

# INTRODUCCIÓN

La factura digital tiene sus inicios en 1997 cuando la iniciativa privada empieza la búsqueda de un esquema legal que permita su empleo, para lo cual se crea un “Grupo de Facturación Electrónica” integrado por 45 grandes empresas que buscan con el fin común el empleo de comprobantes fiscales digitales (CFD). El grupo elaboró y evaluó un modelo e hizo pruebas piloto aplicables a la realidad a la que se buscaba llegar, esto permitió identificar qué modificaciones o adecuaciones debían hacerse a la ley para poder tener el marco jurídico que regulara la emisión de facturación electrónica. Hasta enero de 2004, todos los recibos que se emitían por los actos o actividades que realizaban los declarantes debían ser impresos por establecimientos autorizados por el Servicio de Administración Tributaria. Algunos contribuyentes que contaban con el recurso tecnológico y económico le solicitaron al SAT que se les permitiera emitir comprobantes fiscales digitales con la ayuda de un programa informático que generara recibos electrónicos en el momento deseado, beneficiándolos en cuanto a costos de emisión y mantenimiento. Rodríguez (2018). El objetivo en este proyecto es desarrollar una aplicación web que proporcione una estructura detallada para la organización, almacenamiento y gestión de facturas. Nilsen se dedica a dar seguimiento, acceso y gestión de documentos financieros.

El documento presenta un análisis detallado de los componentes esenciales de un sistema de control optimizado, diseñado para mejorar el funcionamiento de la aplicación en cuestión. Este se divide en cuatro partes principales. La primera sección establece las bases del proyecto, explorando la historia y los desafíos que enfrenta la institución involucrada. Se presenta la justificación del proyecto y los requerimientos técnicos necesarios para su implementación, siguiendo la metodología del modelo en V. En el segundo segmento se desvelan las herramientas tecnológicas que dan vida al software. Se destacan el editor de código de Microsoft (Visual Studio Code) y el lenguaje de programación PHP. También se utiliza el sistema web phpMyAdmin, con el cual se administran las bases de datos MySQL. En la tercera división se describen las características y funcionalidades de la aplicación web implementada, mostrando cómo se materializan las metas del proyecto. En el cuarto sector se incluye un análisis reflexivo sobre su desempeño en un entorno estándar. En la quinta fracción, se presenta las referencias bibliográficas que nutren el proyecto, mientras que el sexto bloque recopila anexos que brindan información adicional de interés.

El documento detalla los componentes esenciales de un sistema de control optimizado, abarcando desde sus bases teóricas hasta su implementación práctica. Se agradece la atención prestada y se espera que este análisis sea útil para futuros desarrollos en este campo.

# **CAPITULO I. MARCO METODOLOGICO**

## **1.1 Antecedentes y justificación**

La empresa Nilsen tiene sus inicios en el año 2005 con la visión de ofrecer soluciones tecnológicas y servicios de consultoría a diversas industrias. Desde sus primeros días ha crecido significativamente, expandiéndose tanto a nivel nacional como internacional. Sin embargo, este crecimiento ha presentado desafíos en la gestión de sus procesos internos, especialmente en la administración de facturas. Hoy en día, Nilsen gestiona sus facturas mediante métodos manuales, utilizando registros en papel y hojas de cálculo de Excel. La acumulación de documentos físicos dificulta el acceso rápido a información financiera crítica, y la introducción manual de datos es propensa a errores, lo que afecta la precisión de los registros.

Esta aplicación web para Nilsen centraliza la organización y gestión de facturas mediante una base de datos estructurada que permite subir, almacenar y consultar documentos digitalizados, registrando detalles como la empresa emisora, el total de la compra y los productos adquiridos. Los empleados pueden acceder a esta información desde cualquier dispositivo con conexión a internet, a través de la interfaz de la aplicación web pueden hacer búsquedas por fecha, empresa o producto, además de permitir la generación de reportes en formato Excel para la revisión y análisis de las transacciones. Esta automatización del proceso reduce las tareas repetitivas, permite organizar la información para la toma de decisiones.

## **1.2 Objetivos**

### **1.2.1 Objetivo general**

Analizar, diseñar y desarrollar una aplicación web para la organización, almacenamiento y consulta de facturas para la empresa Nilsen

### 1.2.2 Objetivos específicos

- Entrevistar al cliente para comprender sus necesidades y funcionalidades esenciales.
- Analizar los requisitos para garantizar la seguridad y eficiencia de la aplicación.
- Diseñar el modelo de la base de datos según los datos requeridos y definir las tablas necesarias.
- Crear la maquetación de la aplicación web para asegurar su intuición y adaptabilidad en distintos dispositivos.
- Desarrollar el frontend y el backend con HTML5, CSS3, PHP y JavaScript.
- Implementar la base de datos en MySQL.
- Realizar pruebas en producción para verificar la estabilidad y el rendimiento bajo carga.
- Recopilar comentarios y retroalimentación de los usuarios finales para efectuar mejoras.
- Documentar el proceso.

### 1.3 Alcance

La aplicación web contendrá los siguientes módulos:

#### **Usuario:**

- Menú de inicio de bienvenida
- Descripción de características principales.
- Demostración breve o vídeo introductorio.
- Formulario de registro
- Información de contacto o soporte

## **Administrador**

Menú de Inicio:

- Acceso rápido a funcionalidades principales.
- Subida de archivos para escaneo y procesamiento de facturas.
- Acceso a archivos recientes y estadísticas de facturas.

Módulo de categorías:

- Visualización de categorías como Compras de Productos, Servicios Profesionales, Gastos Operativos.
- Opción para añadir nuevas categorías con nombre, descripción, código e icono correspondiente.

Módulo de facturas:

- Muestra de facturas escaneadas y procesadas, organizadas por categorías.
- Visualización detallada de cada factura.

Módulo de historial:

- Registro de todas las facturas procesadas con detalles como tienda, nombre, fecha y hora.
- Organización de facturas por categorías y acceso a detalles específicos de cada una.

## **1.4 Metodología**

La metodología por seguir es la de modelo en V, se caracteriza por su enfoque secuencial y lineal, dividiendo el proceso en fases claramente definidas que se representan gráficamente en forma de "V". Esta forma se debe a la correspondencia entre cada fase de desarrollo y su fase correspondiente de pruebas, las cuales se encuentran alineadas y apuntan hacia la validación del producto final. Las fases del modelo en V son los siguientes:

1. **Análisis:** Se define el alcance del proyecto, las necesidades del usuario y los requisitos del sistema. Se crea un Documento de Requisitos del Sistema (DRS).
2. **Diseño:** Se realiza el diseño del sistema, incluyendo arquitectura, interfaces, bases de datos y módulos. Se crea un Documento de Diseño del Sistema (DDS).
3. **Implementación:** Se escribe el código fuente del software de acuerdo con los diseños.
4. **Pruebas unitarias:** Se realizan pruebas para verificar que cada módulo funcione correctamente.

5. Integración: Se integran los módulos y se realizan pruebas para verificar que el sistema funciona como un todo.
6. Pruebas de sistema: Se realizan pruebas para verificar que el software cumple con todos los requisitos del DRS.
7. Aceptación: El usuario prueba el software para verificar que satisface sus necesidades. Si la prueba es exitosa, el software se entrega.

La información de este modelo fue consultada del libro llamado Modelo en V para el desarrollo de software: Autor: Khan, S. S. (2014), y otro llamado Un análisis comparativo del Modelo en V y el Modelo en Cascada: Autores: Yoo, S., & Myung, J. W. (2007).

## **CAPITULO II. REVISIÓN BIBLIOGRÁFICA**

### **2.1 Tecnologías Web**

Las tecnologías móviles son un conjunto de herramientas y soluciones que nos permiten estar conectados a un mundo de información, comunicación y entretenimiento desde dispositivos portátiles como teléfonos inteligentes, tabletas y relojes inteligentes. Estas tecnologías se componen de redes móviles que posibilitan la comunicación inalámbrica, dispositivos móviles que nos dan acceso a estas redes, plataformas y sistemas operativos que permiten ejecutar aplicaciones, y el desarrollo de aplicaciones móviles que amplía aún más las posibilidades de uso de estos dispositivos. En conjunto, las tecnologías móviles han revolucionado la forma en que vivimos, nos comunicamos, impactando en diversos aspectos de la sociedad como la educación, el comercio y la forma en que nos relacionamos con el mundo que nos rodea. Singh (2019).

### **2.2 Aplicación web**

Las aplicaciones web, también conocidas como "web apps" o "software web", son programas que se ejecutan en un navegador web, como Google Chrome, Mozilla Firefox o Safari. A diferencia de las aplicaciones tradicionales que requieren instalación en el dispositivo, las aplicaciones web no necesitan instalación y se pueden utilizar desde cualquier dispositivo con acceso a internet.

En lugar de tener que realizar manualmente ciertas tareas, una aplicación web proporciona una interfaz gráfica y herramientas que permiten llevar a cabo diversas actividades de manera rápida y sencilla.

Cuando se accede a una aplicación web, el navegador envía una solicitud a un servidor web donde está almacenada la aplicación. El servidor web procesa esta solicitud y envía la información necesaria para que el navegador muestre la interfaz de la aplicación y ejecute sus funciones. La interacción entre el usuario y la aplicación se realiza a través del navegador, con datos enviados y recibidos del servidor web.

La variedad de aplicaciones web abarca diversas necesidades e intereses de los usuarios, incluyendo:

- Aplicaciones de productividad: Correo electrónico, calendarios, agendas, gestores de tareas, entre otros.

- Aplicaciones de redes sociales: Facebook, Twitter, Instagram, entre otros.
- Aplicaciones de comercio electrónico: Amazon, eBay, AliExpress, entre otros.
- Aplicaciones de entretenimiento: Netflix, Spotify, YouTube, entre otros.
- Aplicaciones educativas: Khan Academy, Coursera, edX, entre otros.
- Aplicaciones de juegos: Candy Crush, Angry Birds, Clash Royale, entre otros.
- Aplicaciones de herramientas de trabajo: Google Docs, Microsoft Office 365, Trello, Slack.

Las aplicaciones web ofrecen varias ventajas sobre las aplicaciones tradicionales:

- **Accesibilidad:** Se pueden utilizar desde cualquier dispositivo con acceso a internet, eliminando la necesidad de instalar software específico.
- **Actualizaciones automáticas:** Las actualizaciones se realizan automáticamente en el servidor web, sin necesidad de intervención del usuario.
- **Compatibilidad:** Son compatibles con diferentes navegadores web y sistemas operativos.
- **Escalabilidad:** Se adaptan a un gran número de usuarios sin requerir cambios en los dispositivos de los usuarios.
- **Reducción de costos:** No es necesario adquirir licencias de software ni realizar instalaciones en cada dispositivo.

## 2.3 Usabilidad

En el panorama digital actual, donde la interacción con productos y servicios se ha vuelto omnipresente, la usabilidad emerge como un concepto fundamental para garantizar una experiencia satisfactoria para los usuarios. Se trata de un atributo crucial que determina la facilidad con la que las personas pueden utilizar un producto o servicio para alcanzar sus objetivos de manera eficaz, eficiente y agradable.

Más allá de la simple funcionalidad:

La usabilidad va más allá de la mera funcionalidad. Un producto o servicio puede cumplir su cometido principal, pero si su uso resulta complejo, frustrante o poco intuitivo, es probable que los usuarios lo abandonen en busca de alternativas más amigables. Es por ello por lo que la usabilidad se enfoca

en optimizar la experiencia del usuario, considerando aspectos como la claridad, la consistencia, la eficiencia y la satisfacción.

Un enfoque centrado en el usuario:

El núcleo de la usabilidad reside en la adopción de un enfoque centrado en el usuario. Esto implica comprender las necesidades, expectativas y limitaciones de los usuarios que interactúan con el producto o servicio. A través de investigaciones, pruebas y análisis, se identifican los puntos fuertes y débiles de la experiencia, permitiendo así realizar mejoras continuas que optimicen la interacción. Velázquez (2015).

## **2.4 Base de Datos**

Una base de datos es un almacén organizado de información, similar a una biblioteca gigante donde se archivan libros. En lugar de libros, guarda datos estructurados, como números, nombres, fechas o imágenes.

Imagina una tabla con filas y columnas, cada fila representa un registro individual y cada columna un tipo de dato. Esta organización permite encontrar información rápidamente usando un sistema de gestión de bases de datos (SGBD), como buscar un libro específico en la biblioteca.

Las bases de datos son esenciales para empresas, gobiernos y organizaciones que manejan grandes cantidades de información. Permiten almacenar, organizar, consultar y analizar datos de manera eficiente, facilitando la toma de decisiones y el conocimiento del negocio o entorno. Norman (2013)

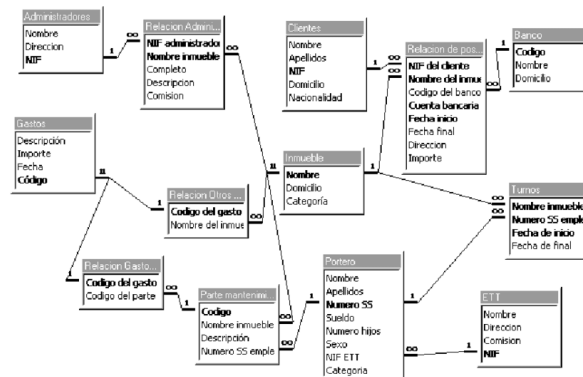
## **2.5 Modelos de bases de datos**

### **2.5.1 Modelo jerárquico**

El modelo jerárquico de bases de datos organiza los datos en una estructura similar a un árbol, donde cada registro, también llamado nodo, tiene un único padre y puede tener múltiples hijos, estableciendo una relación uno a muchos. En este modelo, los datos se representan de manera jerárquica, comenzando con un único nodo raíz y extendiéndose hacia abajo en varios niveles de nodos. Este modelo es eficiente para representar datos con relaciones jerárquicas claras y bien



definidas, como estructuras de organización, catálogos de productos o sistemas de archivos. Morrison (2015)



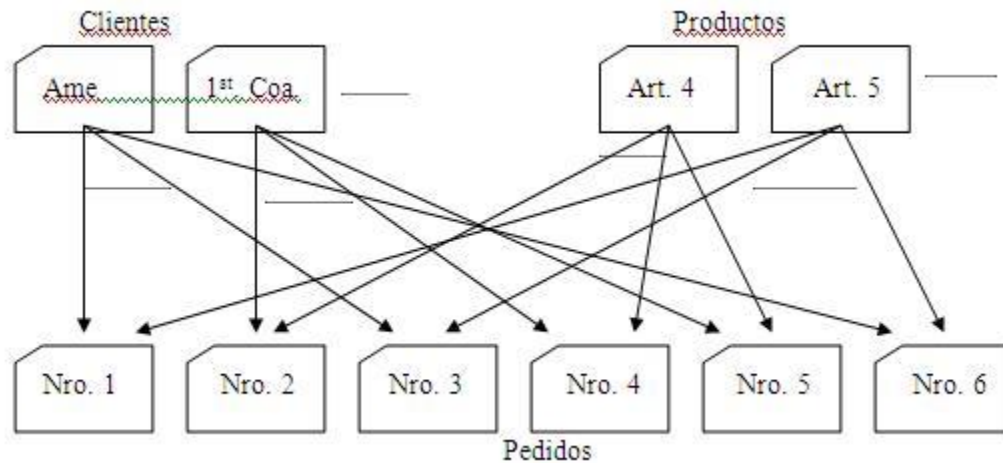
**Figura 1. Modelo jerárquico**

## 2.5.2 Modelo de red

El modelo de red de bases de datos organiza los datos utilizando una estructura de grafo en la que cada registro, o nodo, puede tener múltiples padres y múltiples hijos. Esto permite establecer relaciones muchos a muchos entre los datos, lo que proporciona una mayor flexibilidad comparada con el modelo jerárquico.

En este modelo, los datos se representan mediante registros conectados entre sí a través de enlaces, también llamados aristas, que indican las relaciones entre los registros. Estos enlaces permiten navegar por la base de datos siguiendo diferentes caminos, lo cual facilita la representación de estructuras de datos complejas con múltiples interconexiones.

El modelo de red es especialmente útil en aplicaciones que requieren una representación rica y compleja de relaciones, como los sistemas de gestión de inventarios, las redes de transporte, y las redes de telecomunicaciones. Sin embargo, su implementación y gestión pueden ser más complicadas que otros modelos, debido a la necesidad de mantener múltiples enlaces y la complejidad de las operaciones de actualización y eliminación en la red de datos. Korth (2002).



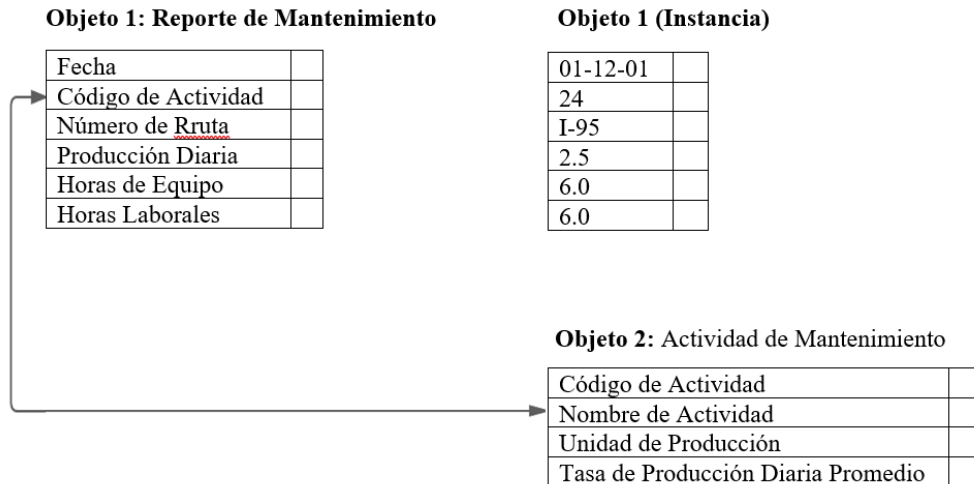
**Figura 2. Modelo de red**

### 2.5.4 Modelo orientado a objetos

El modelo orientado a objetos en bases de datos combina principios de la programación orientada a objetos con las capacidades de gestión de datos de las bases de datos. En este modelo, los datos se representan como objetos, similar a como se estructuran en los lenguajes de programación orientada a objetos.

Cada objeto en la base de datos contiene datos, en forma de atributos, y comportamientos, en forma de métodos. Además, los objetos pueden heredar propiedades y métodos de otros objetos, permitiendo una estructura jerárquica y reutilización de código.

Este modelo es particularmente útil para aplicaciones que manejan datos complejos y tienen una estrecha integración con lenguajes de programación orientada a objetos, como sistemas de diseño asistido por computadora (CAD), sistemas de información geográfica (GIS), y aplicaciones multimedia. La principal ventaja del modelo orientado a objetos es su capacidad para manejar datos complejos de manera más intuitiva y directa que los modelos relacionales tradicionales. García (2020).



**Figura 3. Modelo orientado a objetos**

### 2.5.5 Modelo multidimensional

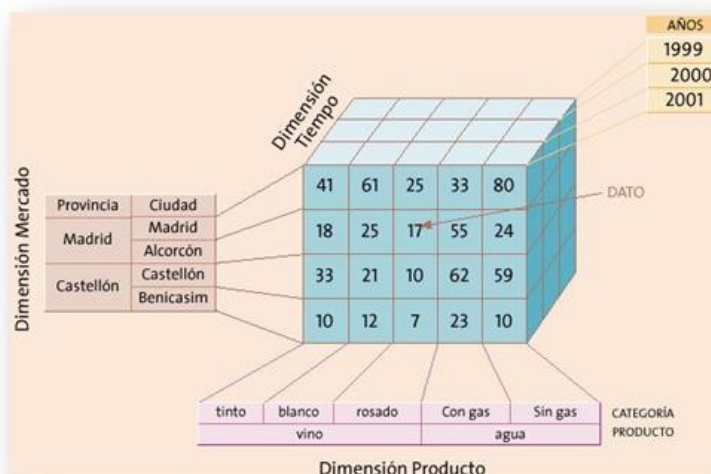
El modelo multidimensional en bases de datos se utiliza principalmente en sistemas de soporte a la toma de decisiones y análisis de datos, como en aplicaciones de inteligencia de negocios y almacenes de datos. Este modelo organiza los datos en un espacio multidimensional, donde cada dimensión representa una perspectiva o atributo diferente de los datos.

En este modelo, los datos se representan en forma de cubos, donde cada celda del cubo contiene datos agregados, como sumas o promedios, que se calculan a lo largo de varias dimensiones. Las dimensiones pueden incluir aspectos como tiempo, ubicación geográfica, productos, clientes, etc.

Las principales características del modelo multidimensional incluyen:

- Dimensiones: Ejes o perspectivas de análisis. Por ejemplo, tiempo (años, meses, días), ubicación (país, región, ciudad), y productos (categoría, marca).
- Medidas: Datos cuantitativos que se analizan, como ventas, ingresos, cantidades.
- Cubos: Conjuntos de datos organizados por múltiples dimensiones, permitiendo la visualización y análisis desde diferentes ángulos.

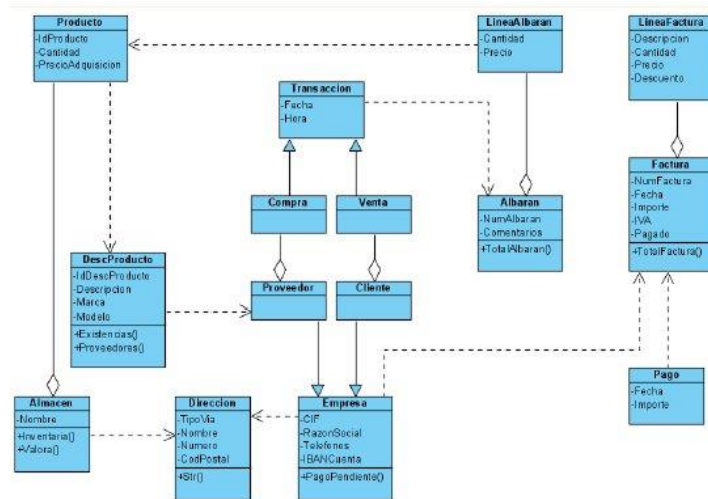
El modelo multidimensional facilita el análisis rápido y flexible de grandes volúmenes de datos, permitiendo a los usuarios realizar consultas complejas y obtener información significativa de manera eficiente. Ponniah. (2022).



**Figura 4. Modelo multidimensional**

## 2.5.6 Modelo Objeto-relacional

El modelo objeto-relacional (O-R) en bases de datos es una extensión del modelo relacional tradicional que incorpora conceptos de la programación orientada a objetos (POO) para el modelado y gestión de datos. Combina la flexibilidad y potencia del modelo relacional con la capacidad de representar objetos complejos y sus relaciones en un entorno de bases de datos. Solís (2017).



**Figura 5. Modelo Objeto-relacional**

## 2.6 Lenguajes de Programación Orientado a Objetos (Terminología)

### 2.6.1 Clases

En programación orientada a objetos (POO), una clase es una estructura fundamental que actúa como una plantilla para crear objetos. Define un conjunto de atributos y métodos que representan las características y comportamientos comunes a todos los objetos que pertenecen a esa clase. La clase proporciona una abstracción del objeto, ocultando los detalles de implementación y exponiendo solo lo necesario para interactuar con él. Los atributos (también conocidos como propiedades o campos) son variables que almacenan el estado de un objeto. Cada objeto de la clase tendrá su propia copia de estos atributos, con valores que pueden ser únicos para ese objeto. Por ejemplo, en una clase *Persona*, los atributos podrían incluir nombre, edad y dirección. Los métodos son funciones o procedimientos definidos dentro de la clase que operan sobre los atributos del objeto y definen su comportamiento. Los métodos permiten que los objetos realicen acciones, manipulen sus datos y respondan a mensajes (llamadas de métodos) de otras partes del programa. Por ejemplo, en la clase *Persona*, podría haber métodos como *hablar()*, *caminar()* y *dormir()*. Una clase puede incluir constructores, que son métodos especiales utilizados para inicializar los objetos de la clase. Los constructores suelen tener el mismo nombre que la clase y se llaman automáticamente cuando se crea un nuevo objeto. Su propósito es establecer los valores iniciales de los atributos del objeto y realizar cualquier configuración necesaria. Además, las clases en POO soportan el encapsulamiento, que es el principio de ocultar los detalles internos de los objetos y exponer solo una interfaz pública.

Esto se logra mediante modificadores de acceso como `public`, `private` y `protected`, que controlan la visibilidad de los atributos y métodos. El encapsulamiento mejora el modularidad y la protección del código, ya que los datos sensibles o internos no son accesibles directamente desde fuera de la clase. Las clases también pueden participar en relaciones de herencia, donde una clase (subclase) hereda atributos y métodos de otra clase (superclase). Esto permite la reutilización de código y la creación de jerarquías de clases que modelan relaciones más complejas entre los objetos. En resumen, una clase en programación orientada a objetos es una plantilla que define el estado y comportamiento de un conjunto de objetos, proporcionando una estructura organizada y modular para el desarrollo de software. Rodríguez (2021).

### **2.6.2 Objetos**

En programación orientada a objetos (POO), un objeto es una instancia concreta de una clase, que representa una entidad individual con un estado específico y un comportamiento definido. Los objetos son las unidades básicas de las aplicaciones orientadas a objetos y se crean a partir de la plantilla proporcionada por una clase. Cada objeto tiene su propio estado, que está determinado por los valores actuales de sus atributos. Los atributos son variables definidas en la clase, y cada objeto tiene su propia copia de estos atributos con valores que pueden ser diferentes de los valores de otros objetos de la misma clase. Por ejemplo, en una clase `Persona`, los atributos podrían incluir nombre, edad y dirección, y cada objeto `Persona` tendría su propio nombre, edad y dirección. El comportamiento de un objeto está definido por los métodos de su clase. Los métodos son funciones o procedimientos que operan sobre los atributos del objeto y definen las acciones que el objeto puede realizar. Cuando un método se llama en un objeto, puede manipular el estado del objeto y llevar a cabo operaciones específicas. Por ejemplo, en la clase `Persona`, podría haber métodos como `hablar()`, `caminar()` y `dormir()`, que describen las acciones que una persona puede realizar. Además, cada objeto tiene una identidad única, lo que significa que incluso si dos objetos tienen el mismo estado (los mismos valores de atributos), siguen siendo entidades distintas con ubicaciones de memoria separadas. En resumen, un objeto en programación orientada a objetos es una instancia específica de una clase que encapsula un estado y un comportamiento definidos por sus atributos y métodos, respectivamente, y tiene una identidad única que lo distingue de otros objetos. G Booch, R. (2007).

### **2.6.3 Abstracción**

La abstracción en programación orientada a objetos (POO) se logra a través de la creación de clases, que actúan como plantillas para crear objetos. Una clase define un conjunto de atributos y métodos

que representan los aspectos esenciales de un objeto, mientras que oculta los detalles de implementación internos. Los atributos representan el estado del objeto, es decir, sus características o propiedades, mientras que los métodos representan las operaciones o acciones que el objeto puede realizar.

Por ejemplo, considera una clase Vehículo que tiene atributos como `marca`, `modelo`, `año`, `color`, etc., y métodos como `acelerar ()`, `frenar ()`, `girar ()`, etc. Estos métodos permiten al objeto Vehículo realizar las acciones relevantes para un vehículo, pero no se preocupan por los detalles internos de cómo se implementan estas acciones. Esto facilita la reutilización de código, ya que los objetos pueden ser creados y usados fácilmente, y se centra en la abstracción de una entidad Seidewitz. (2000).

## 2.6.4 Encapsulamiento

El encapsulamiento en programación orientada a objetos (POO) implica el agrupamiento de los datos (atributos) y métodos (funciones o procedimientos) que operan sobre esos datos en una sola unidad, es decir, en la clase. Los datos están ocultos del mundo exterior y solo se pueden acceder a ellos mediante los métodos de la clase. Estos métodos proporcionan un mecanismo de interfaz pública a través del cual otros objetos pueden interactuar con el objeto.

El encapsulamiento ayuda a garantizar que los objetos mantengan su estado interno coherente y consistente al restringir el acceso directo a sus datos privados. Los atributos de la clase se definen con modificadores de acceso (`private`, `protected`, `public`) para controlar quién puede modificar o leer estos datos. Los métodos públicos permiten que los objetos clientes realicen operaciones permitidas en el objeto, mientras que los detalles internos y la implementación específica del objeto se mantienen ocultos. Larman (2004)

## 2.6.5 Herencia

En programación orientada a objetos (POO), herencia es un principio mediante el cual una clase puede heredar atributos y métodos de otra clase, conocida como superclase o clase base. Este mecanismo permite que las clases hijas, llamadas subclases o clases derivadas, extiendan y especialicen el comportamiento de la clase base. La herencia facilita la reutilización de código al

permitir que las subclases aprovechen la estructura y funcionalidad definida en la superclase, evitando la duplicación de código y mejorando la mantenibilidad del sistema.

Por ejemplo, una clase Vehículo puede ser una superclase que define atributos y métodos comunes a todos los vehículos, como marca, modelo, color, acelerar (), frenar (), etc. Las subclases como Automóvil y Motocicleta pueden heredar estos atributos y métodos y añadir características específicas como numero Puertas y cilindrada, respectivamente. Esto permite modelar jerarquías de clases donde las subclases heredan y extienden el comportamiento y la estructura de la superclase, proporcionando una forma eficiente de organizar y gestionar el código en aplicaciones complejas. Roque (2019).

### **2.6.6 Polimorfismo**

En programación orientada a objetos (POO), polimorfismo se refiere a la capacidad de objetos de diferentes clases de responder al mismo mensaje (llamada a método) de manera distinta. Esto significa que un mismo método puede comportarse de manera diferente según el tipo del objeto que lo llama, permitiendo un comportamiento más flexible y adaptable en el diseño de software.

El polimorfismo se implementa principalmente a través de dos técnicas: sobrecarga de métodos y sobreescritura de métodos. La sobrecarga de métodos permite definir varios métodos con el mismo nombre, pero con diferentes listas de parámetros en la misma clase, mientras que la sobreescritura de métodos permite que una subclase proporcione una implementación específica de un método que ya está definido en su superclase. Esto permite adaptar el comportamiento de un método a las necesidades específicas de cada clase derivada, proporcionando una manera poderosa de reutilizar y extender el código en grandes sistemas de software orientados a objetos. García (2018).

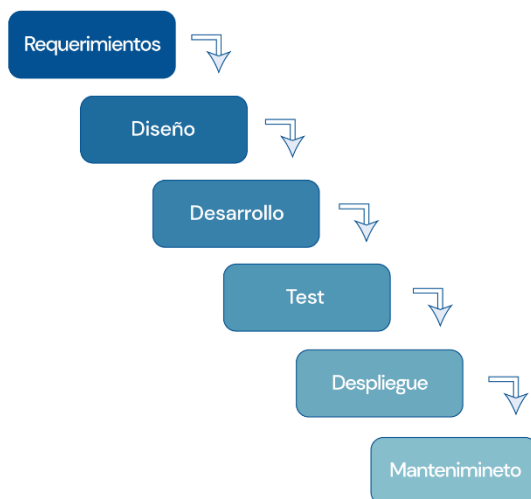
## **2.7 Modelos de desarrollo**

### **2.7.1 Metodología de Cascada**

El modelo en cascada, también conocido como modelo lineal o ciclo de vida en cascada, es una metodología de gestión de proyectos secuencial para el desarrollo de software. Se caracteriza por dividir el proceso de desarrollo en fases rígidas y predefinidas, cada una con objetivos y entregables



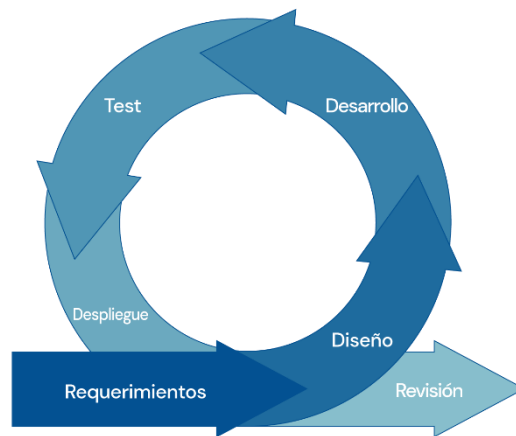
específicos. Cada fase debe completarse antes de iniciar la siguiente, lo que genera un flujo de trabajo lineal y unidireccional. Royce (1970)



**Figura 6. Cascada**

### **2.7.2 Metodología Agile**

Las metodologías ágiles son un conjunto de enfoques para el desarrollo de software que se basan en los valores y principios expresados en el Manifiesto para el Desarrollo de Software Ágil. En contraste con el modelo en cascada, las metodologías ágiles promueven un enfoque flexible, adaptativo e iterativo, donde el desarrollo se divide en ciclos cortos de trabajo con entregas incrementales y retroalimentación continua. Stevens. (2014).



**Figura 7. Metodologías ágiles**

### 2.7.3 Modelo en V

El Modelo V, también conocido como Modelo de Verificación y Validación o Ciclo en V, es una metodología de desarrollo de software que combina enfoques secuenciales y de cascada con actividades de prueba y validación en cada etapa del ciclo de desarrollo. Su nombre se debe a la forma en "V" que representa el flujo de trabajo, donde las actividades de desarrollo en la primera mitad del ciclo se corresponden con las actividades de prueba y validación en la segunda mitad. Haugen. (2019).



**Figura 8. Modelo en V**

## 2.7.4 Metodología en Espiral

El Modelo en Espiral es una metodología de desarrollo de software iterativa y cíclica que combina elementos de modelos secuenciales y de cascada con características de las metodologías ágiles. Fue propuesto por Barry Boehm en 1986 como una alternativa a los modelos tradicionales de desarrollo de software que no se adaptaban bien a proyectos complejos y cambiantes. Boehm (1988).

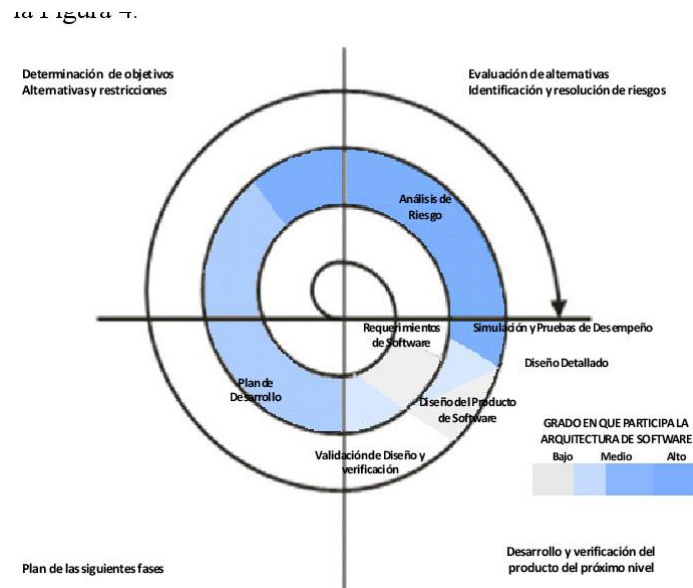
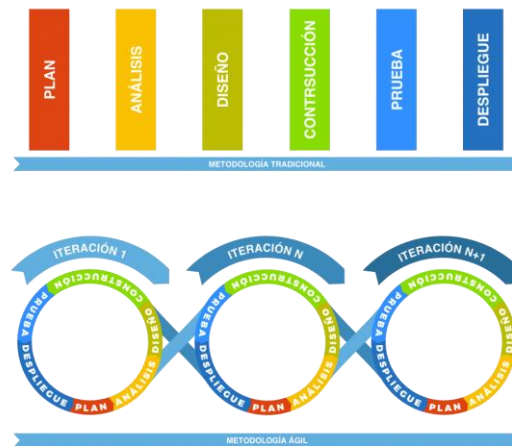


Figura 9. Espiral

## 2.7.5 Scrum

Scrum es una metodología ágil para el desarrollo de software que se basa en un enfoque iterativo e incremental. Se caracteriza por dividir el trabajo en ciclos cortos llamados sprints, que suelen durar de 1 a 4 semanas, con objetivos específicos y entregables funcionales al final de cada ciclo Beublé (2017).



**Figura 10. Scrum**

### 2.7.6 Programación Extrema (XP)

La Programación Extrema (XP) es una metodología ágil para el desarrollo de software que se basa en valores, principios y prácticas para producir software de alta calidad de manera rápida y adaptable. Se caracteriza por un enfoque iterativo e incremental, con énfasis en la comunicación, la colaboración y las pruebas continuas.



**Figura 11. XP**

## CAPÍTULO III RESULTADOS OBTENIDOS

### 3.1 Análisis

#### 3.1.1 Descripción del sistema actual

El problema actual en la empresa Nilsen es la acumulación excesiva de facturas y tickets de compra en formato físico. Este método de gestión manual dificulta considerablemente la realización de declaraciones fiscales, auditorías efectivas y una óptima gestión de cobros. A medida que la empresa ha crecido, la cantidad de documentos físicos ha aumentado, lo que complica aún más el acceso rápido a información financiera crítica. La introducción manual de datos en hojas de cálculo y registros en papel también incrementa el riesgo de errores, afectando la precisión de los registros financieros. Esta situación no solo representa un desafío operativo, sino que también impide la generación ágil de informes necesarios para la toma de decisiones estratégicas y el cumplimiento normativo.

#### 3.1.2 Diagrama del proceso del sistema actual



Figura 12. Diagrama de sistema actual

### **3.1.3 Detección de problemas y necesidades**

#### **Problemas**

- Dificultades para gestionar eficientemente un gran volumen de facturas y tickets de compra en formato físico.
- Limitaciones en la precisión y actualización de los registros financieros debido a métodos manuales de entrada de datos.
- Tiempos prolongados en la recuperación de documentos críticos para auditorías y reportes financieros.
- Riesgos de pérdida, daño o extravió de documentos físicos importantes para la empresa.

#### **Necesidades**

- Implementación de un sistema digital para la gestión segura y organizada de facturas y documentos financieros.
- Mejora en la accesibilidad y búsqueda rápida de información financiera crucial.
- Automatización de procesos para reducir errores humanos y aumentar la precisión en la gestión de registros financieros.
- Capacitación del personal en el uso efectivo de herramientas digitales para optimizar la gestión documental y mejorar la eficiencia operativa.

### **3.1.4 Estudios de factibilidad**

#### **3.1.4.1 Operativa**

En el apartado que pueden visualizar los clientes que no tienen una cuenta, la aplicación web posee una página de bienvenida con un menú de inicio que presenta una descripción de las funcionalidades principales. También se incluye un formulario de registro para la creación de nuevas cuentas y acceso a detalles de contacto o soporte para cualquier consulta adicional. Estas características aseguran que los usuarios tengan toda la información necesaria para comenzar a utilizar la aplicación sin problemas.

Los administradores disponen de un menú de inicio que proporciona acceso rápido a las funcionalidades principales, así como opciones para la subida y procesamiento de archivos de facturas. Además, pueden acceder a archivos recientes y ver estadísticas de las facturas procesadas. La aplicación también cuenta con una sección de categorías para visualizar y añadir nuevas categorías de gastos, una sección de facturas para mostrar y revisar facturas escaneadas, y una sección de historial que registra todas las facturas procesadas con detalles específicos. Estas áreas están diseñadas para facilitar una gestión eficiente y organizada de la información, garantizando que todas las tareas se realicen de manera fluida y estructurada.

### 3.1.4.2 Economía

En la presente tabla, se muestran los costos por mes de cada persona involucrada en el proyecto

Rol	Pago mensual
Líder de desarrollo	\$26,151 pesos
Programador	\$15,820 pesos
Especialista de la base de datos	\$20,000 pesos
Analista	\$12,720 pesos
Tester	\$11,530 pesos
Total	\$86,221 pesos

Tabla 1: Tabla Económica

**Fuente:** Secretaría de Economía (SE) 2024, Desarrolladores y Analistas de Software.

### 3.1.4.3 Técnicas

La aplicación web tiende a trabajar más por el lado del servidor, por ende, no se necesita contar con una computadora potente, los requisitos recomendados se encuentran con detalle en la siguiente tabla.

Dispositivo	Requerimiento
Procesador	i5-8259U
Memoria RAM	8GB DDR4 o superior
Almacenamiento	480 GB SSD o superior
Sistema Operativo	Windows 10 o superior
Otros	Conexión a internet de 80Mbps o más

Tabla 2: Tabla Técnica

### 3.1.5 Alternativa de solución

#### 3.1.5.1 Nombre de la alternativa

Aplicación web de digitalización, almacenamiento y consulta de tickets o recibos de compra

#### 3.1.5.2 Descripción de la alternativa

Esta aplicación web permite organizar y gestionar los tickets o recibos de compra, dando a los usuarios la capacidad de subir, digitalizar y consultar estos documentos. Utilizando tecnología de reconocimiento óptico de caracteres (OCR), el sistema extrae automáticamente la información clave, como la empresa emisora, los productos comprados y el total de la compra. Además, ofrece la opción de generar reportes en formato Excel con un registro detallado de todos los tickets.

#### 3.1.5.3 Mapa de navegación o árbol del sitio

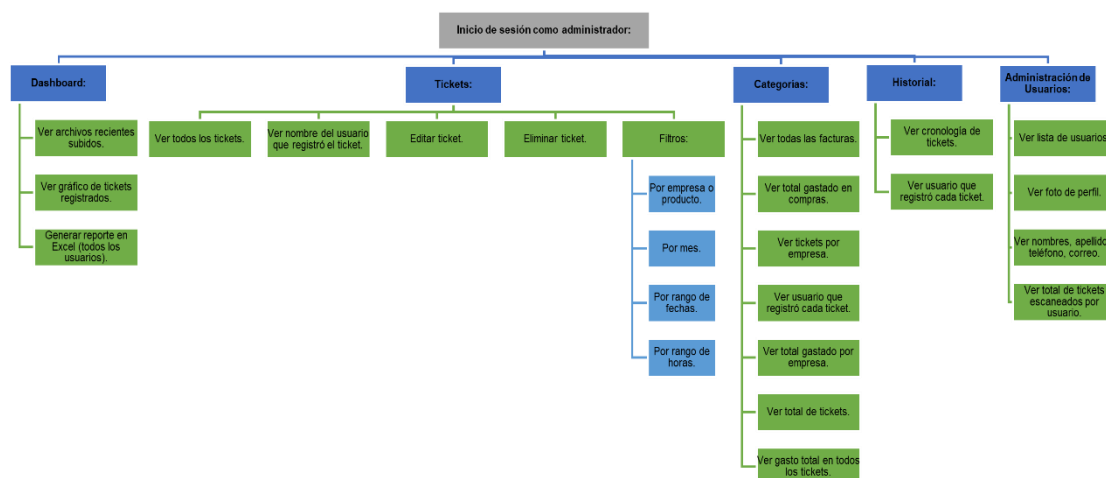


Figura 13. Mapa de navegación de un administrador



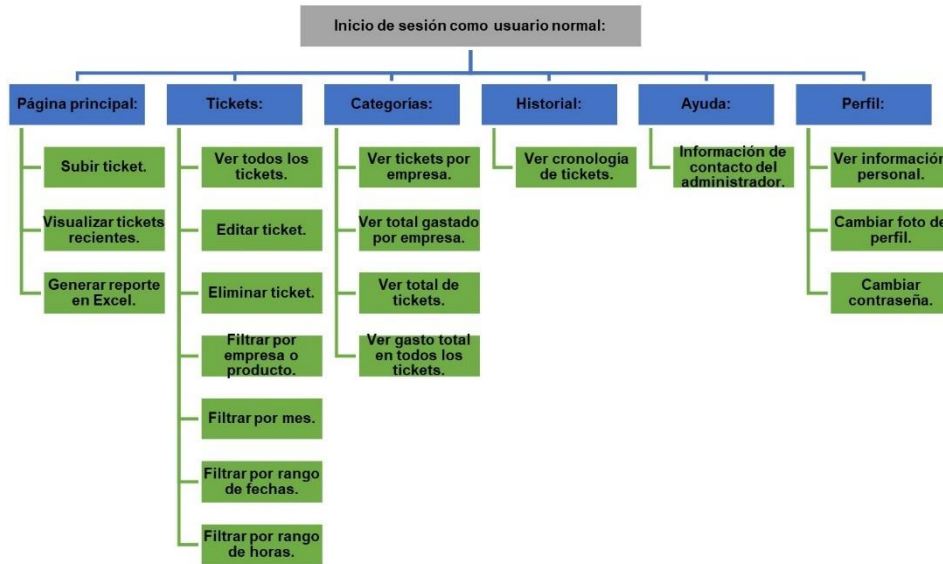


Figura 14. Mapa de navegación de un usuario normal

### 3.1.5.4 Justificación

Esta aplicación web permite una gestión estructurada de tickets de compra, asegurando que la información se registre de manera precisa y esté siempre accesible. Con funcionalidades para digitalizar, almacenar y analizar datos de tickets, la aplicación proporciona una solución integral para el seguimiento de gastos y la administración de documentos. La capacidad de generar reportes detallados y realizar búsquedas específicas apoya una administración eficiente y un control riguroso de la información financiera.

### 3.1.5.5 Tecnologías web a utilizar

**Visual Studio Code:** Es un editor de código multiplataforma desarrollado por Microsoft, conocido por su soporte extensivo de extensiones y plugin. Ofrece funcionalidades como depuración integrada, control de versiones con Git, resaltado de sintaxis, completado automático de código, y refactorización, facilitando el desarrollo y mantenimiento del código.

**HTML:** Es el lenguaje de marcado que define la estructura y el contenido de las páginas web. Utiliza una serie de etiquetas para organizar y presentar texto, imágenes, enlaces, y otros elementos en la interfaz de usuario.

**CSS:** Es un lenguaje de hojas de estilo que se utiliza para definir la presentación y el diseño visual de las páginas web. Permite controlar aspectos como el diseño de la página, los colores, las fuentes y el espaciado de los elementos.

**PHP:** Es un lenguaje de programación del lado del servidor utilizado para el desarrollo de aplicaciones web y la creación de páginas dinámicas. Facilita la interacción entre el servidor y el navegador del usuario, gestionando la lógica de la aplicación y el acceso a bases de datos.

**JavaScript:** Es un lenguaje de programación del lado del cliente que permite añadir interactividad a las páginas web. Se utiliza para manipular el contenido de la página en respuesta a eventos del usuario, realizar validaciones, y mejorar la experiencia del usuario con elementos dinámicos.

**AJAX:** Es una técnica de desarrollo web que permite actualizar partes de una página web de manera asíncrona sin recargar la página completa. Utiliza JavaScript para realizar solicitudes al servidor y actualizar el contenido en tiempo real.

**jQuery:** Es una biblioteca de JavaScript que simplifica la manipulación del DOM (Document Object Model), el manejo de eventos, y las solicitudes AJAX. Proporciona una interfaz más sencilla y consistente para realizar operaciones comunes en JavaScript.

**API de Google Gemini para OCR:** Es una herramienta de reconocimiento óptico de caracteres (OCR) proporcionada por Google que convierte imágenes de texto en texto editable. Permite extraer información de documentos para su almacenamiento y análisis.

**MySQL:** Es un sistema de gestión de bases de datos relacional de código abierto. Se utiliza para almacenar, organizar y gestionar grandes volúmenes de datos, permitiendo realizar consultas y operaciones complejas sobre la información almacenada.

## 3.2 Diseño gráfico de la aplicación

Pantalla de inicio para administradores en la aplicación web, con módulos de Dashboard, Tickets, Categorías, Historial y Administración de usuarios. El Dashboard incluye un botón para subir tickets, secciones para visualizar archivos recientes y un gráfico de tickets subidos anualmente, además de un botón para generar reportes en Excel con todos los datos de los tickets.

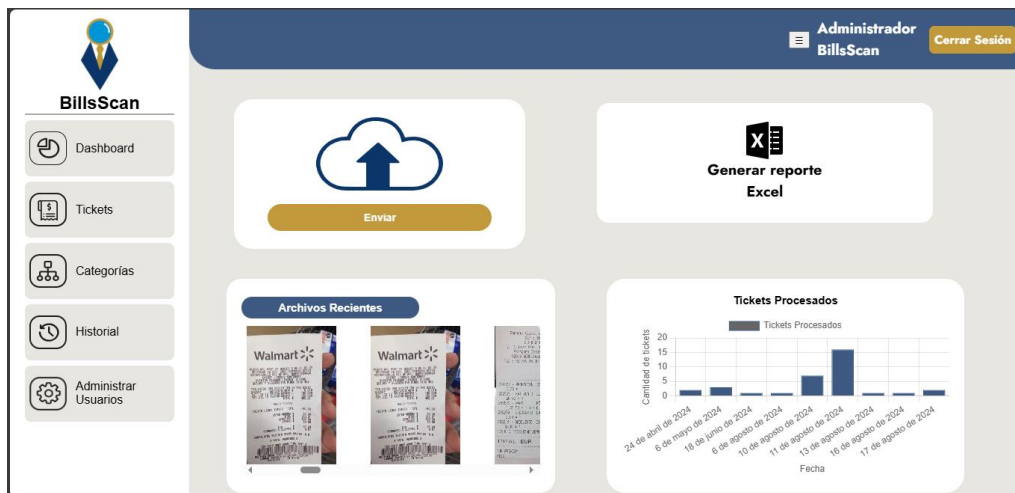


Figura 15. Inicio del usuario tipo administrador

La pantalla principal presenta un botón central para enviar archivos y otro para generar reportes en Excel. A la izquierda, un menú vertical permite acceder a "Tickets", "Categorías" e "Historial". A la derecha, se muestran archivos recientes y un gráfico de barras con la cantidad de tickets procesados. Ofrece acceso rápido a funciones clave y una visión general de la actividad reciente en la aplicación.

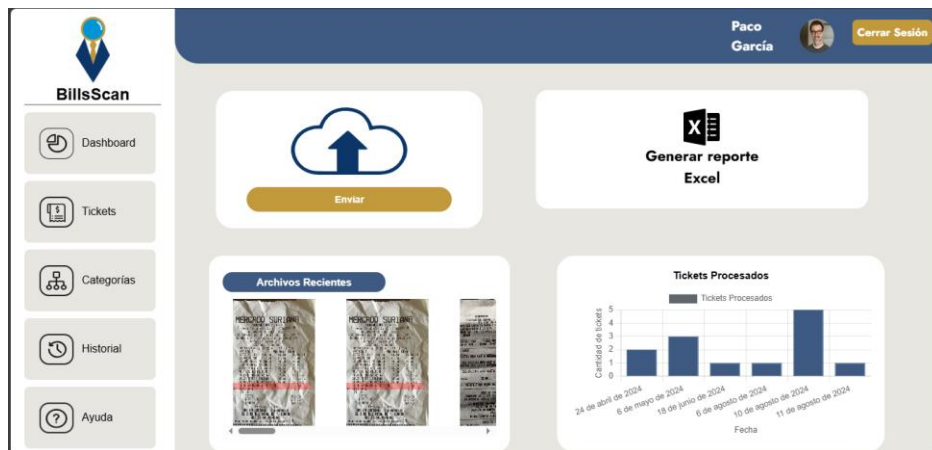


Figura 16. Inicio del usuario de tipo normal

### 3.3 Aplicación de las etapas del modelo de desarrollo de sistemas de información.

Para la realización de la aplicación web se utilizó el modelo en V, este modelo se eligió por su estructura rígida y secuencial que permite un enfoque en la verificación y validación en cada etapa del desarrollo, asegurando la calidad del producto final a medida que se avanza en el proyecto.

#### Fase de Definición:

- **Requisitos:** Antes de comenzar con el desarrollo de la aplicación web, se llevó a cabo una recopilación exhaustiva de los requisitos, donde se entrevistó a los usuarios para entender las características y necesidades que optimizarían su trabajo. Se seleccionaron las herramientas necesarias para cumplir con estos requisitos.
- **Especificación del Sistema:** En esta etapa, se realizó la especificación completa del sistema, definiendo los módulos, componentes y la interacción entre ellos. Se documentó el diseño a nivel alto y se decidió la estructura general de la aplicación llamada "Adobe Illustrator".
- **Diseño de Arquitectura:** A partir de la especificación del sistema, se diseñó la arquitectura que soportaría la aplicación, definiendo la estructura y los componentes necesarios. Se establecieron las tecnologías a utilizar, como PHP y la base de datos, para asegurar la viabilidad técnica del proyecto.

#### Fase de Desarrollo:

- **Diseño Detallado:** Durante esta etapa, se desglosaron los módulos y componentes en diseños más detallados. Se seleccionaron los estilos, colores y la estructura de las páginas que compondrían la aplicación. Aquí se comenzó a delinear cómo sería la interfaz final, asegurando que se implementaran los requisitos definidos anteriormente.
- **Implementación:** Una vez completado el diseño detallado, se procedió a la implementación del código. Cada módulo fue programado y ensamblado de acuerdo con el diseño especificado. La implementación se llevó a cabo utilizando PHP junto con una base de datos para garantizar la funcionalidad del sistema.

#### Fase de Verificación y Validación:

- **Pruebas de Unidad:** Se realizaron pruebas unitarias sobre cada componente desarrollado, verificando que cada módulo funcionara correctamente de manera aislada. Esto permitió identificar y corregir errores a medida que se avanzaba en la codificación.
- **Integración y Pruebas del Sistema:** Una vez que todos los módulos fueron desarrollados y probados de manera individual, se integraron y se realizaron pruebas del sistema en conjunto. Esto permitió asegurar que todos los componentes funcionaran de manera coherente y que la aplicación cumpliera con los requisitos especificados.
- **Entrega del Producto Final:** En esta etapa final, la aplicación completa fue sometida a una revisión exhaustiva para asegurar que cumpliera con las expectativas del cliente y que todos

los requerimientos se hubieran implementado correctamente. Se corrigieron detalles finales y se entregó la primera versión de la aplicación web junto con un video explicativo donde se detallaron todas sus características, funciones y cómo utilizar la plataforma de manera efectiva. Este video sirvió como una guía para el cliente, facilitando la comprensión del producto final y su uso.

### 3.4 Descripción del desarrollo de la aplicación

#### 3.4.1 Diagrama de la base de datos

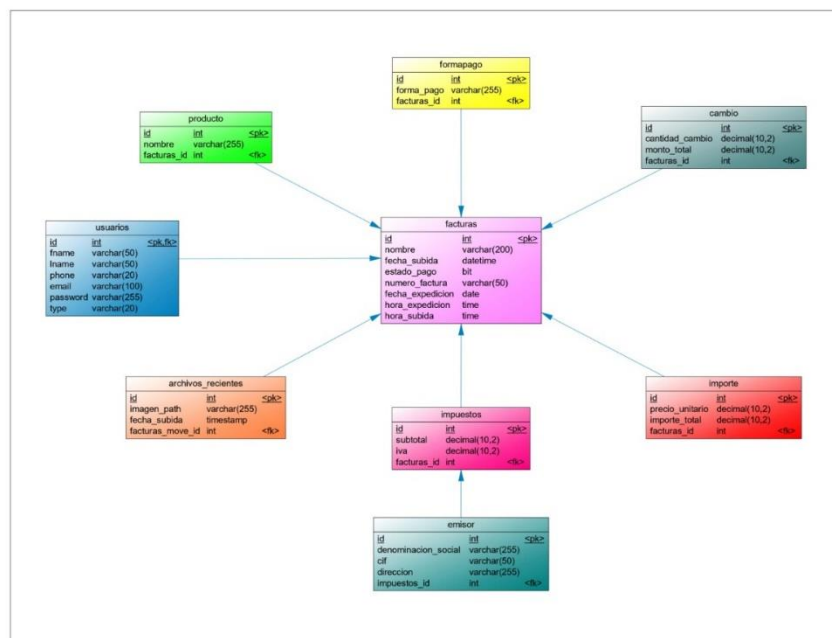


Figura 17. Diagrama de la base de datos

#### 3.4.2 Diccionario de datos

Facturas

Columna	Tipo	Nulo
id	Int	No
nombre	Varchar (20)	No
fecha_subida	datetime	No
estado_pago	bit	No
numero_factura	Varchar (20)	No
fecha_expedicion	date	No
hora_expedicion	time	No
hora_subida	time	No

Tabla 3: Diccionario de Facturas

## Emisores

Columna	Tipo	Nulo
id	Int	No
denominacion_social	Varchar (255)	No
cif	Varchar (20)	No
direccion	Varchar (255)	No
impuestos_id	Int	No

Tabla 4: Diccionario de Emisores

## Cambio

Columna	Tipo	Nulo
id	Int	No
cantidad_cambio	decimal (10,2)	No
monto_total	decimal (10,2)	No
facturas_id	Int	No

Tabla 5: Diccionario de Cambio

## Productos

Columna	Tipo	Nulo
id	Int	No
nombre	Varchar (255)	No
facturas_id	Int	No

Tabla 6: Diccionario de Productos

## Impuestos

Columna	Tipo	Nulo
id	Int	No
subtotal	decimal (10,2)	No
iva	decimal (10,2)	No
facturas_id	Int	No

Tabla 7: Diccionario de Impuestos

## Archivos recientes

Columna	Tipo	Nulo
id	Int	No
imagen_path	Varchar (255)	No
fecha_subida	timestamp	No
facturas_move_id	Int	No

Tabla 8: Diccionario de Archivos recientes

## Importe

Columna	Tipo	Nulo
id	Int	No
precio_unitario	decimal (10,2)	No
importe_total	decimal (10,2)	No
facturas_id	Int	No

Tabla 9: Diccionario de Importe



## Usuarios

Columna	Tipo	Nulo
id	Int	No
fname	Varchar (255)	No
lname	Varchar (255)	No
phone	Varchar (255)	No
email	Varchar (100)	No
password	Varchar (255)	No
type	Varchar (20)	No

Tabla 10: Diccionario de Usuarios

### 3.4.3 Creación de la aplicación web

Inicialmente, se establecieron directorios para organizar los archivos requeridos para el funcionamiento de la aplicación web. Estos directorios son:

- **css:** Esta carpeta, como su nombre indica, contiene todos los archivos de estilos en cascada (CSS) que se utilizarán para dar formato y diseño a tu página web.
- **imágenes:** En esta carpeta se almacenan todas las imágenes que se utilizarán en tu sitio web.
- **js:** Aquí se encuentran todos los archivos JavaScript que contienen el código para añadir interactividad a tu página web.
- **php:** Esta carpeta contiene los archivos PHP que se encargan de la lógica del servidor.
- **plantilla:** Probablemente esta carpeta contiene las plantillas HTML que sirven como base para construir las diferentes páginas de la aplicación web.

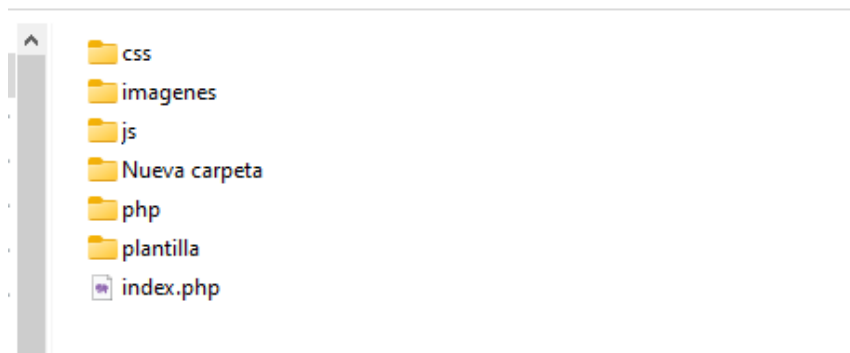


Figura 18. Carpetas del proyecto



Posteriormente, se accederá a las carpetas mencionadas utilizando el editor de código Visual Studio Code.

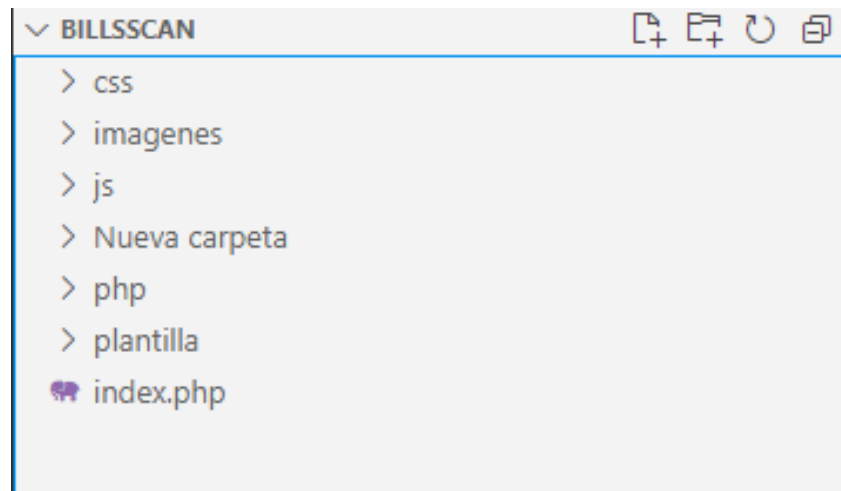


Figura 19. Carpetas del proyecto en Visual Studio Code

Se procedió a crear la base de datos en phpMyAdmin, implementando la estructura definida en el diagrama entidad-relación.

Tabla	Acción	Filas	Tipo	Cotejamiento	Tamaño	Residuo a depurar
archivos_recientes	Examinar Estructura Buscar Insertar Vaciar Eliminar	31	InnoDB	utf8mb4_general_ci	48.0 KB	-
cambio	Examinar Estructura Buscar Insertar Vaciar Eliminar	31	InnoDB	utf8mb4_general_ci	32.0 KB	-
emisor	Examinar Estructura Buscar Insertar Vaciar Eliminar	31	InnoDB	utf8mb4_general_ci	16.0 KB	-
facturas	Examinar Estructura Buscar Insertar Vaciar Eliminar	31	InnoDB	utf8mb4_general_ci	64.0 KB	-
facturas_move	Examinar Estructura Buscar Insertar Vaciar Eliminar	34	InnoDB	utf8_spanish2_ci	32.0 KB	-
formapago	Examinar Estructura Buscar Insertar Vaciar Eliminar	31	InnoDB	utf8mb4_general_ci	32.0 KB	-
importe	Examinar Estructura Buscar Insertar Vaciar Eliminar	31	InnoDB	utf8mb4_general_ci	32.0 KB	-
impuestos	Examinar Estructura Buscar Insertar Vaciar Eliminar	31	InnoDB	utf8mb4_general_ci	32.0 KB	-
producto	Examinar Estructura Buscar Insertar Vaciar Eliminar	31	InnoDB	utf8mb4_general_ci	32.0 KB	-
user	Examinar Estructura Buscar Insertar Vaciar Eliminar	3	InnoDB	utf8mb4_general_ci	16.0 KB	-
10 tablas	Número de filas	285	InnoDB	utf8_spanish2_ci	336.0 KB	0 8

Figura 20. Tablas de la base de datos

### 3.4.4 Diseño de páginas de la aplicación web

Pantalla de inicio para usuarios administradores dentro de la aplicación web. Se observan los siguientes módulos: Dashboard, Tickets, Categorías, Historial y Administración de usuarios. El Dashboard contiene elementos como un botón para subir tickets, una sección para visualizar los archivos recientes, otra que muestra un gráfico de la cantidad de tickets subidos por todos los usuarios a lo largo del año y un botón para generar un reporte en formato Excel con todos los datos de cada ticket de cada usuario.



Figura 21. Inicio del usuario tipo administrador

La pantalla de tickets permite al administrador visualizar todos los tickets que han sido subidos. Esta tecnología extrae información clave de cada ticket, como la empresa emisora, los productos comprados, el total de la compra, el cambio, los impuestos aplicados y la dirección del establecimiento.

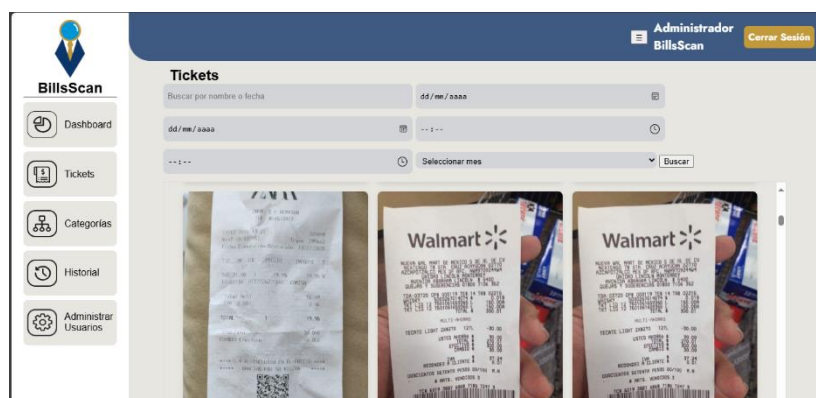


Figura 22. Pantalla de tickets del administrador

La pantalla de "Categorías" ofrece al administrador una vista organizada de los gastos, agrupando los tickets por empresa. Para cada empresa, se muestra el gasto total y el número de tickets asociados. Un buscador permite filtrar las categorías por nombre, y un resumen general en la parte superior indica el gasto total de todos los empleados y el número total de facturas registradas en el sistema.

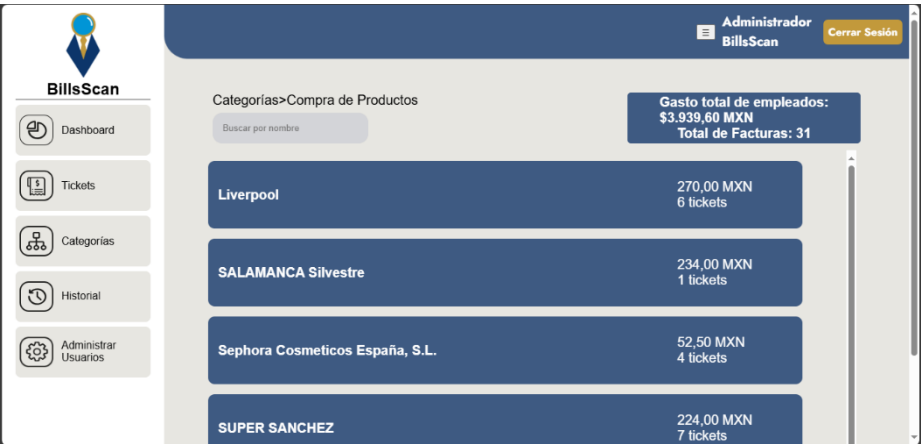


Figura 23. Pantalla de categorías del administrador

La pantalla "Historial" presenta al administrador un registro cronológico de los tickets subidos al sistema. Cada entrada en la historial muestra la empresa donde se realizó la compra, el producto adquirido, la fecha y hora exacta de registro del ticket, y el empleado responsable de subirlo. Esta vista permite al administrador rastrear la actividad de los empleados, llevar un control de los gastos a lo largo del tiempo y tener una visión general del uso de la plataforma.

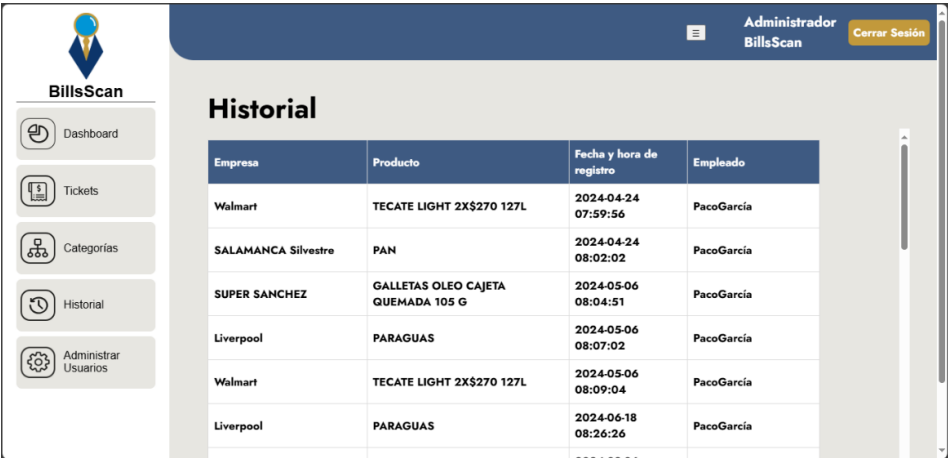


Figura 24. Pantalla de historial del administrador

La pantalla "Administrar empleados" permite al administrador controlar los usuarios del sistema. Muestra una lista de empleados con detalles como nombre, información de contacto, actividad en la plataforma y tipo de usuario. El administrador puede agregar, editar o eliminar usuarios según sea necesario, garantizando la seguridad y el control de acceso a la aplicación.



Figura 25. Pantalla de Administrar usuarios

La pantalla principal ofrece en la parte central, un botón destacado permite enviar nuevos archivos para su procesamiento, y otro genera reportes en Excel. A la izquierda, un menú vertical facilita la navegación a secciones como "Tickets", "Categorías" e "Historial". En la derecha, se visualizan los archivos recientes y un gráfico de barras muestra la cantidad de tickets procesados a lo largo del tiempo. Esta pantalla principal proporciona un acceso rápido a las funciones clave y una visión general de la actividad reciente en la aplicación.

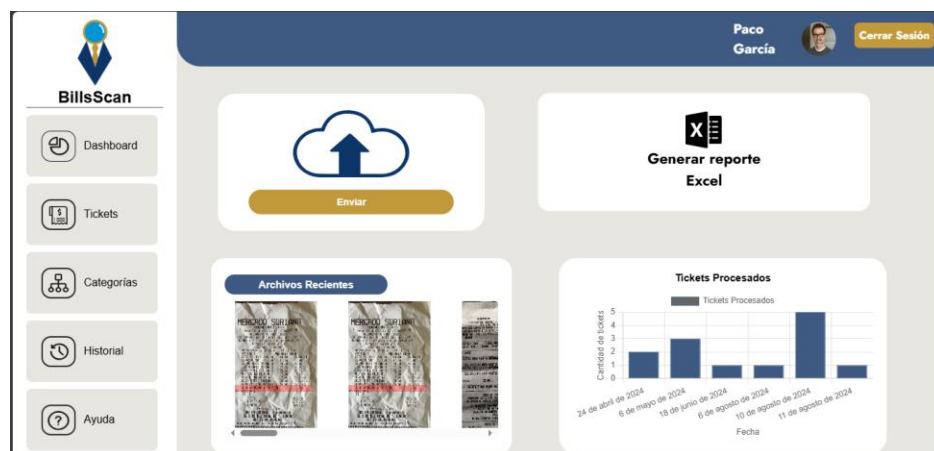


Figura 26. Inicio del usuario de tipo normal

La pantalla "Tickets" presenta una interfaz para visualizar y gestionar los tickets subidos al sistema. Los usuarios pueden buscar tickets específicos utilizando filtros por fechas o seleccionando un mes en particular. Cada ticket muestra información clave, como la empresa emisora, el producto comprado, el total, el cambio y la fecha de registro. Esta pantalla permite una revisión eficiente de los gastos y facilita la búsqueda de tickets concretos gracias a sus opciones de filtrado.

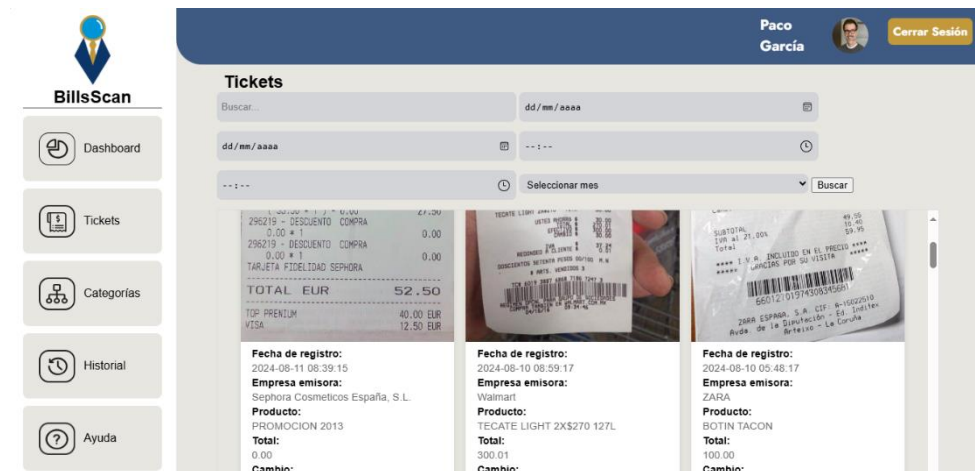


Figura 27. Pantalla de tickets del usuario

La pantalla "Categorías" permite ver de manera organizada los gastos, agrupándolos por empresa. Para cada empresa, se muestra el gasto total y el número de tickets asociados. Un buscador permite filtrar las categorías por nombre, y un resumen general indica el gasto y el número totales de facturas.

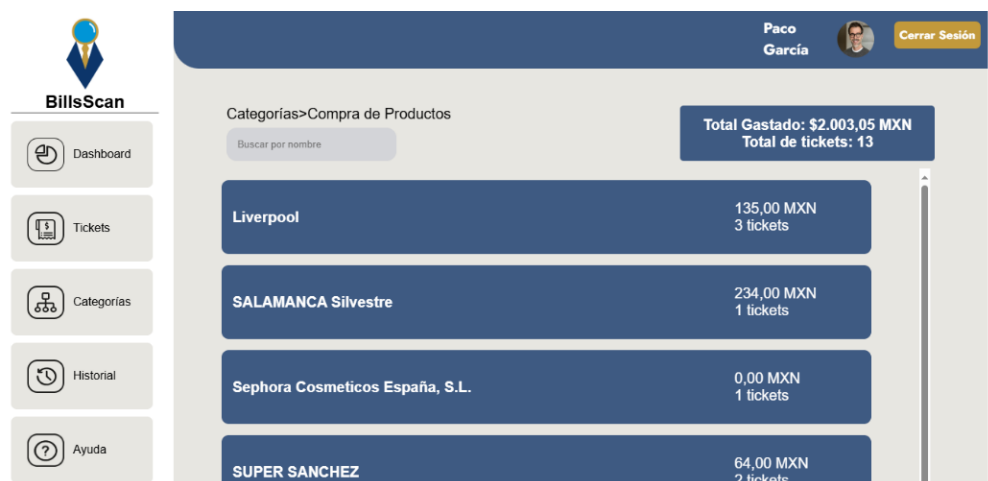
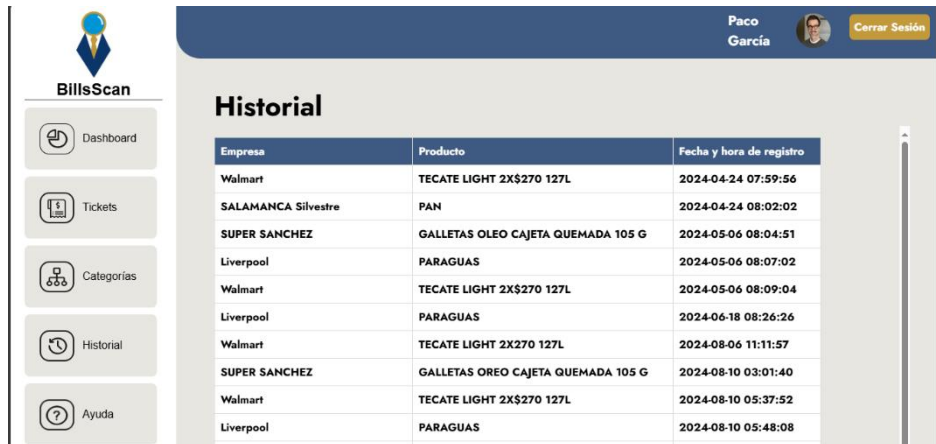


Figura 28. Pantalla de categorías del usuario

La pantalla "Historial" ofrece al usuario un registro cronológico de sus compras, mostrando la empresa, el producto y la fecha y hora de registro de cada ticket. Esta vista le permite realizar un seguimiento de sus gastos a lo largo del tiempo y comprender sus patrones de consumo.



Empresa	Producto	Fecha y hora de registro
Walmart	TECATE LIGHT 2X\$270 127L	2024-04-24 07:59:56
SALAMANCA Silvestre	PAN	2024-04-24 08:02:02
SUPER SANCHEZ	GALLETAS OLEO CAJETA QUEMADA 105 G	2024-05-06 08:04:51
Liverpool	PARAGUAS	2024-05-06 08:07:02
Walmart	TECATE LIGHT 2X\$270 127L	2024-05-06 08:09:04
Liverpool	PARAGUAS	2024-06-18 08:26:26
Walmart	TECATE LIGHT 2X270 127L	2024-08-06 11:11:57
SUPER SANCHEZ	GALLETAS OREO CAJETA QUEMADA 105 G	2024-08-10 03:01:40
Walmart	TECATE LIGHT 2X\$270 127L	2024-08-10 05:37:52
Liverpool	PARAGUAS	2024-08-10 05:48:08

Figura 29. Pantalla de historial del usuario



La pantalla "Ayuda" proporciona a los usuarios la información de contacto necesaria para obtener soporte o asistencia. Incluye un número de teléfono, una dirección de correo electrónico y la opción de enviar un mensaje a través de WhatsApp, brindando flexibilidad en la comunicación con el administrador.



**Figura 30. Pantalla de ayuda del usuario**

## CAPÍTULO IV CONCLUSIONES

En síntesis, la creación de la aplicación web para la gestión de facturas en la empresa Nilsen ha permitido abordar de manera efectiva los problemas asociados con la acumulación de documentos físicos y la ineficiencia en la gestión manual de datos. Al implementar un sistema que digitaliza, organiza y facilita la consulta de tickets y recibos, se ha mejorado significativamente la precisión de los registros financieros y se ha optimizado el tiempo de recuperación de información crítica.

Las implicaciones de este desarrollo son amplias, ya que no solo benefician a Nilsen al mejorar su operativa interna, sino que también sirven como un modelo para otras organizaciones que buscan modernizar sus procesos administrativos. La digitalización de documentos es un paso esencial hacia la mejora de la eficiencia y la reducción de riesgos asociados con la gestión de información en formato físico.

Para el futuro, se sugiere que se realicen estudios adicionales sobre la integración de herramientas de análisis de datos y reportes automatizados, lo que podría enriquecer aún más la funcionalidad de la aplicación. Asimismo, es fundamental que se ofrezca capacitación continua al personal para maximizar el uso de estas herramientas digitales.

En conclusión, la adopción de esta aplicación web no solo representa un avance significativo en la gestión de facturas, sino que también subraya la necesidad de que las empresas se adapten a las nuevas tecnologías para mantenerse competitivas en un entorno empresarial en constante evolución.