes archivos:

- `__init__.py`: para que Python interprete el directorio como un archivo de Python.
- `admin.py`: permite exportar las clases de los modelos de la aplicación a la al sitio de administración.
- `apps.py`: archivo de configuración para la aplicación.
- `migrations/`: es un directorio en donde se almacenan las migraciones de base de datos de la aplicación y depende de las modificaciones que se realicen en el archivo models.py.
- `models.py`: aquí se encuentra todas las clases de modelos creados que se convertirán en tablas de la base de datos.

- `tests.py`: permite verificar el código y lógica de programación de la aplicación para detectar posibles errores.
- `urls.py`: archivo para el manejo de las URLS de cada aplicación que posteriormente serán manejadas por el archivo de Django Urlconf, desde el proyecto principal.
- `views.py`: en este archivo se encuentra la lógica de programación con base a las exigencias del proyecto, compuesta por funciones que usan a los modelos.

En la Tabla 6 se detallan las aplicaciones utilizadas para la ejecución del proyecto.

Tabla 6. Aplicaciones utilizadas en el proyecto.

APLICACIONES DEL PROYECTO		
	Nombre de la aplicación	**Descripción**
Aplicaciones integradas en la creación del proyecto	django.contrib.admin	Sitio o interfaz de administración
	django.contrib.auth	Sistema que permite la autenticación y permisos de usuario
	django.contrib.contenttypes	Aplicación para manejar el contenido por su tipo
	django.contrib.sessions	Marco de trabajo que permite manejar las sesiones de usuario
	django.contrib.messages	Marco de trabajo que permite establecer mensajes flash
	django.contrib.staticfiles	Marco de trabajo para configurar los archivos estáticos (imágenes, videos, iconos)
Aplicaciones para la lógica de negocio	autenticacion.apps.AutenticacionConfig	Aplicaciones para registrar al usuario, presentar el menú del restaurante, carrito de compras y envió del orden
	menu.apps.MenuConfig	
	carrito.apps.CarritoConfig	
	pedidos.apps.PedidosConfig	
Aplicaciones de terceros adicionadas al proyecto	crispy_forms	Permite renderizar los formularios haciendo uso de Bootstrap
	import_export	Facilita exportar el contenido de un modelo desde la interfaz de administración
	colorfield	Librería que permite establecer los temas de la interfaz de administración
	Xhtml2pdf	Librería que permite convertir una plantilla html a pdf (usado para la factura)

Las funcionalidades del proyecto web están compuestas por las siguientes aplicaciones:

a) menú
b) carrito
c) pedidos
d) autenticación

Estas opciones están integradas en todo el proceso de navegación para realizar un pedido en la plataforma digital; para lo cual se hace uso de un panel de navegación descrito en el diagrama de la Figura 11.

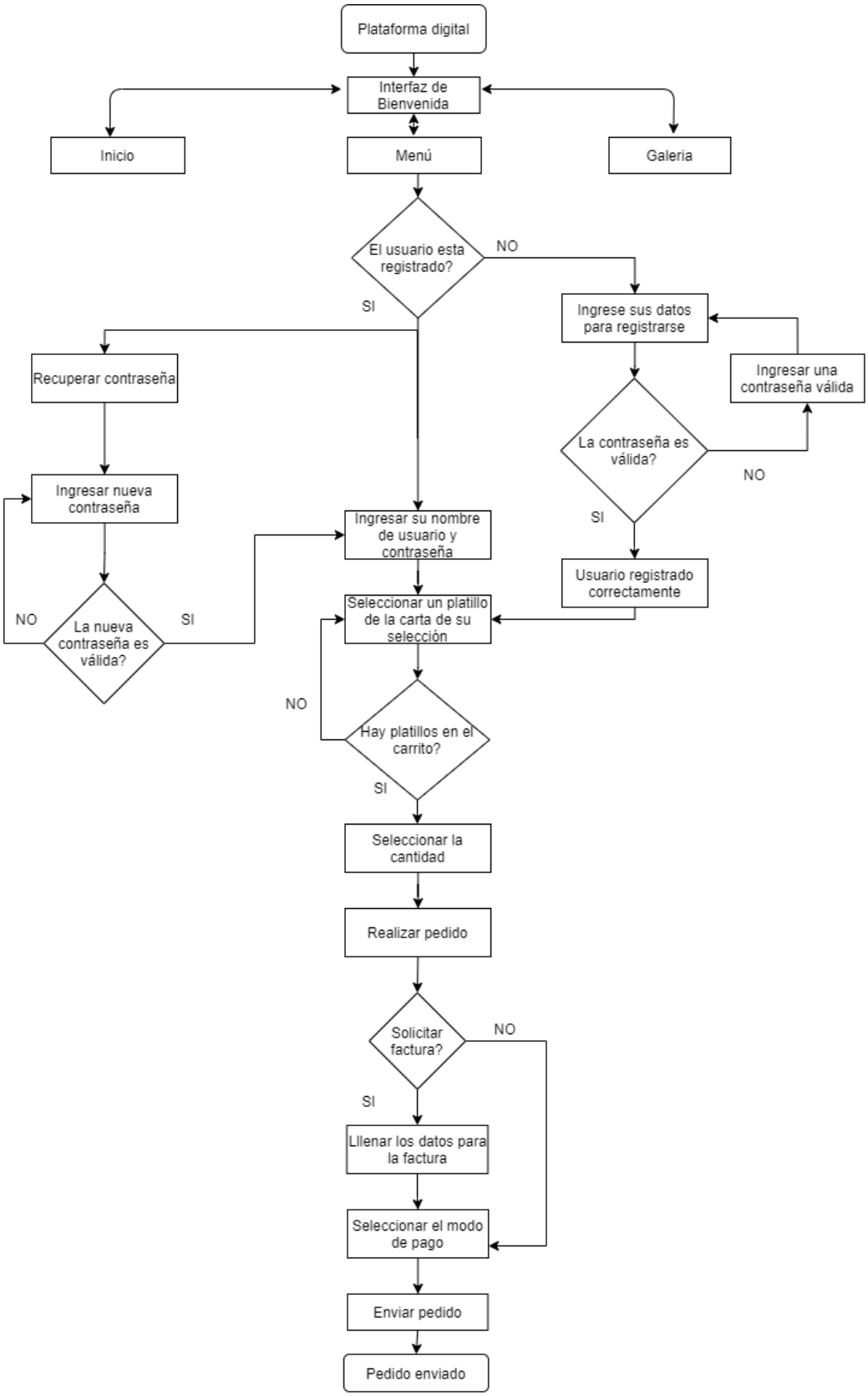


Figura 9. Diagrama de flujo de la plataforma digital

Fuente: Autor

Las aplicaciones creadas en el proyecto web tienen inicio con la aplicación llamada Menú; la cual contiene la vista necesaria (lógica de programación) para acceder a la interfaz de bienvenida que a su vez abarca el panel de navegación (inicio, menú y galería). Para acceder al menú del restaurante es necesario que el usuario realice un registro previo y/o inicie sesión.

A fin de comprender la utilidad de cada una de las aplicaciones a continuación se especifican sus funcionalidades, propósitos y lógica de programación.

- **APLICACIÓN MENÚ**

Esta aplicación tiene como objetivo permitir a los usuarios conocer el menú que ofrece el local (restaurante), pudiendo filtrar la búsqueda por el tipo de carta (mariscos, asados, bebidas, postres, etc.).

Para poder almacenar esta información en la base de datos se creó una clase para el tipo de carta y otra para los platillos en el archivo *models.py*.

En la clase "*tcarta*" de la Figura 11, se establecen los campos como: *name* que permite asignar el nombre del tipo de carta que se quiera adicionar mediante el sitio de administración, además se estableció el campo *slug* que se autocompletará con base al nombre que se asigne a la nueva carta. Este campo slug es útil para las direcciones urls ya que haciendo uso de la función *get_absolute_url* en la línea 18 de la Figura 11, se puede asignar ese nombre a la direccion url que redirija al tipo de carta seleccionado. Esto evita la tarea de disponer de varias direcciones url en la plantilla html.

La clase *meta* de los modelos permite establecer el nombre de la tabla en la base de datos, además de su nombre en singular, plural y ordenamiento de la información.

```
5   class Tcarta(models.Model):
6       name = models.CharField(max_length=250, verbose_name ='Nombre', db_index=True)
7       slug = models.SlugField(max_length = 250, unique = True)
8
9       class Meta:
10          db_table = ' Carta'
11          verbose_name = ' tcarta '
12          verbose_name_plural = 'cartas'
13          ordering = ['id']
14
15      def __str__(self):
16          return self.name
17
18      def get_absolute_url(self):
19          return reverse('menu:lista_platos_por_tcarta', args=[self.slug])
```

Figura 10. Modelo para los tipos de carta del restaurante.

Fuente: Autor

La clase denominada *Plato* (ver Figura 12) dispone de campos para establecer las características del platillo. Entre ellas se destaca el campo disponibilidad que permite postear o quitar un platillo del menu haciendo uso del sitio de administración. Al campo *tcarta* se le estableció una relación de base de datos *Foreignkey* (uno a muchos) con la tabla *Tcarta* antes citada, esto permite asignar los platillos a un tipo de carta.

```
22  class Plato(models.Model):
23      tcarta = models.ForeignKey(Tcarta, related_name='platos', verbose_name ='tipo de carta', on_delete=models.CASCADE)
24      name = models.CharField(max_length=250, verbose_name ='Nombre')
25      slug = models.SlugField(max_length = 250, db_index= True)
26      imagen = models.ImageField(upload_to = 'tipo_plato/%Y/%m/%d', null=True, blank= False)
27      ingredientes = models.TextField(max_length=300, verbose_name= 'descripción del plato')
28      pvp = models.DecimalField(max_digits=4, decimal_places=2)
29      disponibilidad = models.BooleanField(default=True)
30
31      class Meta:
32          db_table = 'Platos'
33          verbose_name = 'Plato'
34          verbose_name_plural = 'Platos'
35          ordering = ['name']
36          index_together = (('id', 'slug'),)
37
38      def __str__(self):
39          return self.name
```

Figura 11. Modelo para los platillos del restaurante.

Fuente: Autor

Para poder presentar el menú al usuario se hace uso de la función *lista_platos* (ver Figura 13). En esta función se accede a la base de datos para consultar los tipos de carta que tiene el restaurante y filtrar la disponibilidad de los platillos haciendo uso de los modelos antes definidos, posteriormente se renderiza una plantilla html (ver línea 16 del código).

```
 9  def lista_platos(request, tcarta_slug=None):
10      tcarta = None
11      cartas = Tcarta.objects.all()
12      platos = Plato.objects.filter(disponibilidad = True)
13      if tcarta_slug:
14          tcarta = get_object_or_404(Tcarta, slug = tcarta_slug)
15          platos = platos.filter(tcarta = tcarta)
16      return render(request, 'menu/plato/lista.html', {'tcarta':tcarta, 'cartas':cartas, 'platos':platos})
```

Figura 12. Función lista_platos para mostrar el menú del restaurante

Fuente: Autor

Especificado el objetivo de esta aplicación, cabe mencionar que se trabajó en ella para la interfaz de bienvenida al usuario que incluye los menús de navegación que son: Inicio, Menú y Galería.

Para los ítems Inicio y Galería del menú de navegación se definieron funciones que únicamente permiten renderizar una plantilla html con contenido estático (imágenes, videos, texto, etc.) y para el ítem Menú se usa la función *lista_platos* antes mencionada.

A cada función se le debe asignar una dirección url, para lo cual se importan las *vistas* al archivo *urls.py* (ver Figura 14).

```
5   app_name = 'menu'
6   urlpatterns = [
7       path('galeria_restaurante/', views.galeria, name = 'galeria_restaurante'),
8       path('<int:nmesa>/', views.ingreso, name = 'presentacion'),
9       path('bienvenid@', views.inicio, name = 'inicio'),
10      path('lista_platos/', views.lista_platos, name = 'lista_platos'),
11      path('<slug:tcarta_slug>/', views.lista_platos, name = 'lista_platos_por_tcarta'),
```

Figura 13. Direcciones urls de la aplicación menú

Fuente: Autor

- **APLICACIÓN AUTENTICACIÓN**

Para el proceso de autenticación, registro y cambio de contraseñas de usuario, se utilizaron vistas basadas en clases mediante el sistema de autenticación *'django.contrib.auth'*.

Las clases que emplea este sistema de control son las siguientes: "*LoginView, LogoutView, PasswordChangeView, PasswordChangeDoneView, PasswordResetView, PasswordResetDoneView, PasswordResetConfirmView, PasswordResetCompleteView"*.

Para autenticar al usuario se emplea la clase *authenticate,* que contrasta el nombre de usuario y la contraseña para validar si el usuario ya tiene una cuenta. Además se hace uso de login para iniciar la sesión de usuario.

Para las clases de registro y cambio de contraseña de usuario se crearon plantillas Html.

El diseño de los formularios se realizó con la aplicación Django Crispy Forms que usa Bootstrap para la maquetación del formulario (ver Figuras 15(a) y 15(b)).

(a) (b)

Figura 14. (a) Formulario de registro de usuario; (b) Formulario de inicio de sesión.

Fuente: Autor

- **APLICACIÓN CARRITO**

Esta aplicación contiene la lógica de programación para llevar a cabo el proceso de añadir productos al carrito de compras, pudiéndose ejecutar operaciones como: agregar, disminuir y eliminar productos que se encuentren en la sesión actual del usuario. La interfaz del carrito de compras se la puede apreciar en la Figura 16.

Figura 15. Interfaz de presentación del proceso de añadir productos al carrito.

Fuente: Autor

Con la finalidad de limitar los permisos de usuario en el menú del restaurante y en el carrito de compras se utilizó el decorador *login_required*. Esta es una función de Django que permite validar si el usuario se ha registrado e iniciado sesión. En caso de cumplir con esta condición, redirecciona al usuario a la url programada (ver el menú y los platillos agregados al carrito de compras). Si el usuario no cumple con esta condición se lo redirecciona a la *vista* de registro. Para hacer uso de esta función es necesario incluir el decorador antes de la función que condicione al usuario para su ingreso (ver Figura 17).

```
@login_required(login_url='/login')
def carrito_agregar(request, plato_id):
    carrito = Carrito(request)
    plato = Plato.objects.get(id=plato_id)
    carrito.agregar(plato=plato)
    return redirect('carrito:detalle_carrito')
```

Figura 16. Decorador login_required para permisos de usuario

Fuente: Autor

Sesión de usuario

Para tener un control de la sesión de usuario y conservar las opciones agregadas al carrito de compras, se construyó un archivo *carrito.py* dentro del directorio de la aplicación, en el cual se elaboró una clase *Carrito* con las funciones que se detallan en la Tabla 7.

Tabla 7. Funciones para el carrito de compras.

Función	Objetivo
__init__	Verificar si existe una sesión actual del usuario, caso contrario crear una.
agregar	Adicionar productos al carrito o incrementar la cantidad en caso de existir en la sesión actual.
disminuir	Restar los productos del carrito de compras.
guardar	Guardar en la sesión, el proceso que se realice en las demás funciones.
eliminar	Eliminar el producto sin importar la cantidad actual que se encuentre guardada en la sesión de usuario.
limpiar	Vaciar el carrito de compras.

Procesadores de Contexto

En esta aplicación se hace uso de procesadores de contexto para dar a conocer al usuario la información del carrito de compras (cantidad de productos y precio total a cancelar) de la sesión actual. Los procesadores de contexto son funciones que permiten guardar valores temporales, pudiendo hacer uso en cualquier plantilla html del proyecto, funcionando como una variable global (ver Figura 18).

```python
def carrito_cant_total(request):
    total = 0.0
    if 'carrito' in request.session:
        for key, value in request.session['carrito'].items():
            total = total + (float(value ['pvp']) * value['cantidad'])
    return {'carrito_cant_total': total}

def cant_total(request):
    cant =0
    if 'carrito' in request.session:
        for key, value in request.session['carrito'].items():
            cant = cant + int(value ['cantidad'])
    return {'cant_total': cant}
```

Figura 17. Procesadores de contexto para la cantidad de productos y precio total.

Fuente: Autor

Los procesadores de contexto se deben incluir en la variable *Templates* del archivo de configuración del proyecto como:

```
'carrito.context_processors.carrito_cant_total'
'carrito.context_processors.cant_total'
```

- **APLICACIÓN PEDIDOS**

Esta aplicación tiene como propósito enviar los pedidos gestionados por parte del usuario. Los usuarios luego de haber agregado sus pedidos al carrito de compras pueden elegir entre generar la factura con sus datos o como consumidor final. Indistintamente de la selección se envía el reporte de compra (con datos o consumidor final) a través de correo electrónico. Además, se disponen de opciones para seleccionar la forma a pagar (efectivo, tarjeta de crédito) y vajilla adicional (cucharas, vasos, tenedores) previo al envío del pedido (ver Figura 19).

Figura 18. Formulario para datos de factura, tipo de pago y vajilla adicional

Fuente: Autor

Para guardar la información del formulario se usa la clase *Pedido* (ver Figura 20), que permite aportar datos para el envío de la factura y opciones extras como el tipo de pago y vajilla adicional.

En esta clase se definió un campo denominado *"mesa"*, en el que se guarda el número de mesa que se recibe como parámetro en la dirección url de bienvenida al restaurante, información que no es visible para el usuario.

Con base al modelo definido *Pedido* se pueden designar los campos que se quieren mostrar al usuario con la clase *forms* de Django, para lo cual se creó el archivo *forms.py*. En este archivo se crearon dos clases para los pedidos con factura y sin factura.

```
15  class Pedido(models.Model):
16      User._meta.get_field('email')._unique = True
17      Usuario = models.ForeignKey(settings.AUTH_USER_MODEL, on_delete=models.CASCADE)
18      Nombre = models.CharField(max_length=50, blank=True)
19      Apellido = models.CharField(max_length=50, blank=True)
20      Email = models.EmailField(blank=True)
21      Direccion = models.CharField(max_length=100, blank=True)
22      teléfono=models.CharField(max_length=10, blank=True)
23      CI_RUC = models.CharField(max_length=13, blank=True)
24      Fecha = models.DateTimeField(auto_now_add=True)
25      pago = models.BooleanField(default=False)
26      mesa = models.IntegerField(default=get_nmesa)
27      Tipo_de_pago = MultiSelectField(max_length=10, choices=opcion_pago)
28      cucharas = models.PositiveIntegerField(default=0)
29      cubiertos = models.PositiveIntegerField(default=0)
30      vasos = models.PositiveIntegerField(default=0)
31      img = models.ImageField(default='media/Logo.png/')
```

Figura 19. Modelo Pedido perteneciente a la aplicación pedidos.

Fuente: Autor

En esta aplicación se usa la clase *ListadoPedido* (ver Figura 21), que tiene una relacion *Foreignkey* con el modelo *Plato* de la aplicación *menu* y una relación del mismo tipo con el modelo *Pedido* antes mencionado. Esto permite, mediante la interfaz de administracion, conocer información de los *platillos* de la orden e información del usuario.

```
48  class ListadoPedido(models.Model):
49      Fecha_pedido = models.DateTimeField(auto_now_add=True)
50      pedido = models.ForeignKey(Pedido, related_name='items', on_delete=models.CASCADE)
51      plato = models.ForeignKey(Plato, related_name='pedido_items', on_delete=models.CASCADE)
52      pvp = models.DecimalField(max_digits=4, decimal_places=2)
53      cantidad = models.PositiveIntegerField(default=1)
```

Figura 20. Modelo ListadoPedido perteneciente a la aplicación pedidos.

Fuente: Autor

- **Factura**

Con el propósito de dar a conocer al usuario el informe de compra, se crearon funciones que permitan obtener valores de iva, subtotal y total. También se configuró un servidor de correos electrónicos con gmail, cuya configuración es la siguiente:

```
ADMINS=(('La Delicia Cuencana', 'ladeliciacuencana@gmail.com'),
('Javier Cordero', 'edumasflow@gmail.com'),)
“EMAIL_HOST” = 'smtp.googlemail.com'
“EMAIL_PORT” = 587
“EMAIL_HOST_USER” = 'ladeliciacuencana@gmail.com'
“EMAIL_HOST_PASSWORD” = 'contraseña'
“EMAIL_USE_TLS” = True
“EMAIL_BACKEND” = 'django.core.mail.backends.smtp.EmailBackend'
“EMAIL_SUBJECT_PREFIX” = '[Django]'
```

Para el reporte de la factura se utilizó la librería *xhtml2pdf* que permite generar un archivo pdf con base a una plantilla Html (ver Figura 22).

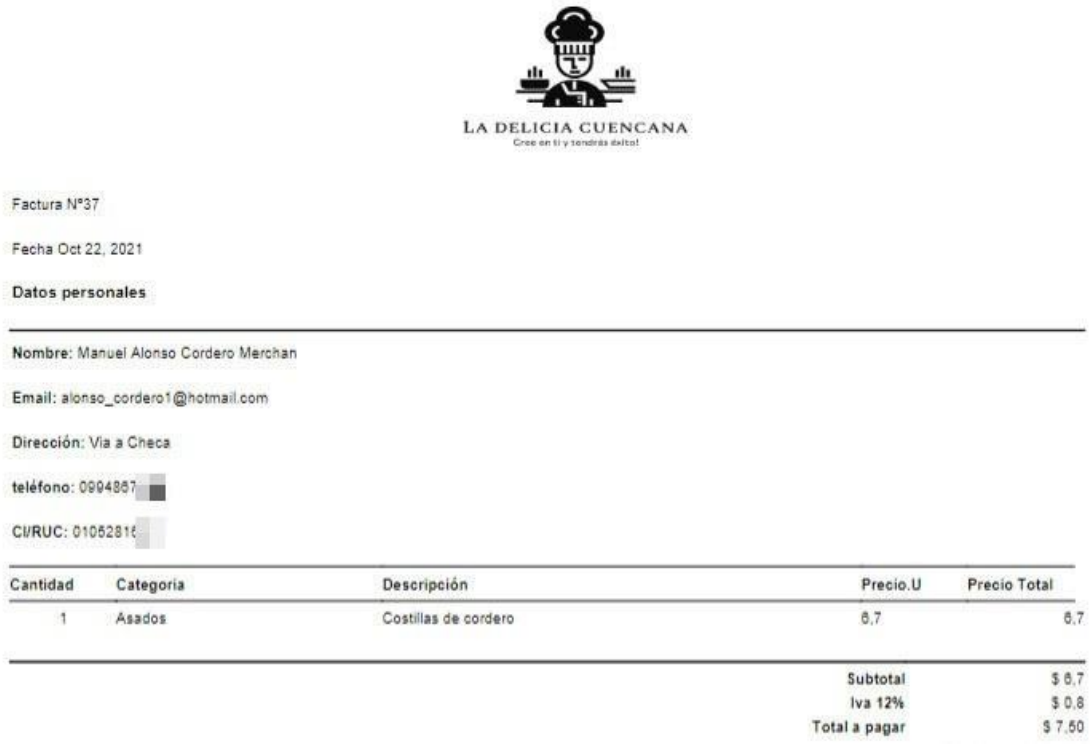


Figura 21. PDF de la factura del cliente.

Fuente: Autor

Identificación del número de mesa del restaurante

Para establecer permisos de usuario en el restaurante se utilizaron urls pasando parámetros a fin de identificar la mesa en la que se encuentra el usuario.

El parámetro que se usa en la url es de tipo entero y puede variar entre 1 y 10.

```
path('<int:nmesa>/', views.ingreso, name = 'presentacion')
```

2.4.4 INTERFAZ PARA ADMINISTRACIÓN

Django dispone de una interfaz de administración haciendo uso de los modelos de las aplicaciones creadas. Para usar esta interfaz se debe instalar la aplicación *django.contrib.admin* en el archivo *settings.py* del proyecto. Esta aplicación trabaja en conjunto con las aplicaciones "*django.contrib.sessions*, *django.contrib.auth*, y *django.contrib.contenttypes"*.

Para hacer uso de esta interfaz se deben importar los modelos de cada aplicación a su archivo *admin.py* ubicado en el mismo directorio de *models*.

En la Figura 23 se presenta el código perteneciente a la Aplicación Menú.

```
from django.contrib import admin
from .models import Tcarta, Plato
@admin.register(Tcarta)
class CartasAdmin(admin.ModelAdmin):
    prepopulated_fields = {'slug': ('name',)}
    list_display = ['name', 'slug']

@admin.register(Plato)
class PlatosAdmin(admin.ModelAdmin):
    list_display = ['name', 'pvp', 'ingredientes', 'disponibilidad', 'tcarta']
    list_filter = ['disponibilidad', 'tcarta', 'pvp']
    list_editable = ['pvp', 'disponibilidad', 'ingredientes', 'disponibilidad']
    search_fields = ['name']
    prepopulated_fields = {'slug': ('name',)}
```

Figura 22. Registro de modelos en la interfaz de administración de Django

Fuente: Autor

Para que los modelos de datos puedan ser visibles en el sitio de administración se usa la propiedad @admin.register("nombre del modelo"), que permite registrar dichos modelos. En los modelos importados se pueden especificar propiedades de campos que pueden ser modificados para establecer modos de uso.

En la Tabla 8 se indican las propiedades utilizadas.

Tabla 8. Propiedades para modelos del sitio de administración

Propiedad	Objetivo
prepopulated_fields	Establecer de forma automática una dirección url con base al *slugfield.*
list_display	Establecer los campos que serán visibles en la interfaz de administración.
list_editable	Establecer todos los campos que podrán ser modificados.
list_filter	Filtrar información con base a los campos establecidos en la lista.
search_fields	Realizar búsquedas de acuerdo a los campos que se establezcan mediante una lista.

Django permite importar los meta-datos de los modelos de las aplicaciones que constituye el proyecto web para el sitio de administración, para gestionar el contenido permitido para los usuarios y otorgar permisos a los usuarios administradores. La estructura y diagrama de flujo del sitio administrativo se puede ver en la Figura 24. Los procesos como crear, editar y eliminar están permitidos para usuarios administradores no técnicos con permisos ya sea de superusuario (usuario administrador con todos los permisos) o staff (usuario administrador con permisos restringidos).

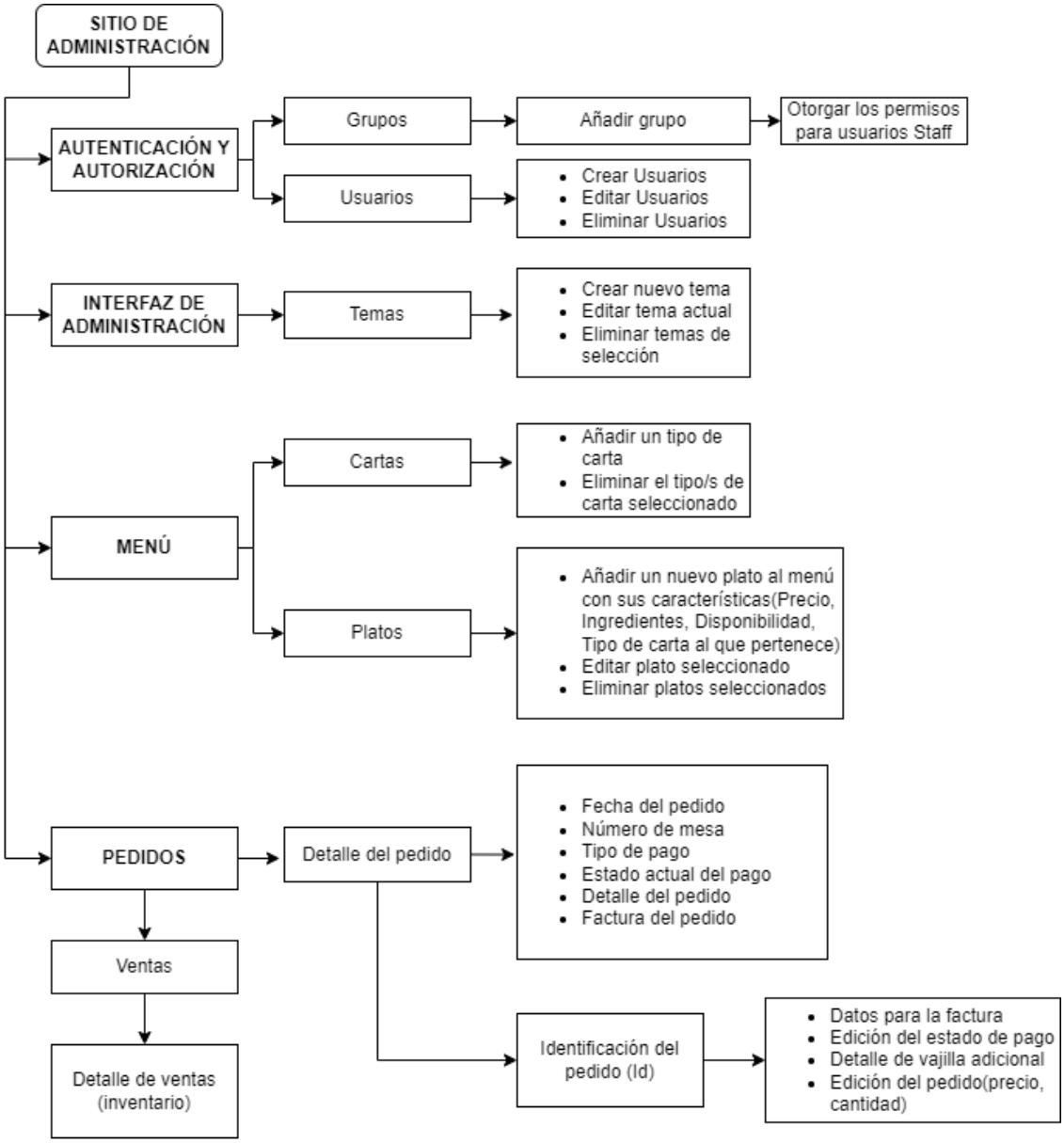


Figura 23. Diagrama de flujo de la interfaz de administración de la plataforma digital.

Fuente: Autor

En la Figura 25 se muestra la interfaz de administración de la plataforma digital.

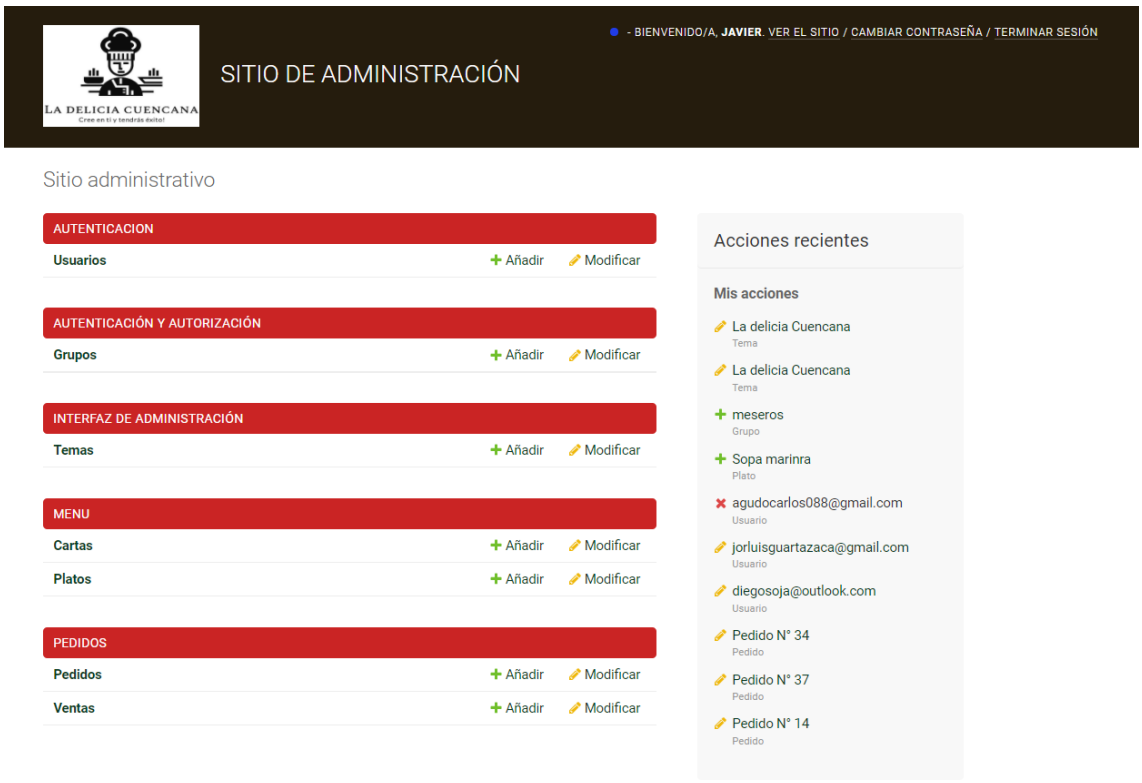


Figura 24. Sitio para la administración.

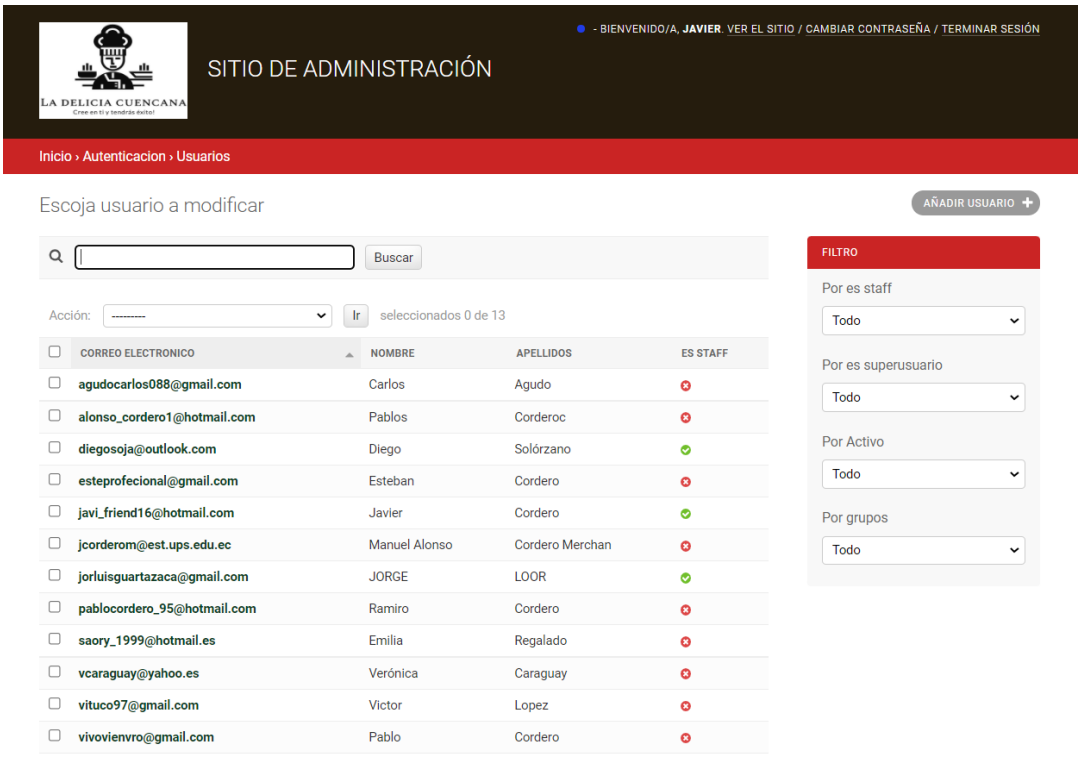
Fuente: Autor

- **AUTENTICACIÓN Y AUTORIZACIÓN**

A través de la sección de autenticación y autorización se pueden gestionar accciones sobre los usuarios registrados permitiendo realizar clasificaciones por grupos y crear nuevos usuarios, como se puede observar en la Figura 26 (a).

Es posible asignar permisos a los usuarios administradores sobre los recursos que se pudieren crear, modificar y eliminar (ver Figura 26 (b)).

(a)



(b)

Figura 25. (a) Permisos de usuario administrador por grupos. (b) Usuarios registrados y permisos otorgados

Fuente: Autor

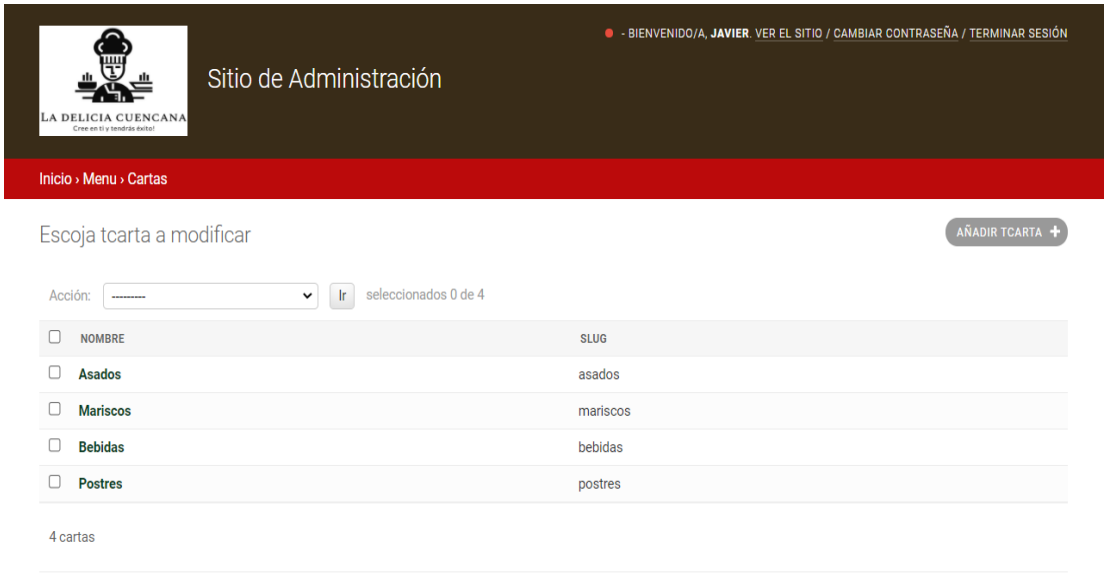
- **TEMAS**

La sección de temas permite la maquetación del sitio de administración permitiendo establecer reglas de presentación sobre la información (colores de

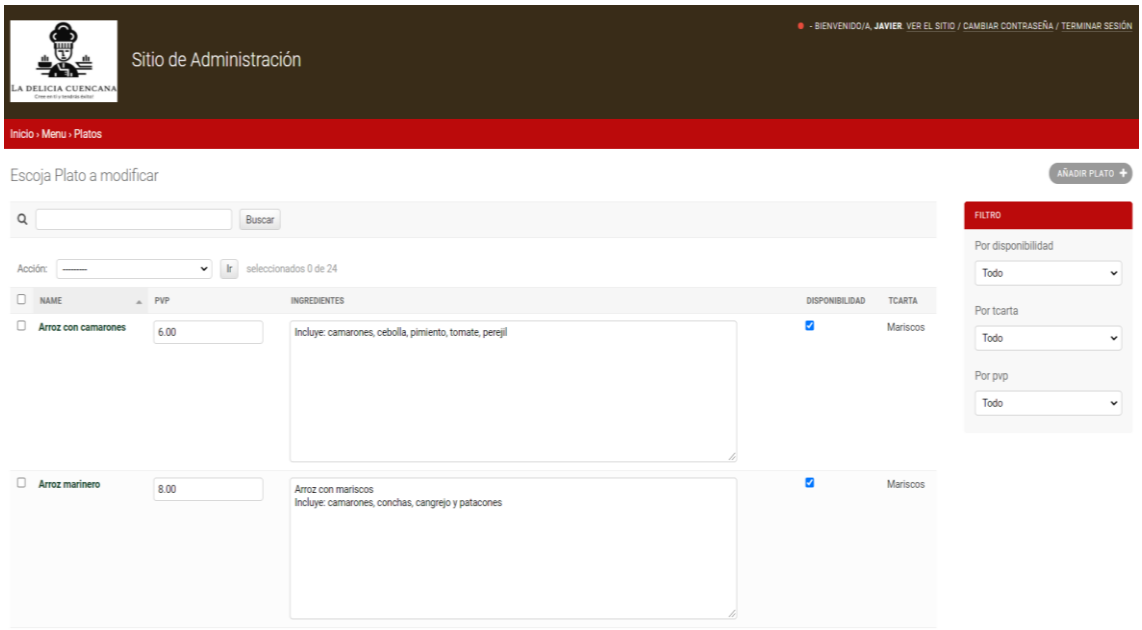
encabezados, botones, texto, estilos, entre otros) para lo cual se instalaron los paquetes de soporte *admin_interface* y *colorfield*.

- **MENÚ**

Sección de administración en la cual se puede clasificar por tipo de carta La opciones del restaurante (ver Figura 27 (a)). En esta sección también se establecen las características de los platillos (nombre, precio, ingredientes, disponibilidad), lo cual se puede visualizar en la Figura 27 (b).



(a)



(b)

Figura 26. (a) Sección de tipos de carta que oferta el restaurante,

(b) Sección de platos que ofrece el restaurante con sus características

Fuente: Autor

- **PEDIDOS**

En esta sección se puede visualizar (Figura 28) información sobre todos los pedidos gestionados por parte de los usuarios (clientes).

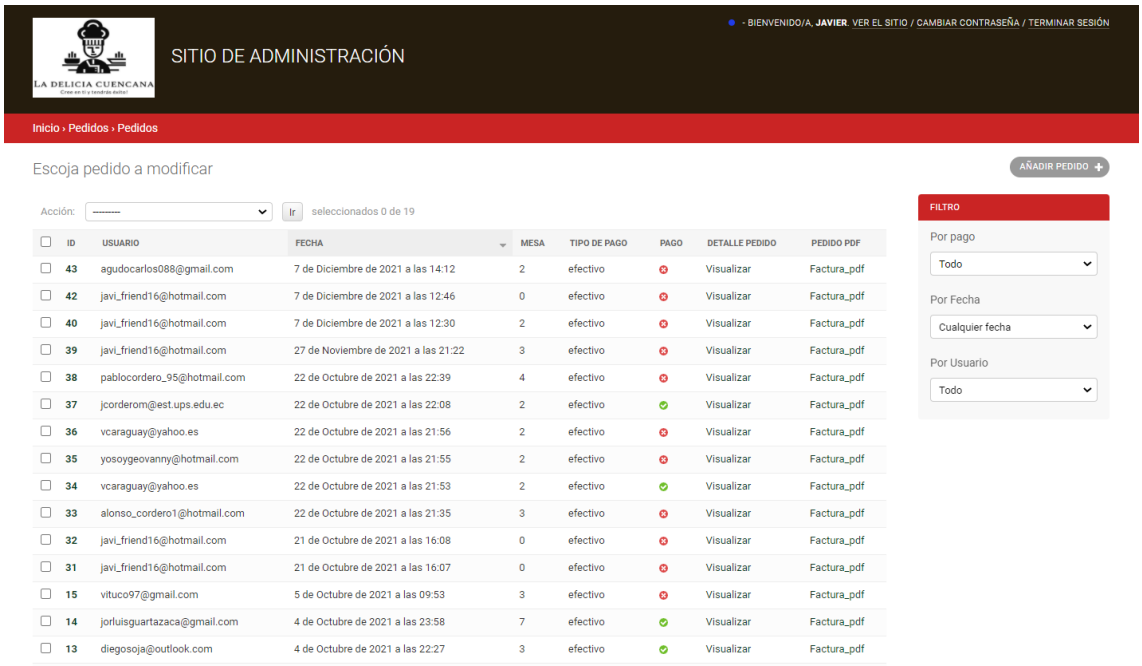


Figura 27. Interfaz de los pedidos gestionados por los usuarios (clientes)

Fuente: Autor

La sección de pedidos también está programada para que se pueda entregar de forma impresa la factura si así se desea, además de poder revisar el detalle del pedido que incluye el número de mesa, fecha y estado del pago lo cual se puede observar en la Figura 29.

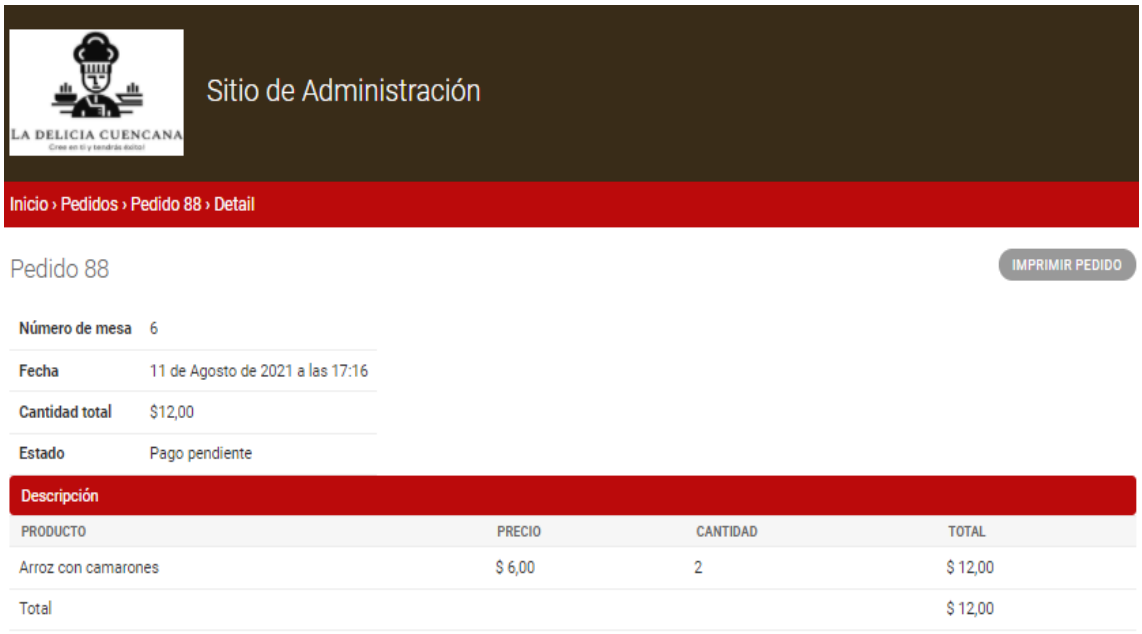


Figura 28. Detalle de un pedido de usuario

Fuente: Autor

2.4.5 SISTEMA GESTOR DE BASE DE DATOS

La base de datos empleada para el proyecto fue MySQL debido a que soporta gran volumen de datos, es multiplataforma y porque cuenta con una excelente velocidad de respuesta [22].

Para vincular la base de datos con el proyecto Django se establecieron los siguientes parámetros de configuración:

```
“DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'restaurante1',
        'USER': 'nombre de usuario',
        'PASSWORD': 'clave de la base de datos',
        'HOST': '127.0.0.1',
        'PORT': 3306,
    }
}”
```

DIAGRAMA ENTIDAD RELACIÓN

El diagrama entidad relación perteneciente a la base de datos con base al proyecto construido en Django se muestra en la Figura 30.

Una de las grandes ventajas que ofrece Django es el ORM (Mapeo de Objeto Relacional) que mediante la información contenida en el archivo models.py evita involucrarse en el lenguaje SQL. ORM permite además elaborar las tablas de la base de datos mediante la definición de clases y los atributos de la clase representan columnas de la tabla.

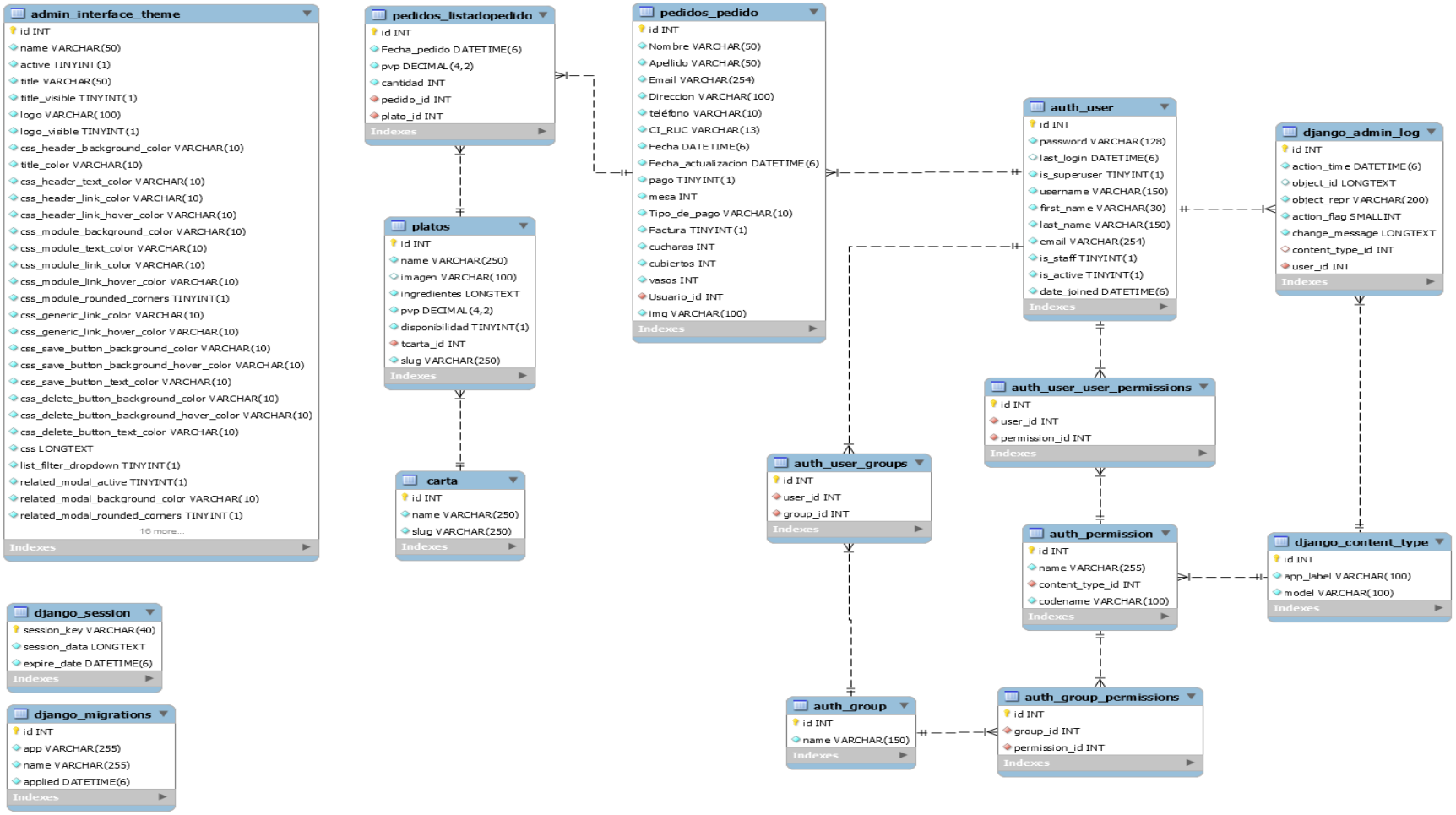


Figura 29. Diagrama entidad relación de la base de datos.

Fuente: Autor

2.5 PRUEBAS

Con el objetivo de validar el prototipo se realizaron pruebas técnicas y pruebas de uso del prototipo con usuarios a quienes se le informó sobre el procedimiento a seguir para la gestión de un pedido a través del manual. Posteriormente se evaluó la aceptación del prototipo con una encuesta SUS.

2.5.1 PRUEBAS TÉCNICAS

Las pruebas técnicas incluyen:

- La verificación del funcionamiento de la interfaz para la generación de códigos QR, mediante el ejecutable en diferentes computadoras.
- Pruebas del correcto funcionamiento al servidor local
- Pruebas de estrés al servidor web de producción.

- **Interfaz para la generación de Códigos QR**

Con la finalidad de evaluar la generación de códigos QR se puso a disposición de los usuarios administradores la creación de códigos QR para posteriormente poner a disposición de los usuarios (clientes) los códigos para el acceso a la plataforma digital. También se distribuyeron copias del ejecutable para probarlo en diferentes computadoras y validar las funcionalidades del sistema.

- **Sitio para administración**

En la interfaz para la administración de la plataforma es posible verificar los pedidos gestionados por los usuarios. Durante las pruebas se busca generar la factura del pedido para que los usuarios puedan comprobar si el pedido está correcto. Por otra parte, para la interfaz para administración se busca verificar la navegación en el sitio solicitando a los usuarios administradores utilizar con total libertad el recurso. Se puso a disposición los manuales que se encuentran en Anexos.

- **Tiempos de Respuesta**

Se analizaron los tiempos de respuesta del servidor web y el número máximo de solicitudes que puede atender, haciendo uso de Apache Bench.

2.5.2 PROTOCOLO DE PRUEBAS

Las pruebas de uso del prototipo con usuarios se realizaron con la colaboración de 15 participantes entre las edades de 12 años a 35 años de edad, a quienes se les manifestó el propósito de las pruebas y sus alcances.

Para las pruebas de usabilidad se estableció el siguiente protocolo:

1. Se puso a disposición de los usuarios el manual (Anexo 1) con un código QR diferente por cada mesa.
2. El usuario debe escanear el código e ingresar a la plataforma digital.
3. Se debe iniciar sesión en su cuenta o en caso contrario crear una .
4. Se debe adicionar productos al carrito de compras, seleccionar si desea su factura con datos y enviar el pedido.

Con la finalidad de que el usuario pueda interactuar con la plataforma se puso en funcionamiento el servidor de producción Apache24, sobre el cual se ejecuta el servicio de la plataforma digital. También se gestionó el rendimiento del computador al estar en funcionamiento mediante el administrador de tareas de Windows.

Se solicitó a los participantes interactuar con la plataforma para gestionar un pedido y tener presente todas las opciones que ofrece la plataforma. Luego de gestionar el pedido se generó una factura para corroborar lo solicitado.

Posteriormente a los usuarios se les solicitó diligenciar una encuesta del Sistema de Escalas de Usabilidad (SUS) con el objetivo de realizar mejoras y analizar el porcentaje de aceptación de la plataforma digital.

CAPÍTULO 3: RESULTADOS Y ANÁLISIS DE RESULTADOS

A continuación, se detalla los resultados de las pruebas realizadas al sistema.

3.1 INTERFAZ PARA GENERAR CODIGOS QR

En relación con la interfaz para crear los códigos QR, se distribuyeron copias del ejecutable a tres usuarios con permisos de administradores.

Los usuarios siguieron las indicaciones para la creación de los códigos QR, basados en el manual de usuario administrador (ver anexo 2). La interfaz cuenta con mensajes de alerta para que el usuario crear los códigos QR.

El ejecutable se podrá usar solamente en sistemas operativos Windows, debido a que la programación se realizó en un sistema operativo Windows. En caso de quererlo ejecutar en otro sistema operativo, se puede crear un nuevo ejecutable.

3.2 PLATAFORMA DIGITAL

Referente a la plataforma digital para la gestión de pedidos, se comprobó que mantiene un estilo *responsive*, ya que se acopla a los diferentes tamaños de pantalla de los dispositivos celulares, tablets y computadoras.

Se validó su funcionamiento en un computador y en un celular. Se verificó por tanto que la plataforma funciona correctamente en los navegadores más comunes como: Chrome, OPERA, Safari, Firefox y Microsoft Edge; e indistintamente en los sistemas operativos de los dispositivos celulares IOS y Android.

EVALUACION DE USABILIDAD

Los usuarios interactuaron con los códigos QR para acceder a la plataforma digital sin presentarse inconvenientes (ver Figura 31).

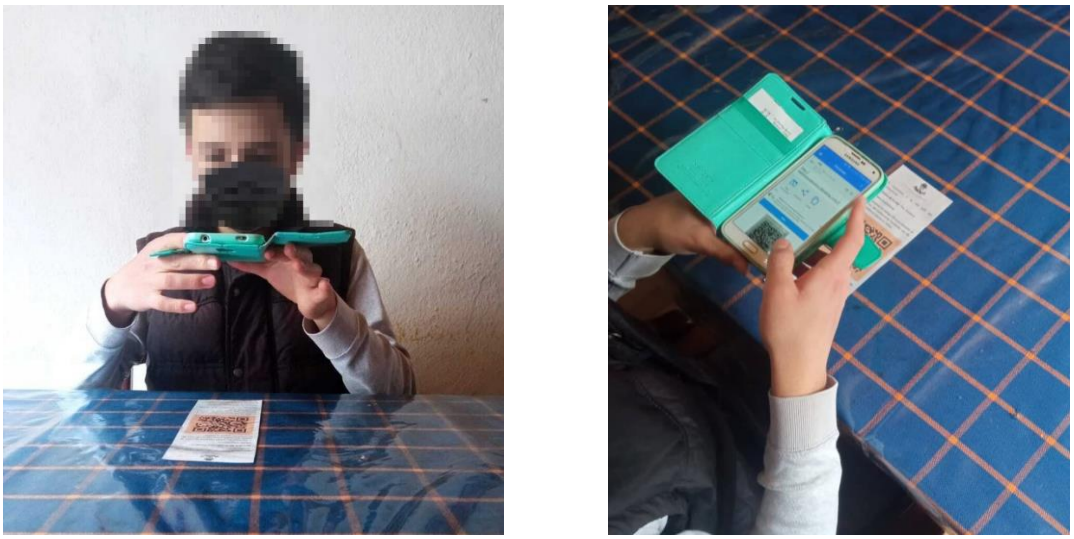


Figura 30. Escaneo de código QR por parte de un usuario para acceso a la plataforma digital

Seguido de esto, los usuarios pudieron ejecutar el protocolo para evaluar la usabilidad del sistema navegando en la plataforma digital para gestionar pedidos como se indica en la Figura 32.

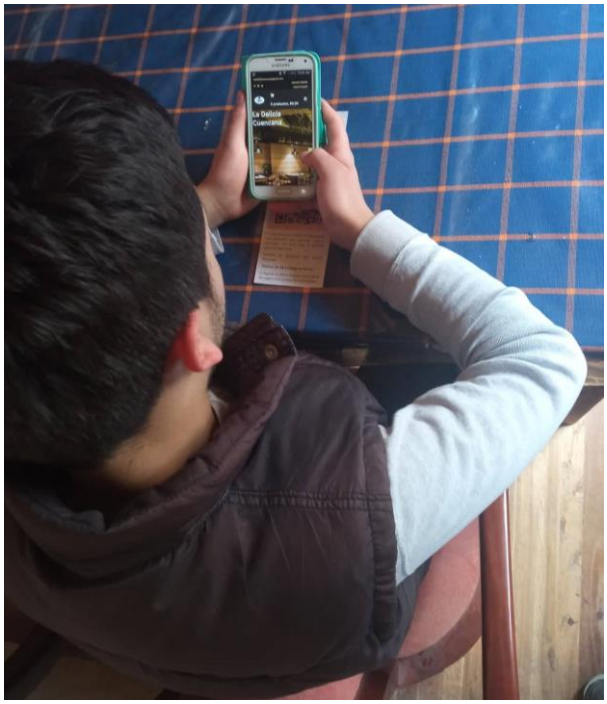


Figura 312. Navegación en la plataforma digital para gestionar un pedido

En la Tabla 9 y Tabla 10 se detallan los resultados de la encuesta SUS.

Tabla 9. Resultados de las preguntas 1, 2, 3, 4 y 5 de la encuesta SUS.

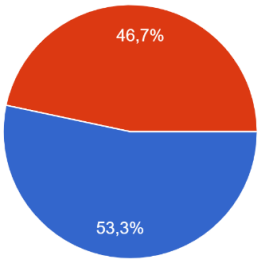
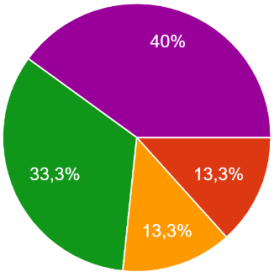
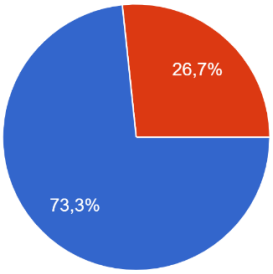
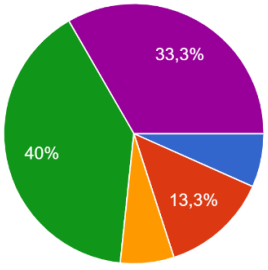
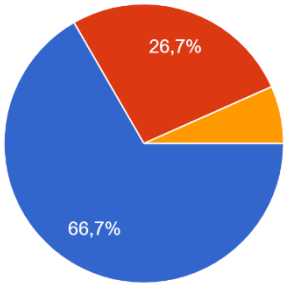
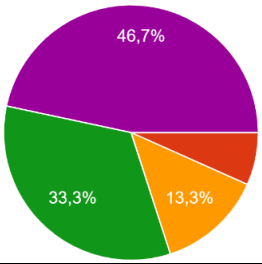
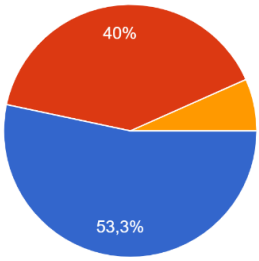
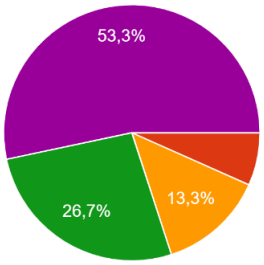
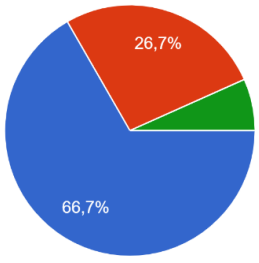
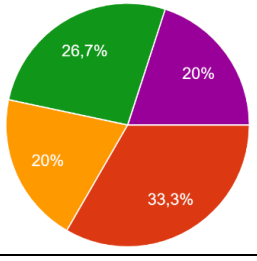
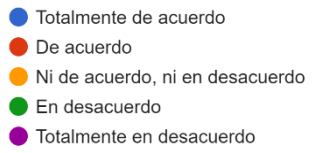
No	Consulta	Resultado
1	"Creo que me gustaría utilizar frecuentemente esta plataforma digital"	
2	"Encontré que la plataforma digital es más complicada de lo que debería ser"	
3	"Pienso que la plataforma digital es fácil de usar"	
4	"Creo que necesitaré el apoyo de personal técnico para poder utilizar esta plataforma digital"	
5	"Encontré que varias de las funciones en la plataforma digital funcionan bien y estaban bien integradas"	

Tabla 10. Resultados de las preguntas 6, 7, 8, 9, y 10 de la encuesta SUS.

No	Consulta	Resultado
6	"Pienso que hay demasiada inconsistencia o irregularidades en la plataforma digital"	
7	"Pienso que la mayoría de la gente puede aprender este sistema rápidamente"	
8	"Encontré que la plataforma digital es muy complicado de usar"	
9	"Me sentí muy seguro al utilizar esta plataforma digital"	
10	"Creo que hay muchas cosas que aprender antes de poder comenzar a usar esta plataforma digital"	

Las respuestas de la encuesta reciben la puntuación indicada en la Tabla 11.

Tabla 11. Puntuación de las preguntas de la encuesta SUS.

Respuestas	Puntaje	Leyenda
"Totalmente de acuerdo"	5	
"De acuerdo"	4	
"Ni de acuerdo, ni en desacuerdo"	3	
"En desacuerdo"	2	
"Totalmente en desacuerdo"	1	

Para saber el porcentaje de aceptación se clasifica por separado las preguntas de la encuesta por impares y pares. Al valor que se obtiene de cada pregunta impar se le debe restar el valor de 1 y para las preguntas pares se debe restar de 5 el valor obtenido; posteriormente se debe sumar los valores obtenidos y multiplicar por un valor de 2.5 como se muestra a continuación:

Respuestas de preguntas impares:

$$\big((8x5)+(7x4)\big)+\big((11x5)+(4x4)\big)+\big((10x5)+(4x4)+(1x3)\big)$$
$$+\big((8x5)+(6x4)+(1x3)\big)+\big((10x5)+(4x4)+(1x2)\big)$$

$$Promedio\,\frac{(343)}{15}=22.86$$

Respuestas de preguntas pares:

$$\big((2x4)+(2x3)+(5x2)+(6x1)\big)+\big((2x4)+(1x3)+(6x2)+(5x1)+(1x5)\big)$$
$$+\big((1x4)+(2x3)+(5x2)+(7x1)\big)+\big((1x4)+(2x3)+(4x2)+(8x1)\big)$$
$$+\big((5x4)+(3x3)+(4x2)+(3x1)\big)$$

$$Promedio\,\frac{(156)}{15}=10.4$$
$$25-10.4=14.6$$

Porcentaje de aceptación:

$$17.86+14.6=32.46$$
$$32.46x2.5=81.15\%$$

El resultado obtenido equivale al porcentaje de aceptación, siendo equivalente al 81.15%.

La encuesta SUS establece que los porcentajes de 80% o superiores se considera como sobresaliente, los que están sobre el 70% está sobre el promedio y los porcentajes inferiores al 50% es ineficiente.

Con base a este argumento y los resultados adquiridos, se puede deducir que la plataforma tiene un buen porcentaje de aceptabilidad.

Durante las pruebas no se presentaron problemas para la interpretación de las instrucciones. Cabe recalcar que los participantes contaron con acceso al manual de usuario que brinda las indicaciones para emplear el sistema.

Adicional a la encuesta SUS se solicitó a los usuarios brindar observaciones y recomendaciones adicionales para realizar mejorar al sistema. Con base a los comentarios recibidos se realizaron las siguientes modificaciones:

- Poner a disposición de los usuarios la opción de factura con y sin datos.
- Modificación del inicio de sesión, para que sea mediante el correo electrónico y no por el nombre de usuario.
- Incorporar mensajes de alerta para una mejor navegación en la plataforma digital.

3.3 PRUEBAS A LOS SERVIDORES

Antes deponer el proyecto en un servidor de producción se evalúo todo el desempeño de la plataforma ante posibles errores de programación mediante el servidor de desarrollo que incluye Django, obteniendo resultados exitosos.

Para alojar el proyecto en el servidor de producción Apache24 se realizó la instalación del módulo *mod_wsgi* que se compila con Microsoft Build Tools en Windows 10. El módulo *mod_wsgi* permite la comunicación del proyecto web y el servidor Apache24.

Con la finalidad de disponer de un DNS (Servicio de nombre de dominio), se usó el servicio de NO-IP, que es un proveedor de nombres de dominio que ofrece su servicio gratuito por el lapso de 30 días. La configuración del DNS en el router se lo puede observar en la Figura 33.

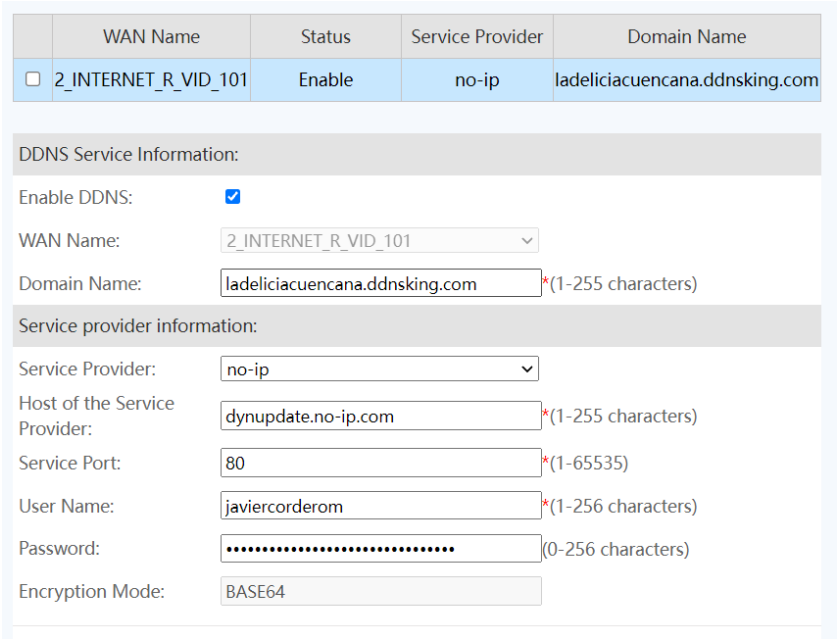


Figura 323. Configuración del servicio DNS en el router

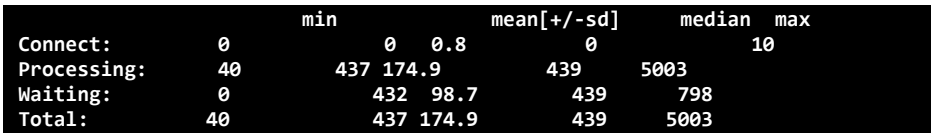
Las pruebas se realizaron con el servidor trabajando de forma local debido a las limitaciones del "ISP (proveedor de servicios de internet)". Por seguridad de la red el ISP dispone de dos cuentas, una de usuario (cliente) y otra de administrador.

Para acceder a todos los permisos de configuración del router (firewalls) es necesario la cuenta de administrador.

Para evaluar el volumen de solicitudes http que se pueden atender y la rapidez de respuesta del servidor de producción se utilizó la herramienta Apache Bench. Para la prueba se estableció una concurrencia de 65 usuarios con un total de 1000 solicitudes y en la Figura 34 se puede observar a detalle los resultados.

```
Server Software:        Apache
Server Hostname:        ladeliciacuencana.ddnsking.com
Server Port:            80
Document Path:          /1/
Document Length:        9433 bytes
Concurrency Level:      65
Time taken for tests:   11.722 seconds
Complete requests:      1000
Failed requests:        1
   (Connect: 0, Receive: 0, Length: 1, Exceptions: 0)
Keep-Alive requests:    999
Total transferred:      9773217 bytes
HTML transferred:       9423567 bytes
Requests per second:    85.31 [#/sec] (mean)
Time per request:       761.934 [ms] (mean)
Time per request:       11.722 [ms] (mean, across all concurrent requests)
Transfer rate:          814.20 [Kbytes/sec] received

Connection Times (ms)
```



```
                    min            mean[+/-sd]      median  max
Connect:          0              0   0.8          0              10
Processing:      40         437 174.9          439     5003
Waiting:          0            432  98.7           439      798
Total:           40            437 174.9          439     5003
```

Figura 334. Resultados de la evaluación del volumen de solicitudes con Apache Bench.

Se verificó que la cantidad promedio de solicitudes por segundo es de 85,31 y el consumo de tiempo medio por solicitud es de 11,722 [ms].

Se determinó que a partir de 1000 solicitudes el servidor Apache empieza a tener solicitudes fallidas.

El tiempo máximo de la conexión es de 10 [ms] y para el procesamiento de 5003 [ms].

3.4 ANÁLISIS DE COSTO DEL PROTOTIPO

Este análisis se realiza para establecer el costo del prototipo al consumidor final, para lo cual se considera los gastos en servicios, mano de obra y el tiempo.

En la tabla 12 se detallan los valores estimados del prototipo.

Tabla 12. Detalle de costos en el diseño del prototipo

Activos fijos	**Cantidad**	**Valor Unitario**	**Valor Final**
Computadora (Laptop Hp)	1	$1250,00	$1250,00
Regulador de voltaje	1	$15,00	$15,00
Escritorio	1	$100,00	$100,00
Monitor	1	$80,00	$80,00
Activos nominales			
Costos Fijos			
Agua potabilizada	1	$10,00	$10,00
Servicio Internet (40 Mbps)	1	$32,00	$32,00
Energía eléctrica	1	$15,00	$15,00
Arriendo	1	$120,00	$120,00
		Total	$177,00
Costos controlables			
Mano de obra	**Horas de programación**	**Valor Unitario**	**Valor Final**
Desarrollador web	70	$3,33	$233,33
Costo indirecto de producción (servicio de dominio y hosting**)**	**Cantidad** **1**	**Valor Unitario (mensual)** $2,45	**Valor Final** $29,40
		Total	$262,73

Costos no controlables:

Mediante los costos no controlables se puede obtener el valor respecto al arriendo y servicios básicos del desarrollo de una plataforma digital.

$$Costo_{Arriendo} = \frac{\left(\frac{Precio\_Arriendo}{\#\ Dias\ del\ mes}\right)}{\#\ horas\ del\ día}(\#horas\ de\ programación)$$

$$Costo_{Arriendo} = \frac{\left(\frac{120}{30}\right)}{24}(70) = \$11{,}67$$

$$Costo_{servicios\ basicos} = \frac{\left(\frac{Precio\_Servicios\_Basicos}{\#\ días\ del\ mes}\right)}{\#\ horas\ de\ lajornada\ de\ trabajo}(\#horas\ de\ programación)$$

$$Costo_{Arriendo} = \frac{\left(\frac{57}{30}\right)}{8}(70) = \$16{,}63$$

Costos controlables:

A continuación, se detalla el costo generado por la mano de obra para llevar a cabo el sistema.

$$Mano_{Obra} = \frac{(sueldo\ mensual)}{\#\ horas\ de\ labor\ mensual}(\#horas\ de\ programación)$$

$$Mano_{Obra} = \frac{800}{240}(70) = \$\ 233{,}33$$

Costo sin margen de utilidad:

El costo de la plataforma digital es referente a los costos no controlables, controlables y costo indirecto de producción.

$$CSMU = Costos\ no\ controlables + Costos\ controlables + Costo\ indirecto\ de\ producción$$

$$CSMU = 11.67 + 16.63 + 233.33 + 29.40$$

$$CSMU = \$\ 291{,}03$$

Costo con margen de utilidad

Para saber el precio de venta al público se consideró un margen de utilidad del 30% respecto al costo sin margen de utilidad.

$$CCMU = CSMU + Margen_{utilidad}\,(30\%)$$

$$CCMU = \quad 291.03 + \ 87.30$$

$$CCMU = \$\,378{,}34$$

Punto de equilibrio

El punto de equilibrio permite establecer el número de plataformas que se debería de diseñar mensualmente para obtener rentabilidad, para lo cual CF son los costos fijos, PvP es el precio de venta al público lo cual se aproximó al valor de $ 380,00 y CVu costo variable unitario correspondiente a la mano de obra y costo indirecto de producción.

$$QE = \frac{CF}{PvP - CVu}$$

$$QE = \frac{177}{380 - \ 262{,}73}$$

$$QE = 1.51$$

$$QE \approx 2$$

A partir de estos datos se puede concluir que se debería vender como base 2 plataformas digitales mensualmente.

CAPÍTULO 4: CONCLUSIONES Y RECOMENDACIONES

El prototipo de sistema para la gestión de decisiones a distancia (plataforma digital y códigos QR) desarrollado en el presente proyecto puede ser considerado como una herramienta útil para minimizar el contacto social y busca ser un aporte para reducir el número de contagios de la enfermedad de tipo viral COVID-19 en ambientes como los restaurantes. Debido a la concepción de su diseño, el sistema puede ser adaptado a otros ambientes en los cuales se requiera el mínimo contacto con superficies expuestas a estar contaminadas.

Etapas como la plataforma digital desarrollada en Django (considerado como un recurso escalable) tiene la ventaja de permitir presentar gran cantidad de contenido web, pudiendo adaptarse a diferentes entornos como son: educativo, comercial y empresarial.

La librería Qtdesigner, utilizada para el diseño de la interfaz de códigos QR, permite personalizar interfaces gráficas con un aspecto profesional, lo cual puede ser escalable a proyectos de otro interés. De ahí que el sistema de generación de códigos QR desarrollado puede emplearse para acceder a información y realizar trámites en línea sin necesidad de imprimir documentos o tickets para atención al cliente.

Es importante mencionar las ventajas de las tecnologías utilizadas como el Fronted, Bootstrap y CSS3 que permiten establecer un diseño profesional y adaptable a las necesidades de negocio. Esto se reflejó en las excelentes calificaciones que realizaron los usuarios que participaron en las pruebas y de quienes colaboraron para evaluar la interfaz de administrador.

Con base a los resultados de la prueba SUS, se puede deducir que el sistema es robusto en términos de su facilidad de uso, debido a que los participantes indicaron en altos porcentajes estar totalmente de acuerdo con esta afirmación. La retroalimentación de los usuarios también permitió realizar mejoras al prototipo. Después de realizar los ajustes son necesarias nuevas pruebas con personas de mayor rango de edad con el fin de evaluar la usabilidad del sistema.

Para hacer uso del sistema se recomienda poner a disposición de los usuarios los respectivos manuales de usuario cliente y administrador.

El servidor en producción se ejecutó de manera local debido a las limitantes del plan de contrato de servicio por parte del ISP que limita la liberación de puertos del router, ante posibles vulnerabilidades de información por parte de hackers. A fin de evitar estas restricciones se recomienda arrendar un servicio de hosting que en la mayoría de planes incluye un DNS gratuito y un certificado SSL. Es recomendable hacer uso de estos servicios ya que se evitaría estar al pendiente de la actividad del servidor ante posibles caídas por energía u otros factores las 24 horas del día.

Como trabajos futuros se propone la integración de sensores que permitan monitorear la temperatura en los locales y sensores de conteo para evitar aglomeraciones, poniendo a disposición esta información mediante un código QR.

También se propone la integración de un servicio de entregas a domicilio con pagos mediante tarjetas de crédito si su finalidad tiene un enfoque comercial y además de poder monitorear el servicio de entrega en tiempo real haciendo uso de Google Maps. También es posible adaptar el sistema para aplicaciones en el ambiente educativo, como: entregas de tareas, controles de asistencia, carga de documentos para fortalecer los conocimientos de los estudiantes, entre otros.

REFERENCIAS BIBLIOGRÁFICAS

[1] Johns Hopkins University & Medicine, "CORONAVIRUS RESOURCE CENTER," *Global Map*, 2021. https://coronavirus.jhu.edu/map.html (accessed Nov. 24, 2021).

[2] F. J. Días Castrillón and A. I. Toro Montoya, "Medicina y Laboratorio," *Med. Lab.*, vol. 24, 2020, doi: 10.36384/01232576.268.

[3] "Traqueotomía en pacientes COVID-19: un procedimiento necesario de alto riesgo. Experiencia de dos centros," no. January, 2020, [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7261441/.

[4] Centros para el Control y la Prevención de Enfermedades, "Cómo protegerse y proteger a los demás," *Centros para el Control y la Prevención de Enfermedades*, 2021. https://espanol.cdc.gov/coronavirus/2019-ncov/prevent-getting-sick/prevention.html (accessed Apr. 16, 2021).

[5] J. Craven, "Rastreador de vacunas COVID-19," 2021. https://www.raps.org/news-and-articles/news-articles/2020/3/covid-19-vaccine-tracker (accessed Apr. 16, 2021).

[6] C. Chino, I. Internacional, E. Parte, R. Sanitario, I. Rsi, and R. S. Internacional, "Actualización Epidemiológica Enfermedad por coronavirus," vol. 2019, no. 2005, pp. 6–7, 2021.

[7] P. Rodríguez Canfranc, "Prohibido tocar: tecnología para movernos por el mundo sin contacto físico," *15-06-2020*, 2020. https://telos.fundaciontelefonica.com/la-cofa/prohibido-tocar-tecnologia-para-movernos-sin-contacto-fisico-por-el-mundo/ (accessed Aug. 25, 2020).

[8] O. Islas, "14,576CIU, Empresas del Sector TIC Sumadas en el Combate Contra el COVID-19," 2020, [Online]. Available: https://octavioislas.com/2020/04/11/14576-ciu-empresas-del-sector-tic-sumadas-en-el-combate-contra-el-covid-19/.

[9] R. G. Acosta and A. T. Bernilla, "El nuevo Coronavirus y la pandemia del," pp. 125–131, 2020.

[10] A. Harrison, T. Lin, and P. Wang, "Mechanisms of SARS-CoV-2 Transmission and Pathogenesis," *PubMed*, 2020, doi: 10.1016/j.it.2020.10.004.

[11] M. R. Perez Abereu, J. J. Gomez Tejeda, and R. A. Dieguez Guach, "Características clínico-epidemiológicas de la COVID-19," *Rev. Habanera Ciencias Médicas*, vol. 19, [Online]. Available: http://www.revhabanera.sld.cu/index.php/rhab/article/view/3254.

[12] Ministerio de sanidad igualdad y asuntos sociales, "Información Científica-Técnica Coronavirus," *Cent. Coord. Alertas y Emergencias Sanit.*, vol. 1, p. 73, 2021.

[13] L. Tong, X. Gu, and F. Dai, "QR code detection based on local features," *ACM Int. Conf. Proceeding Ser.*, no. Version 7, pp. 319–322, 2014, doi: 10.1145/2632856.2632860.

[14] J. M. Huidobro, “Código QR,” *Bit 172*, pp. 47–49, 2009, [Online]. Available: https://cmapspublic2.ihmc.us/rid=1NS6XZ211-1V8WNZ2-2555/Microcodigos qr.pdf.

[15] J. Carlos, A. García, and S. Okazaki, “El uso de los códigos QR en España,” 2012.

[16] C. D. E. I. D. E. Sistemas, “Análisis, diseño e implementación de firmas electrónicas en documentos institucionales y la verificación mediante direccionamiento con códigos QR para la Universidad Politécnica Salesiana,” Universidad Politécnica Salesiana.

[17] D. A. Cedeño Ramos and L. E. Moncayo Solis, “Diseño e Implementación de una Plataforma de Control de Acceso Magnético/Biomñetrico y Supervisión Remota basado en Raspberry Pi,” Universidad Politécnica Salesiana, 2019.

[18] B. Neira Chavarría and E. Gudiño, “Implementación de un servidor web y un diseño de una página utilizando herramientas de software libre para el dispensario ¨SAGRADA FAMILIA¨ de la ciudad de Guayaquil,” Universidad Politécnica Salesiana, 2017.

[19] H. D. Baldassari Valencia, “‘ Estudio comparativo de motores de bases de datos SQL y NoSQL para la gestión de información transaccional ,’” Pontifica Universidad Católica del Ecuador, 2019.

[20] R. Marín, “Los gestores de bases de datos mas usados en la actualidad,” 2019, [Online]. Available: https://revistadigital.inesem.es/informatica-y-tics/los-gestores-de-bases-de-datos-mas-usados/.

[21] P. López Herrera, “Comparación del desempeño de los Sistemas Gestores de Bases de Datos MySQL y PostgreSQL,” Universidad Autónoma del Estado de México, 2016.

[22] S. Liawatimena, “Django Web Framework Software Metrics Measurement Using Radon and Pylint,” Jakarta Indonesia, 2018. doi: 10.1109/INAPR.2018.8627009.

[23] J. Rieken, “Visual Studio Code.” https://code.visualstudio.com/learn (accessed Apr. 15, 2021).

[24] A. M. Mejías Velló, “‘ Programa basado en Python para integrar la gestión de documentos y procesos de trabajo en una empresa ,’” Universidad Polit’ecnica de Valencia, 2019.

[25] A. Holovaty and J. Kaplan, *La guia definitiva de django*. Boston, 2018.

[26] H. Ayman, *Learning Website Development with Django*. Olton: April 2008, 2008.

[27] R. G. J. Antonio, *HTML5, css3 Y JQuery*, Ra-Ma 2016. España, 2016.

[28] N. Li, “The Design and Implementation of Responsive Web Page Based on HTML5 and CSS3,” *2019 Int. Conf. Mach. Learn. Big Data Bus. Intell.*, pp. 373–376, 2019, doi: 10.1109/MLBDBI48998.2019.00084.

[29] W. Jiang, M. Zhang, B. Zhou, Y. Jiang, and Y. Zhang, “Responsive Web Design Mode and Application,” pp. 1303–1306, 2014, doi: 10.1109/WARTIA.2014.6976522.

ANEXOS

ANEXO 1: MANUAL DE USUARIO

MANUAL DE USUARIO

SISTEMA PARA LA GESTIÓN DE DECISIONES A DISTANCIA

Objetivo: Dar a conocer al usuario la información sobre el modo de funcionamiento del sistema.

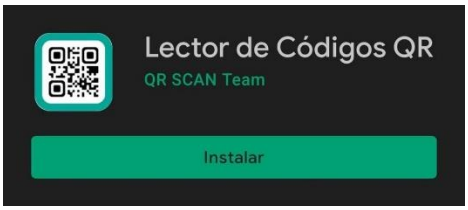
Para hacer uso del sistema se debe contar con lo siguiente:

Conexión a internet

Aplicación escáner de códigos QR en su teléfono móvil

En el caso de no poseer la aplicación podrá descargar una de Play Store o App Store, para los sistemas operativos Android y iOS respectivamente.

Se recomienda hacer uso de la aplicación “Lector de Codigos QR”.



Pasos

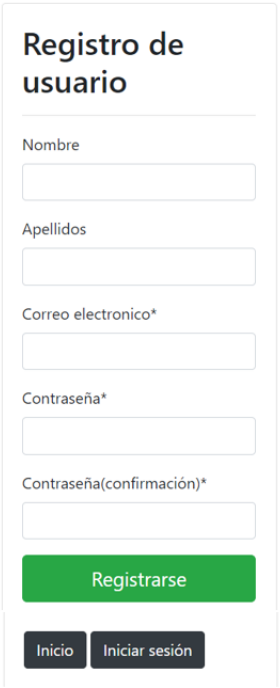
1. Ingresar al siguiente enlace para tener acceso a la plataforma digital:

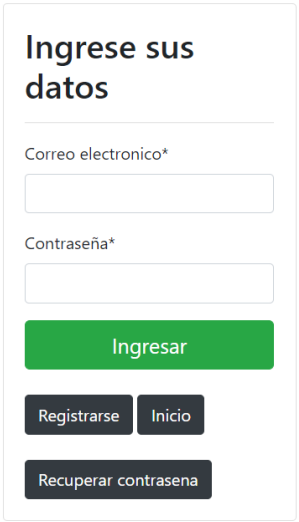
ladeliciacuencana.ddnsking.com/1

En el enlace de ingreso a la plataforma, el número del final puede variar debido a que cada mesa estará numerada y los códigos QR son diferentes.

2. Realizar el registro de Usuario

Por única vez el usuario deberá crear un registro (Login) en la plataforma. Después de contar con su perfil debe iniciar sesión ingresando sus datos para poder revisar el menú que se oferta.





3. Verificar el despliegue del menú

Una vez el menú se haya visualizado, el usuario podrá revisar la oferta del restaurante, con la opción de poder hacer una búsqueda en concreto por el tipo de carta

4. Añadir al carrito

El usuario por cada elemento elegido deberá seleccionar la opción ***añadir al carrito*.** Esto lo redirigirá a la interfaz para variar la cantidad del producto escogido o caso contrario eliminar el producto. Además, podrá regresar al menú y escoger más productos y revisar el costo total

5. Realizar el pedido

Para esta acción el usuario tendrá podrá solicitar las siguientes opciones:

- Facturas con datos o consumidor final
- Tipo de pago (Efectivo – Tarjeta de crédito)
- Vajilla adicional (cucharas, cubiertos, vasos)

ANEXO 2:

MANUAL DE ADMINISTRADOR PARA LA PLATAFORMA DIGITAL

MANUAL DE ADMINISTRADOR

SISTEMA PARA LA GESTIÓN DE DECISIONES A DISTANCIA _ PLATAFORMA DIGITAL

Objetivo: Dar a conocer al administrador la información sobre el modo de funcionamiento del sistema.

Para hacer uso del sistema se debe contar con lo siguiente:

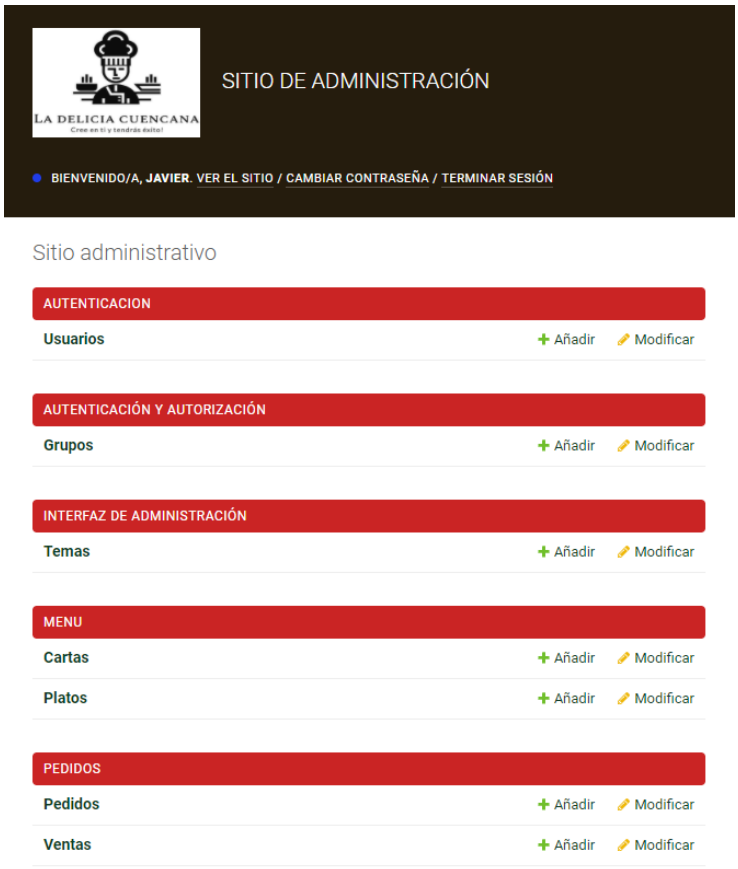
Conexión a internet

Pasos

1. El usuario administrador deberá dirigirse al enlace del sitio de administración e ingresar su nombre de usuario y contraseña.

ladeliciacuencana.ddnsking.com/admin

El sitio de administración dispone de las siguientes opciones:



- Autenticación y Autorización: Se pueden registrar a nuevos usuarios y administrar los permisos o roles que pueden realizar en el sitio de administración.
- Interfaz de Administración: Permite personalizar el aspecto visual de la interfaz gráfica.
- Menú: Dispone de la opción *Cartas* con la se puede añadir o eliminar los tipos de carta que oferta el restaurante y con la opción *Platos* para puede añadir platillos, seleccionando un tipo de carta a la que pertenece, además de poder editar el precio, ingredientes y disponibilidad
- Ventas: Existe la opción de *Listado de pedidos* para visualizar todas las ventas y exportar a un archivo de Excel. Está la opción *Pedidos* para visualizar todos los pedidos con los siguientes datos:

Nombre del cliente
Fecha de pedido
Fecha de actualización
Número de mesa
Tipo de pago
Estado del pago
Detalle del pedido
Factura

Es posible ingresar al número de identificación (id) del pedido y editar los campos.

En cualquier momento es posible adicionar o eliminar la selección realizada por el usuario.

ANEXO 3: MANUAL DE ADMINISTRADOR PARA LA INTERFAZ DE CODIGOS QR

MANUAL DE ADMINISTRADOR

SISTEMA PARA LA GESTIÓN DE DECISIONES A DISTANCIA _ INTERFAZ DE CÓDIGOS QR

Objetivo: Dar a conocer al administrador la información sobre el modo de funcionamiento del sistema para la generación de códigos QR.

Para hacer uso del sistema se debe contar con lo siguiente:

Conexión a internet

Pasos

1. Abrir el ejecutable de la interfaz con extensión .exe

La interfaz se visualiza en la siguiente figura.

El usuario podrá hacer uso del ejecutable en un sistema operativo Windows.

2. Ingresar en la barra superior la dirección URL con el número de mesa incluido, por ejemplo: ladeliciacuencana.ddnsking.com/1. El identificador de las mesas está establecido en la lógica de programación en un rango de 1 al 10.

3. Seleccionar el color de fondo y de la matriz de puntos para el código QR.

4. Para personalizar el código deberá seleccionar los colores a su elección.

5. Seleccionar la versión del código. Aquí, el usuario podrá seleccionar el número de la versión para el tamaño de la matriz en un valor comprendido de 1 al 5.

6. Generar el código QR con base a la información ingresada con la opción “Generar código QR”.

7. Guardar la imagen del código QR

8. Limpiar los campos para crear un código nuevo.