UNIVERSIDAD DEL VALLE DE GUATEMALA

Corte #3

BRANDON JAVIER REYES MORALES - 22992
CARLOS ALBERTO VALLADARES GUERRA - 221164
GUSTAVO ADOLFO CRUZ BARDALES- 22779
PAULA REBECA BARILLAS ALVAREZ - 22764
RODRIGO ALFONSO MANSILLA DUBÓN - 22611

Catedrático: Lynette García Pérez

Ingeniería de Software
Guatemala, 2024

I. Resumen

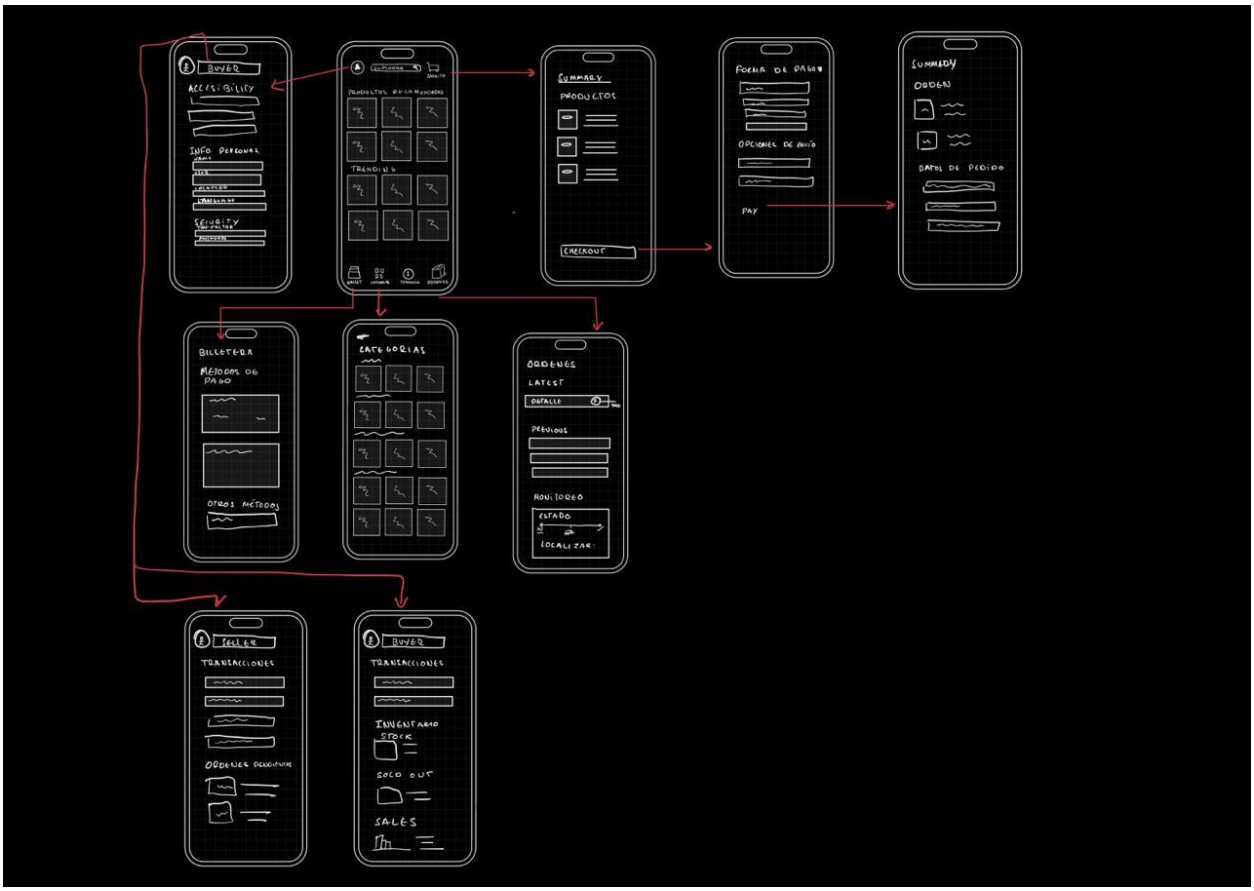
El trabajo se enfoca en desarrollar una plataforma de comercio electrónico adaptada a las necesidades de emprendedores y compradores casuales. Esta plataforma busca atender las demandas de seguridad en transacciones, gestión eficiente de inventarios y una experiencia de compra intuitiva y accesible. La necesidad de este proyecto surge de la creciente tendencia del comercio electrónico y la importancia de apoyar a pequeños emprendedores en este entorno digital. Los objetivos concretos incluyen el diseño de una interfaz de usuario amigable, implementación de sistemas seguros de transacciones y la integración de herramientas de gestión de inventario eficaces para facilitar el crecimiento de pequeños y medianos negocios en el mercado digital.

II. Introducción

Los emprendedores y vendedores casuales operan generalmente a través de plataformas de redes sociales, estás plataformas presentan una oportunidad de expansión a los emprendedores y permiten expandir sus redes de ventas. Debido a las limitaciones de las diversas plataformas de operación, buscan una forma más segura y eficiente de realizar transacciones y gestionar sus negocios en linea. Sin embargo, los procesos de compra-venta pueden ser tediosos. Esto debido a las formas de transacción, la falta de seguridad en las mismas y la posibilidad de fraudes. Estos desafíos representan limitaciones al crecimiento de los emprendedores y comprometen su eficacia. En consecuencia, es necesaria la implementación de una solución que satisfaga las necesidades de los emprendimientos, vendedores casuales y compradores. El objetivo general es presentar un plan detallado que permita abordar y comprender la problemática de mejor manera y desarrollar una solución al mismo. Los objetivos específicos incluyen el análisis de las necesidades , comprender a los posibles usuarios y obtener insights que favorezcan el desarrollo de una solución eficiente a la problemática.

II. Design Thinking

a. Prototipos



Cambios con respecto a la versión anterior:

- Se incluyo una nueva pantalla dedicada al carrito de compras, que permite realizar un solo pago por varios artículos.
- Se incluyo una pantalla para vendedores y otra pantalla para compradores.

Cambios propuestos a través de entrevistas:

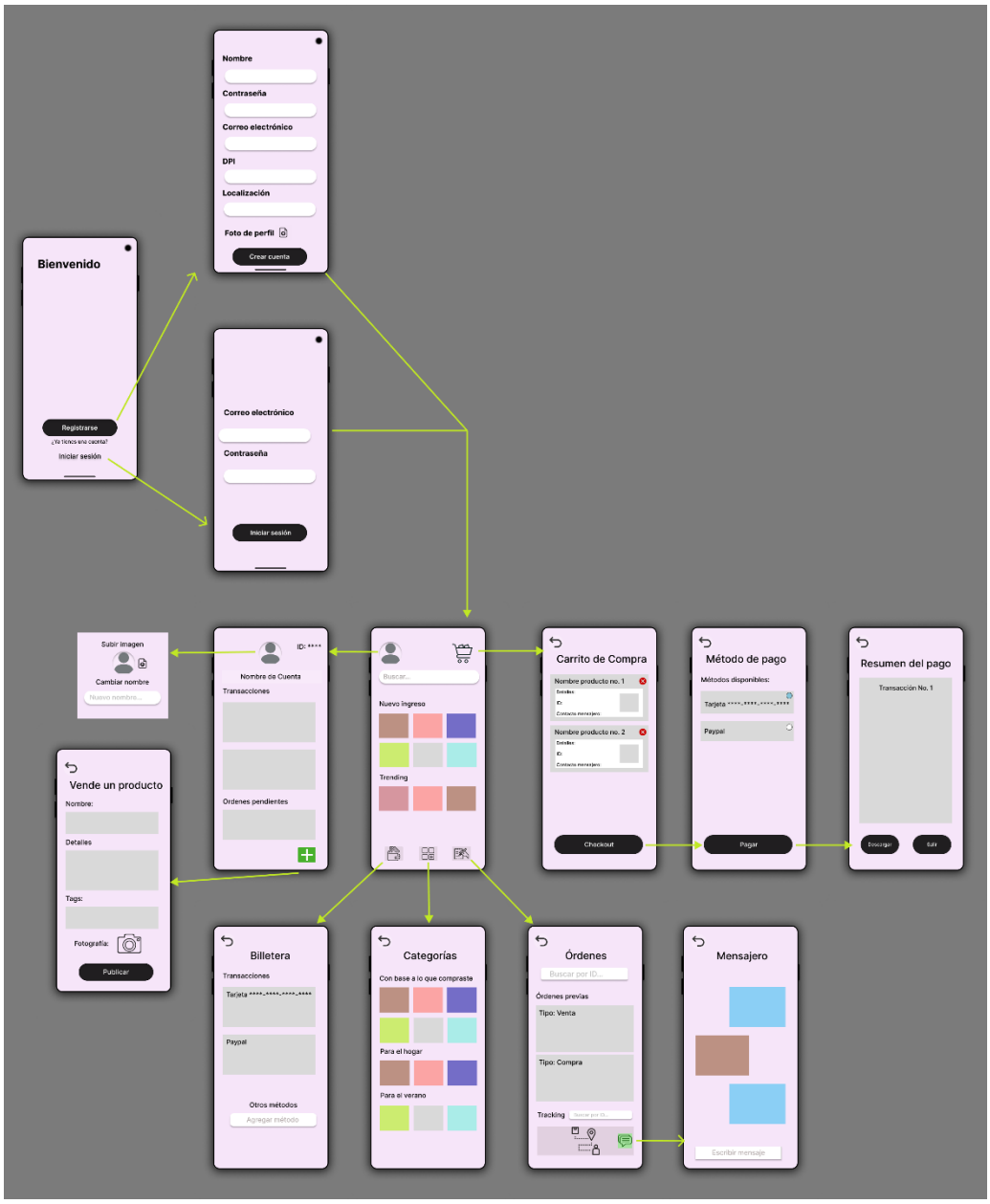
- Chat privado a mensajero, tanto de parte del vendedor como del comprador, para asegurar un tracking del producto en tiempo real.
- Permitir que las opciones de vendedor y comprador se puedan utilizar al mismo tiempo.
- Que los datos de las transacciones se puedan guardar desde la app, no únicamente por captura de pantalla.
- Modificar el perfil para cambiar de foto y nombre

A partir de la retroalimentación, se determinaron los siguientes requisitos funcionales:

- Chat privado con mensajero.
- Botón de captura desde la aplicación.
- Botón que permita modificar nombre y foto de perfil.

A partir de la retroalimentación, se determinaron los siguientes requisitos no funcionales:

- Unificar opciones de comprador y vendedor en la misma sesión, sin necesidad de cambiar manualmente el tipo de cuenta.
- Implementar un ID único para identificar perfiles, que no se pueda cambiar, para asegurar la veracidad de un perfil, aunque cambie de foto y nombre.



Diseño pantalla completa en figma

Cambios con respecto a la versión anterior:

- Se permite las opciones de comprador y vendedor en la misma sesión, sin necesitar de un botón físico que cambia entre tipo de cuenta.
- Se implemento un ID a la cuenta de cada usuario, con el objetivo de identificar cada
- Se implemento un botón de guardar transacción.
- Se implemento un botón para cambiar datos de perfil, sin afectar la veracidad del mismo.
- Se implemento un botón para iniciar un chat con el vendedor.
- Todas las búsquedas se hacen a través de un ID para asegurar que un individuo no está duplicado.

Cambios propuestos a través de entrevistas:

- Se puede implementar una ventana de compra sin necesidad de mandar a otra pantalla, para que el proceso sea más rápido.
- Se puede incorporar una sección de ajustes para cambiar tamaño de letra, tema oscuro, etc.

A partir de la retroalimentación, se determinaron los siguientes requisitos funcionales:

- Botón para nueva ventana en cambio de datos de perfil.
- Botón a menú de ajustes, con opciones de accesibilidad.

A partir de la retroalimentación, se determinaron los siguientes requisitos no funcionales:

- No se identificaron a partir de lo propuesto, sin embargo, se considera que será necesario hacer que el sistema aplique un blur al resto de la pantalla al aparecer una ventana flotante, para que la atención se concentre en la ventanca y no el resto de la pantalla.

Testeo de prototipos:

III. Análisis

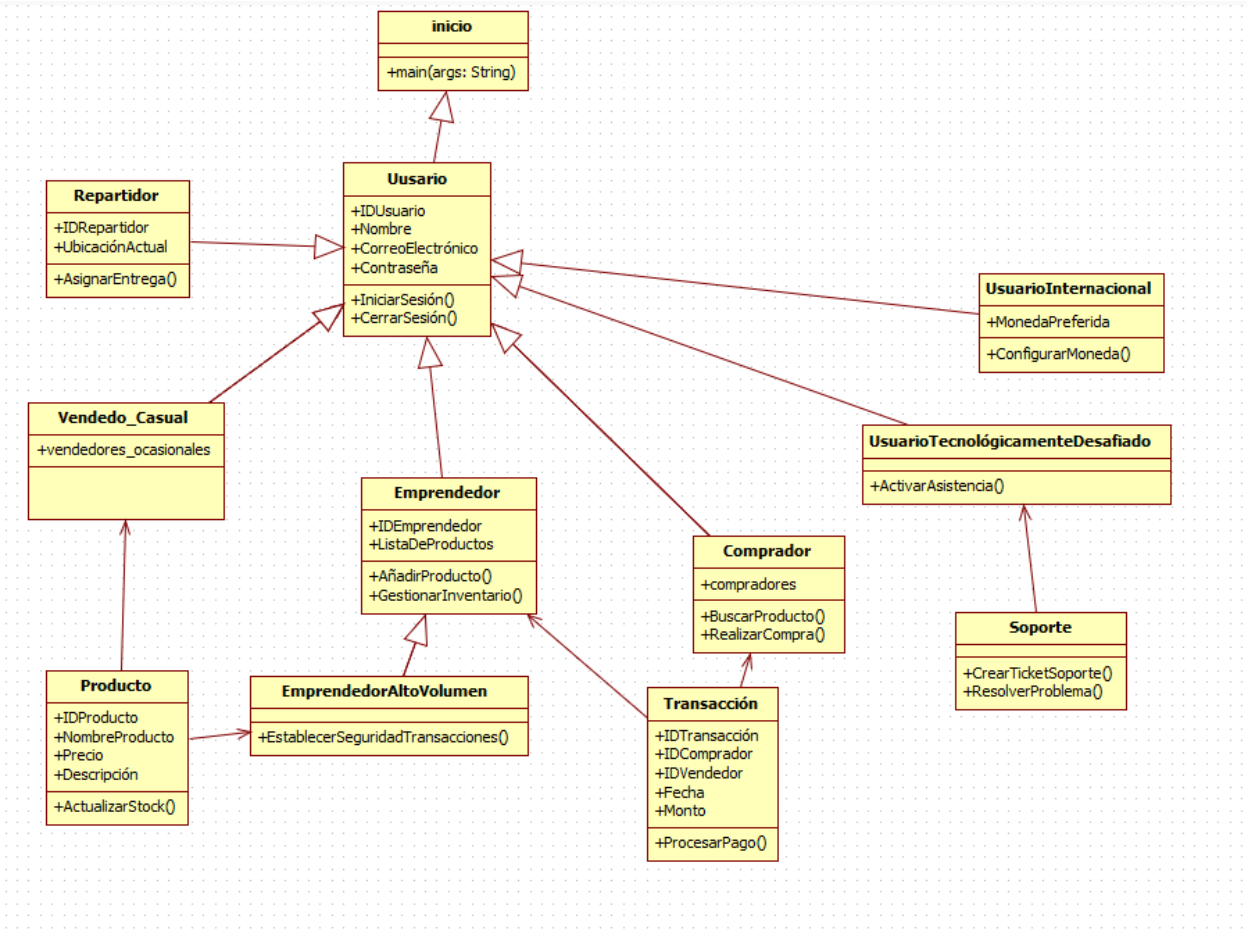
a. Lista de Requerimientos Funcionales

Historia	Requisito
Emprendedor	• Sistema De Gestión de inventarios que actualice el stock después de cada transacción. • Análisis de ventas en tiempo real para identificar tendencias y ajustar estrategias • Panel de control para el seguimiento de pedidos y gestión de devoluciones.
Vendedor Casual	• Lista de productos con plantillas y asistencia. • Opciones de configuración de privacidad avanzadas. • Sistema de Valoraciones y reseñas. • Autenticación de dos factores para acceder al área de gestión de ventas.

Comprador	• Búsqueda Avanzada por medio de filtros por categoría, precio y ubicación del vendedor. • Pasarela de pago seguro con múltiples opciones de pago. • Sistema de garantía de compra para protección contra fraudes o productos no entregados. • Acceso a historial de compras detallado para el seguimiento de los gastos.
Emprendedor con volumen de ventas alto	• Integración con plataformas de pago con soporte transacciones a gran escala. • Sistema de detección de fraude y alertas en tiempo real. • Herramientas de CRM para gestionar relaciones con clientes y personalización de ofertas. • Funcionalidades de reporte y análisis financiero para el seguimiento del flujo de caja y la rentabilidad.
Usuario en situaciones de riesgo	• Opciones de pago anónimas o cifradas para la protección de la identidad y los datos financieros. • Sistema de revisión de seguridad de la cuenta con monitoreo de actividad inusual. • Opción para establecer límites de compra para prevenir el uso indebido en caso de robo de cuenta.
Usuario con conocimiento tecnológico limitado	• Interfaz de usuario intuitiva con navegación simple y diseño responsive. • Soporte al cliente mediante chat en vivo, chatbots , teléfono, y correo electrónico con guías paso a paso. • Tutoriales en vídeo y FAQs para las funciones más comunes de la plataforma.
Usuario Internacional	• Soporte multilingüe, atención al cliente, y documentación de ayuda. • Conversión automática de monedas basada en la ubicación del usuario.
Repartidor	• Funcionalidad de selección de pedidos por ubicación para repartidores. • Sistema degestión de itinerarios que optimice las rutas de entrega basadas en la geolocalización, datos en tiempo real y prioridad de los pedidos. • Sistema de calificación y feedback de los usuarios para mejorar la calidad del servicio de entrega. • Notificaciones en tiempo real sobre nuevos pedidos disponibles acorde a la ubicación y preferencias del repartidor. • Herramientas de seguimiento en tiempo real para los clientes, permitiendo a los repartidores actualizar el estado de la entrega.

b. Backend

i. Diagrama de Clases:



ii. Descripción de paquetes y componentes

Clase: Usuario

- Atributos:
  - IDUsuario: Un identificador único para el usuario.
  - Nombre: El nombre del usuario.
  - CorreoElectrónico: La dirección de correo electrónico del usuario.
  - Contraseña: La contraseña del usuario para acceder al sistema.
- Métodos:
  - IniciarSesión(): Permite al usuario iniciar sesión en el sistema.
  - CerrarSesión(): Permite al usuario cerrar su sesión en el sistema.

Subclase: Emprendedor (hereda de Usuario)

- Atributos:

    - IDEmprendedor: Un identificador único para el emprendedor.
    - ListaDeProductos: Una lista de productos que el emprendedor está manejando.
- Métodos:
    - AñadirProducto(): Agrega un producto a la lista de productos del emprendedor.
- GestionarInventario(): Administra el inventario de productos.

Subclase: VendedorCasual (hereda de Usuario)

- No se especifican atributos o métodos adicionales. Esto implica que VendedorCasual utiliza las características básicas de la clase Usuario.

Subclase: Comprador (hereda de Usuario)

- Métodos:
    - BuscarProducto(): Busca productos dentro del sistema.
    - RealizarCompra(): Ejecuta la acción de comprar un producto.

Subclase: EmprendedorAltoVolumen (hereda de Emprendedor)

- Métodos:
    - EstablecerSeguridadTransacciones(): Establece mecanismos de transacción seguros para ventas de alto volumen.

Subclase: UsuarioTecnológicamenteDesafiado (hereda de VendedorCasual)

- Métodos:
    - ActivarAsistencia(): Activa asistencia adicional para usuarios con dificultades tecnológicas.

Subclase: UsuarioInternacional (hereda de Comprador)

- Atributos**:**
    - MonedaPreferida: La moneda preferida del usuario internacional.
- Métodos:
    - ConfigurarMoneda(): Configura el sistema para utilizar la moneda preferida del usuario.

Clase: Repartidor

- Atributos:
    - IDRepartidor: Un identificador único para el repartidor.
    - UbicaciónActual: La ubicación actual del repartidor.
- Métodos:
    - AsignarEntrega(): Asigna una tarea de entrega al repartidor.

Clase: Producto

- Atributos:

    - IDProducto: Un identificador único para el producto.
    - NombreProducto: El nombre del producto.
    - Precio: El precio del producto.
    - Descripción: Una descripción del producto.
- Métodos:
    - ActualizarStock(): Actualiza la disponibilidad del stock del producto.

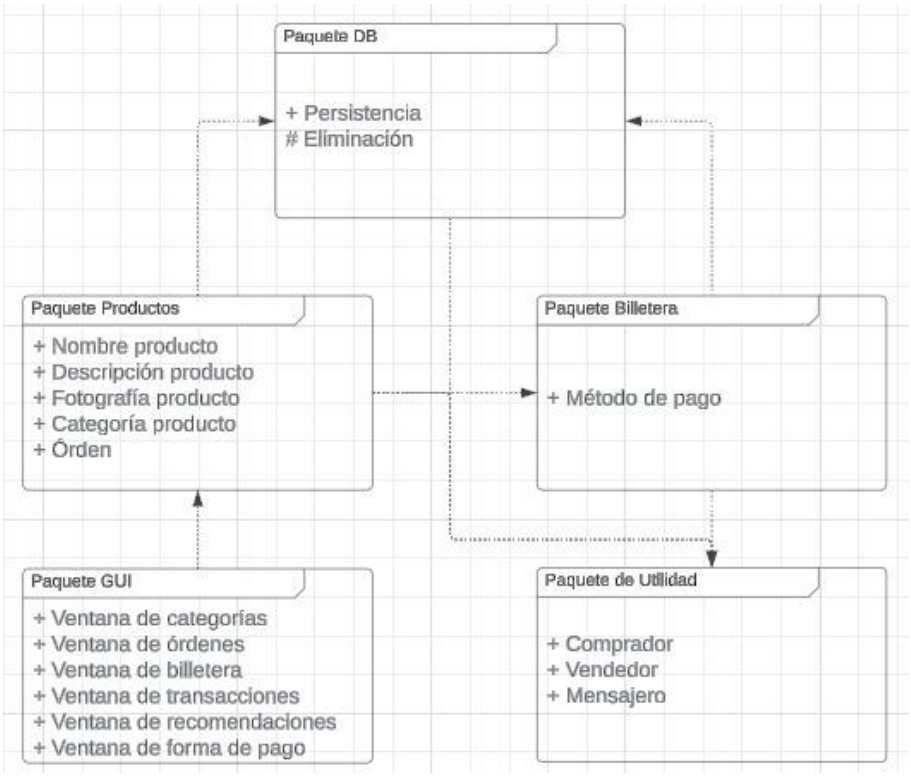
Clase: Transacción

- Atributos:
    - IDTransacción: Un identificador único para la transacción.
    - IDComprador: El identificador del comprador en la transacción.
    - IDVendedor: El identificador del vendedor en la transacción.
    - Fecha: La fecha en que se produjo la transacción.
    - Monto: La cantidad de dinero intercambiada en la transacción.
- Métodos:
    - ProcesarPago(): Procesa el pago de la transacción.

Clase: Soporte

- Métodos:
    - CrearTicketSoporte(): Crea un ticket de soporte para problemas de usuarios.
    - ResolverProblema(): Atiende y resuelve el problema reportado.

iii. Diagrama de Paquetes y Componentes



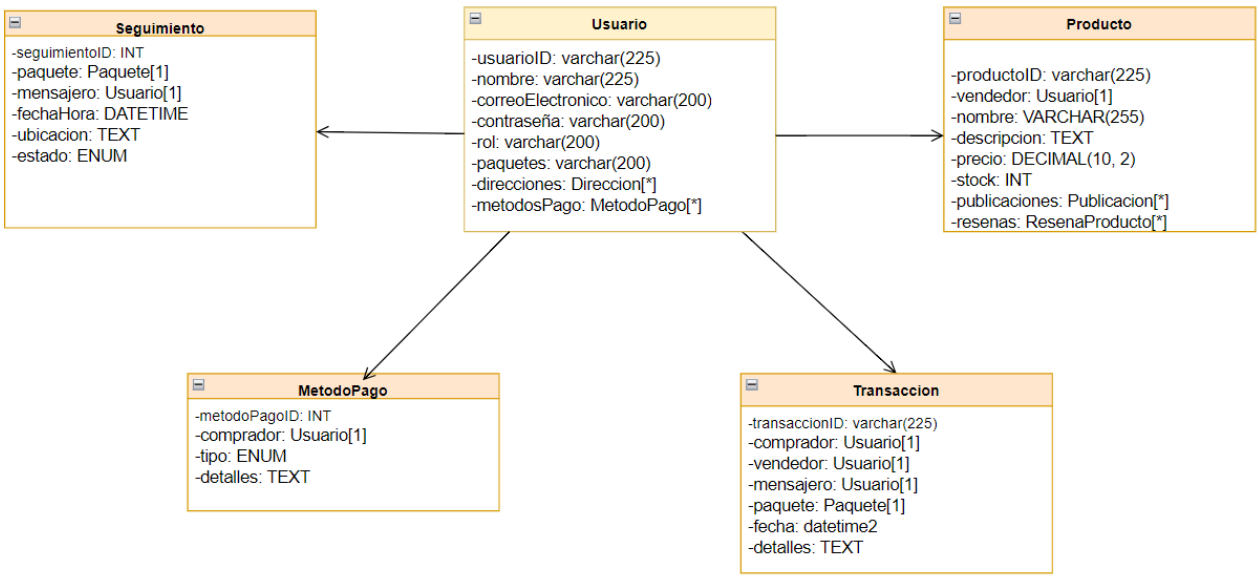
iv. Descripción de paquetes y componentes

- **Paquete Productos:** Este paquete contiene elementos que están relacionados con los productos dentro del sistema. Incluye el nombre del producto, descripción, fotografía, categoría y orden.

- **Paquete GUI:** Es el paquete de la Interfaz Gráfica de Usuario, con ventanas para categorías, órdenes, billetera, transacciones, recomendaciones y forma de pago.
- **Paquete DB:** El paquete de la base de datos, donde se maneja la persistencia y eliminación de datos.
- **Paquete Billetera:** Contiene elementos relacionados con el método de pago.
- **Paquete de Utilidad:** Este paquete incluye utilidades generales del sistema como comprador, vendedor y mensajero.

c. Persistencia

i. Diagrama de clases persistentes



ii. Tipo de almacenamiento

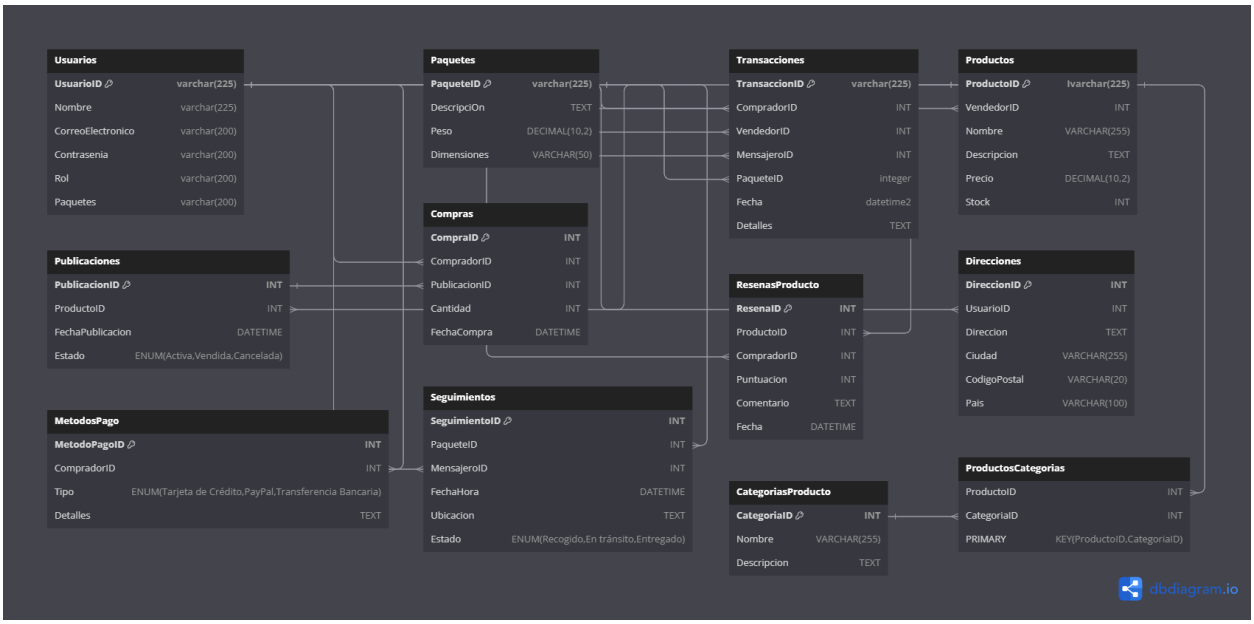
Debido al tipo de datos que se va a almacenar se escogió una base de datos relacional. Esto debido a que los datos tienen una estructura fija, donde representa que la estructura de datos es definida y ayuda con las relaciones entre las diferentes entidades. Por otro lado, las bases de datos

relacionales cuentan con mecanismos robustos para mantener la integridad de datos como lo son las claves primarias y foráneas; lo cual es indispensable ya que es necesario para que la consistencia de datos sea crucial. Donde por ejemplo no puede haber dos personas con el mismo dpi o numero de tarjeta para realizar algún tipo de transacción.

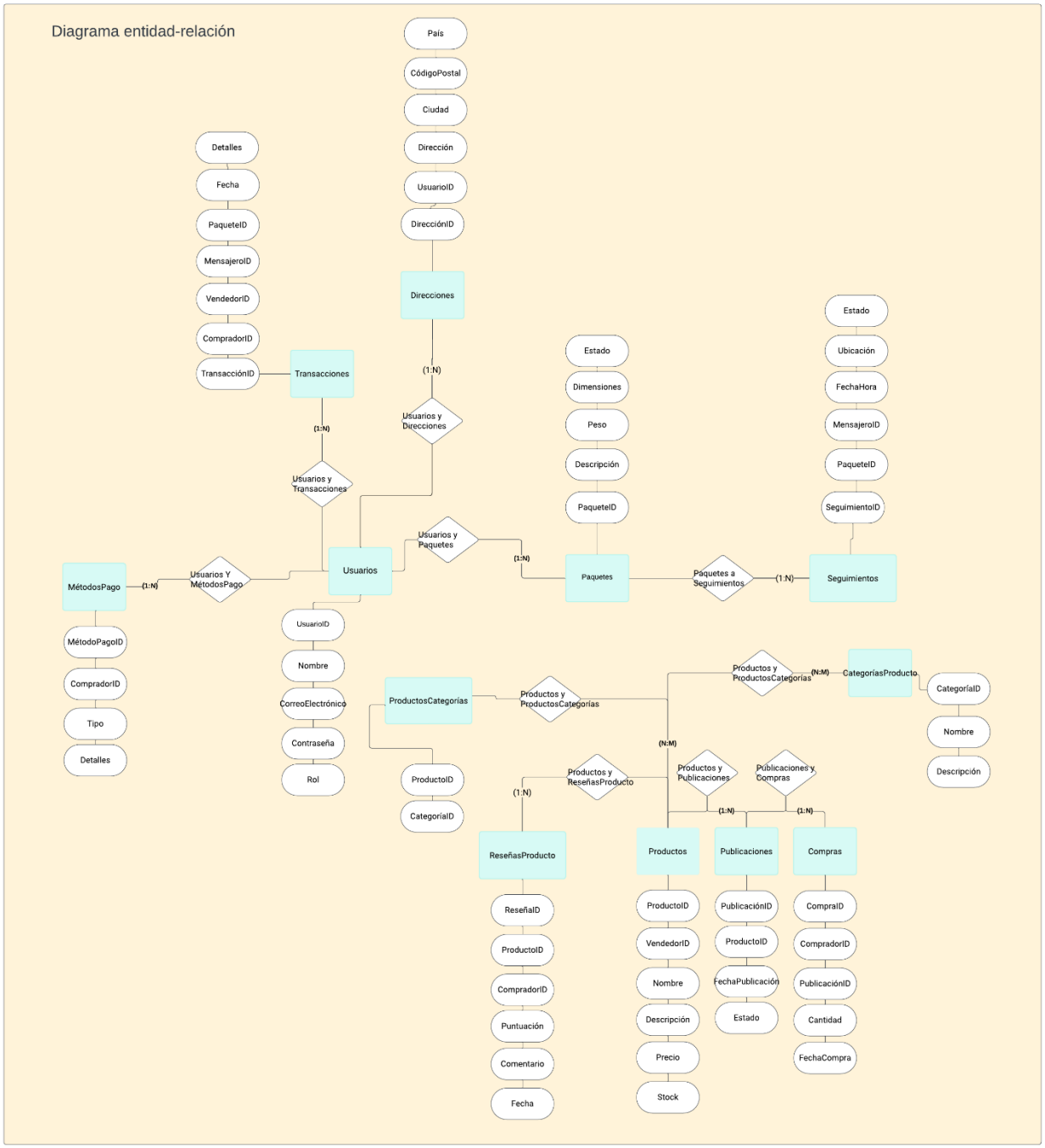
Otro punto a favor del almacenamiento de datos relacional es la recuperación de datos donde también es esencial debido a que si se va a manejar bastante comercio es necesario combinar la información de diferentes entidades como usuarios, el producto de compra, las compras, etc.

En general, Las bases de datos relacionales brindan soporte para transacciones ACID (Atomicidad, Consistencia, Aislamiento y Durabilidad), lo cual garantiza la integridad de los datos en caso de operaciones concurrentes o errores. Y por último las bases de datos relacionales, como MySQL, PostgreSQL o SQL Server, son tecnologías maduras y ampliamente adoptadas, con una gran comunidad de soporte y una vasta cantidad de recursos y herramientas disponibles.

iii. Diagrama entidad relación(Diccionario y Diagrama)



https://dbdiagram.io/d/IngSoftware1-65f3df0bae072629ce123559



https://lucid.app/lucidchart/411be89c-71bc-4aa5-bcf9-cc74f6bbdf00/edit?viewport_loc=4225%2C-2263%2C2958%2C1547%2C0_0&invitationId=inv_6e8ec0c1-aff5-4236-9ba4-1f9d5f63a7ce

Descripción de relaciones

1) Usuarios a Paquetes (1:N):
   - Un usuario (mensajero) puede entregar múltiples paquetes.

2) Usuarios a Transacciones*(1:N):
   - Un usuario (vendedor o comprador) puede tener múltiples transacciones.
3) Productos a Publicaciones* (1:N):
   - Un producto puede tener múltiples publicaciones a lo largo del tiempo
4) Publicaciones a Compras (1:N):
   - - Una publicación puede resultar en múltiples compras (en el caso de productos con múltiples unidades disponibles).
5) Usuarios a Direcciones (1:N):
   - Un usuario puede tener múltiples direcciones registradas.
6) Productos a ReseñasProducto*(1:N):
   - Un producto puede tener múltiples reseñas.
7) Productos a ProductosCategorías** (N:M):
   - Un producto puede pertenecer a múltiples categorías, y una categoría puede incluir múltiples productos.
8) Usuarios a MétodosPago (1:N):
   - Un usuario puede tener múltiples métodos de pago registrados.
9) Paquetes a Seguimientos (1:N):
   - Un paquete puede tener múltiples registros de seguimiento.

IV. Diseño

a. Estimaciones:

i. ***Capacidad de manejo de transacciones:*** Capacidad de procesar hasta 2,000 transacciones por segundo.

ii. ***Escalabilidad de Recursos:*** Automatización de escalado de recursos en menos de 5 minutos ante incrementos del 20% en la demanda.

iii. ***Satisfacción del usuario:*** Puntaje promedio de 4.5/5 en encuestas de satisfacción con usuarios.

iv. ***Tiempo promedio de completación de tareas comunes:*** Los usuarios nuevos completan tareas en menos de 2 minutos y medio.

v. ***Índice de error y recuperación:*** 95% de recuperación en los errores y menos del 5% del índice de errores en tareas comunes.

vi. ***Detección de transacciones sospechosas****:* 99% las transacciones sospechosas se identifican y bloquean automáticamente.

vii. ***Tiempo medio de detección de vulnerabilidades:*** Menos de 24 horas para detección de vulnerabilidades.

viii. ***Tiempo medio de respuesta a incidentes de seguridad:*** Respuesta y mitigación en menos de 4 horas.
ix. ***Tiempo de carga de páginas:*** Tiempo de carga promedio inferior a 2 segundos en condiciones normales y menos de 4 segundos bajo carga alta.
x. ***Procesamiento de transacciones:*** Tiempo de procesamiento y confirmación en mendos de 5 segundos.
xi. ***Cumplimiento de contraste de colores:*** 100% de cumplimiento en estándares de contraste de colores.
xii. ***Texto alternativo en elementos multimedia:*** 100% de contenido multimedia con texto alternativo adecuados.

b. Tecnologías

i. Tecnologías Consideradas:

1. Frontend:

a. React
b. React Native
c. Vue JS

2. Backend:

a. Laravel
b. Express
c. Node.js

3. DBM:

a. PostgreSQL
b. Mysql

4. Fullstack:

a. Meteor.js

c. Tecnologías Seleccionadas y su Análisis:

- **Lenguaje de programación:**

1. Javascript:

a. Ventajas:

- JavaScript es usado y soportado en la mayoría de navegadores modernos, permitiendo universalidad en el desarrollo.
- Cuenta con una vasta comunidad que proporciona recursos, bibliotecas y frameworks.
- Puede programarse entorno a objetos o a funcionalidades permitiendo flexibilidad.

b. Desventajas:

- Puede presentar inconsistencias en la interpretación en algunos navegadores.

        - Pueden existir fugas de memoria si no se manejan adecuadamente.

    c. **Razón:** En el desarrollo web moderno, Javascript es esencial para el desarrollo web, permite una transición fluida entre bacjend y frontend.

- **Frameworks de desarrollo:**
    1. **Meteor.js:**
        a. **Ventajas**
            - Presenta una integración completa de frontend y backend que permite desarrollar aplicaciones de manera rápida y reactiva en tiempo real.
            - Actualizaciones automáticas de cambios en la base de datos, Websockets integrados.
            - Comunidad activa y propio repositorio de paquetes.
            - Desarrollo multiplataforma a través de cordova.
        b. **Desventajas:**
            - Su popularidad es menor en comparación con frameworks específicos para backend y frontend.
        c. **Razón:** Al ofrecer un entorno de desarrollo completo facilita la creación de aplicaciones en tiempo real y sincronización multiplataforma.

    2. **React:**
        a. **Ventajas:**
            - Componentes reutilizables, alto rendimiento con virtual DOM, soporte comunitario.
        b. **Desventajas:**
            - Cubre la capa de la vista y requiere la integración con otras bibliotecas para el enrutamiento y la gestión del estado.
        c. **Razón:** Facilita el desarrollo de frontend dinámico y responsive, lo que complementa la arquitectura en tiempo real de Meteor.js

    3. **Node.js**
        a. **Ventajas:**
            - El rendimiento asincrónico soporta operaciones E/S unificado bajo javascript para todo el Stack.
        b. **Desventajas:**
            - Al tener una naturaleza de un solo hilo presenta dificultades para tareas de cpu intensivas.

c. **Razón:** Integración natural de Meteor.js y permite transiciones entre el cliente y servidor dentro de un mismo lenguaje.

- **Transpilador:**
  1. **Babel**
     a. **Ventajas:**
        - Permite el uso de Javascript moderno y garantiza compatibilidad con navegadores antiguos.
        - Es altamente configurable con plugins y presets para adaptarse a necesidades específicas.
     b. **Desventajas:**
        - La transpilación puede aumentar el tiempo de desarrollo.
        - Requiere configuración y mantenimiento para gestión de dependencias y plugins.
     c. **Razón:** Es fundamental para escribir código JS moderno y compatible con entornos de ejecución.

- **BDD:**
  1. **PostrgeSQL:**
     a. **Ventajas:**
        - Alto rendimiento, robustez y compatibilidad con SQL.
        - Soporta funciones avanzadas como transacciones, subqueries, triggers y procedimientos almacenados.
     b. **Desventajas:**
        - Complejidad en administración.
     c. **Razón:** PostgreSQL y sus capacidades avanzadas y escabilidad apoyan las necesidades para apoyar una aplicación en tiempo real y data-intensive que Meteor.js soporta.

- **DevOps:**
  1. **Docker:**
     a. **Ventajas:**
        - Facilidad en creación, despliegue y ejecución de aplicaciones en contenedores, asegurando la consistencia del entorno.

     b. **Desventajas:**
        - Si se presentan múltiples contenedores, la gestión de estos puede ser compleja.
     c. **Razón:**
        - Es un entorno de desarrollo y despliegue portable y consistente que reduce complicaciones entre entornos.

2. **Github:**
    a. **Ventajas:**
        - Colaboración que integra control de versiones con funcionalidades de revisión de código y gestión de proyectos.
    b. **Desventajas:**
        - Dependencia de una plataforma externa.
    c. **Razón:**
        - Por la capacidad y funcionalidades de gestión se ha vuelto imprescindible y un estándar de la industria.

Anexos

Informe de Gestión

- Desglose Del proyecto
    - Preparación de Documentos
        - Resumen
        - Introducción
    - Design Thinking
        - Prototipos
        - Elaborar versiones del prototipo
        - Documentar los cambios.
    - Análisis
        - Requisitos Funcionales
        - Backend
            - Creación de diagrama de clases
            - Describir las clases con sus paquetes
            - Persistencia de datos
            - Elaborar diagrama de clases persistentes.
            - Seleccionar el tipo de almacenamiento de datos y justificación.
            - Diseño de diagrama Entidad-Relación y alternativas según el tipo de almacenamiento elegido.
        - Diseño
            - Selección de tecnología

Tareas por integrante

Tarea	Integrante	Observaciones
Hoja de Presentación	Todos	Creación del documento y hoja de presentación.
Resumen	Todos	Síntesis clara y concisa del proyecto.
Introducción	Todos	Descripción completa y objetivos.
Prototipos	Gustavo Cruz, Carlos Valladares	Diseño y pruebas con usuarios, análisis de feedback.
Requisitos Funcionales y Análisis	Rodrigo Mansilla	Identificación precisa de requisitos, análisis detallado.
Backend y Persistencia de Datos	Paula Barillas, Brandon Reyes	Diseño técnico y selección de tecnologías de almacenamiento.

Selección de Tecnología	Rodrigo Mansilla	Investigación y justificación de las tecnologías elegidas.
Informe de Gestión y Desglose de Tareas	Rodrigo Mansilla	Organización del trabajo y seguimiento del progreso.
Diagrama Entidad-Relación	Paula Barillas	Diseño de la base de datos y relaciones.
Preparación de Documentación Final	Todos los miembros	Revisión final y ajustes.
Repositorio de GitHub	Todos	Configuración, manejo del repositorio y documentación.

Enlace de formulario LOGT:

https://docs.google.com/spreadsheets/d/1aKTDuX9Zo0OR0LNzrWuVDZNhRheaq6deXiGLJOcPQVA/edit?usp=sharing

Enlace de Repositorio:

https://github.com/paulabaal12/PROY-SOFTWARE