Manual técnico Bitevia software.

Juan Esteban caballero Goenaga.
Santiago David zambrano izaquita.
Andrw Stiven barrera poveda
Wilber robles mercado.
Miguel Sarabia soto

SENA-Servicio nacional de aprendizaje.
Adso-Análisis y desarrollo de software
Ficha: 2928707

Ing. Edwin Suarez Palacio.
Ing. Henry Guzmán Escorcia.

Barranquilla-Atlántico.
Diciembre 2025

Contenido

Manual técnico.

En este documento va dirigido al soporte técnico de una organización a quien haga uso del software, en el cual se da a detalle sobre la estructura del sistema Y dar explicación de una manera detallada lo que hemos creado.

Se presenta una explicación del software desarrollado paso a paso de lo creado Para una mejor compresión herramientas implementadas, modelos y controladores en trabajo del sistema.

Componentes

Algunos de los conceptos tratados en este manual se dan a entender en este manual.

Alcance

El software fue desarrollado con el fin de sistematizar los procesos de ventas En el local de Comidas rápidas "La Terraza del Pri", ubicado en la Calle 56 #34-6, barrio Ciudadela Metropolitana, Soledad, Atlántico.

El software ofrece una interfaz de usuario optimizada (lado del cliente) que incluye las siguientes funcionalidades clave:
Exploración de Productos: Un menú interactivo permite a los usuarios visualizar la oferta completa del establecimiento, organizada por categorías claras (ej. Grupo Salchipapa, Grupo Hamburguesas), facilitando la navegación y la selección.
Gestión de Pedidos: Implementa una funcionalidad de carrito de compras eficiente para la fácil realización y gestión de pedidos.
Soporte Multilingüe: Para garantizar la accesibilidad, la aplicación es compatible con dos idiomas, incluyendo el inglés, para atender a clientes no hispanohablantes.

Panel del Administrador (Admin)

El sistema ofrece un Panel de Administración completo con las siguientes herramientas clave para la gestión y optimización del negocio:
Análisis y Reportes: Proporciona un módulo de estadísticas para el seguimiento del rendimiento, incluyendo métricas críticas como el producto más vendido y el desempeño de las categorías de productos.
Gestión de Inventario y Menú: Permite la administración detallada de ingredientes, la modificación de grupos o categorías de productos, y la actualización de contenido visual mediante la modificación de *banners* promocionales.

Interfaz de Cocina (Chef/Cocinero)

Esta interfaz está diseñada para optimizar el flujo de trabajo en la cocina y garantizar la eficiencia en el cumplimiento de pedidos:
Visualización de Pedidos: El personal de cocina puede visualizar en tiempo real los pedidos entrantes a medida que se generan.
Gestión de Estados del Pedido: Permite la actualización del progreso de cada orden mediante la modificación de los siguientes estados predefinidos:
En Preparación: Proceso iniciado.
Terminado: Listo para ser recogido/entregado.

En Tránsito (o Llevado): Si aplica la entrega interna o al *delivery*.
Entregado: Proceso de cumplimiento finalizado.

Descripción de los procesos.

Interfaz de Usuario y Navegación (Front-End)

Accesibilidad Universal: La aplicación garantiza el acceso a través de un enfoque agnóstico al navegador.
Visualización de Inventario: Permite a los usuarios interactuar con el catálogo de productos, organizado en categorías o grupos de comida.
Gestión de Transacciones: Implementa un sistema de carrito de compras robusto que facilita la personalización de pedidos. Los usuarios pueden añadir, modificar o eliminar ítems de forma dinámica antes de la confirmación final.
Soporte Multilingüe: El sistema ofrece funcionalidad de cambio de idioma (localización), soportando el inglés como idioma secundario mediante la implementación de los mecanismos de código necesarios.

2. Arquitectura de Datos y Transaccional (Back-End)

Procesamiento de Pedidos: Las solicitudes generadas en el Front-End son procesadas por el Back-End, asegurando la integridad de los datos transaccionales y la persistencia de los pedidos en la base de datos.
Gestión de Cuentas: Se incluye un módulo de creación de cuentas obligatorio para la formalización de pagos. Este módulo garantiza la captura y el almacenamiento seguro de los datos del cliente en la base de datos para futuras transacciones.

Diagramas.

En este espacio se mostrará los diagramas más relevantes del aplicativo.

Diagrama de clase:

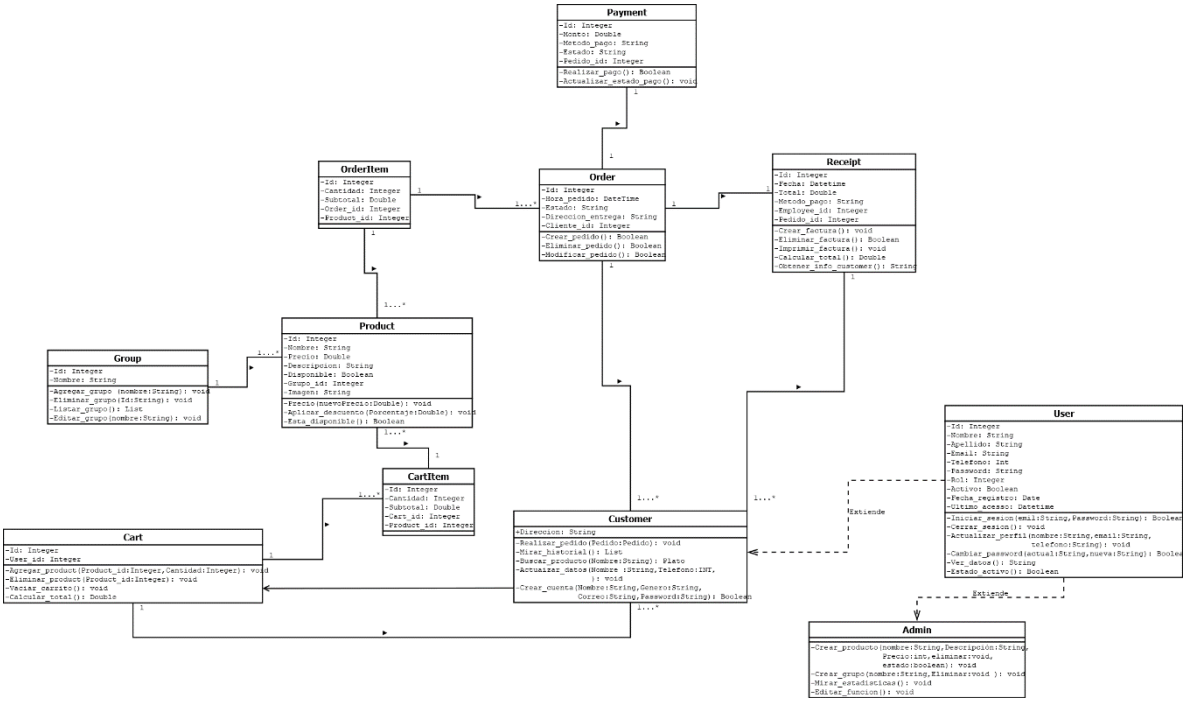
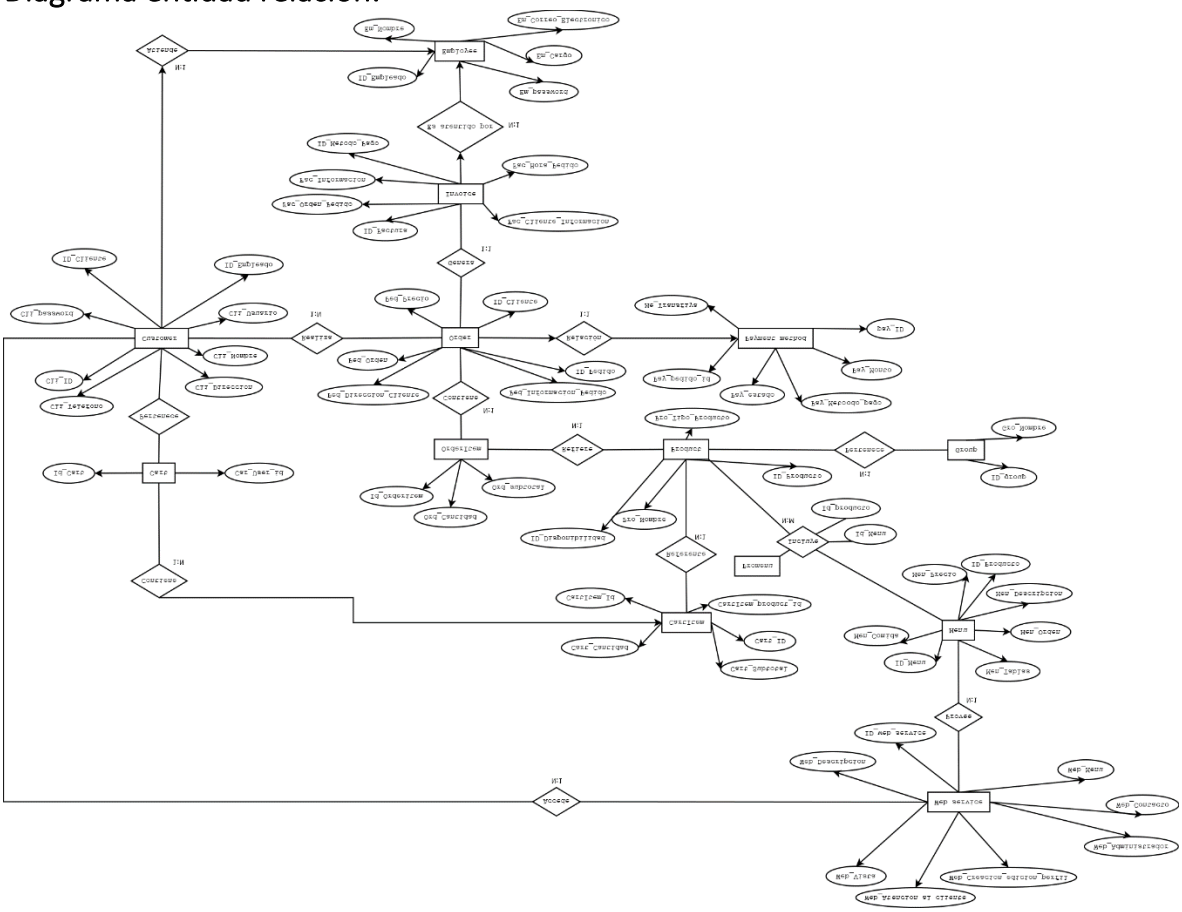


Diagrama entidad relación:



Mapa de navegación:

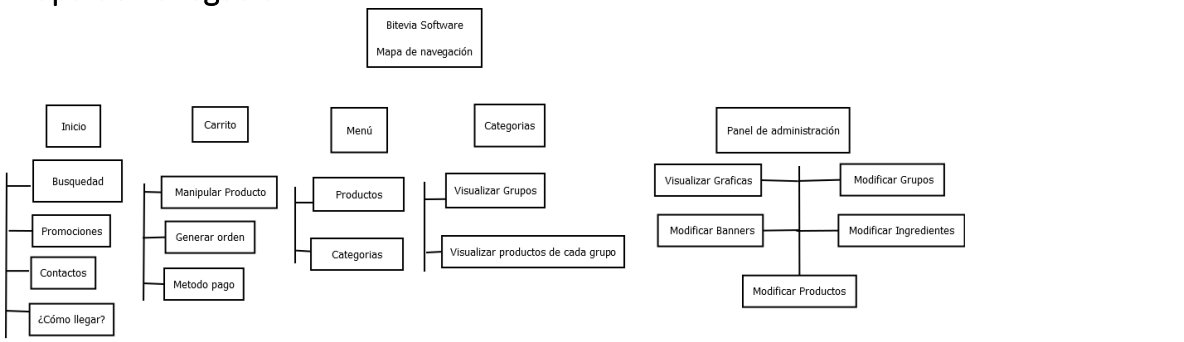
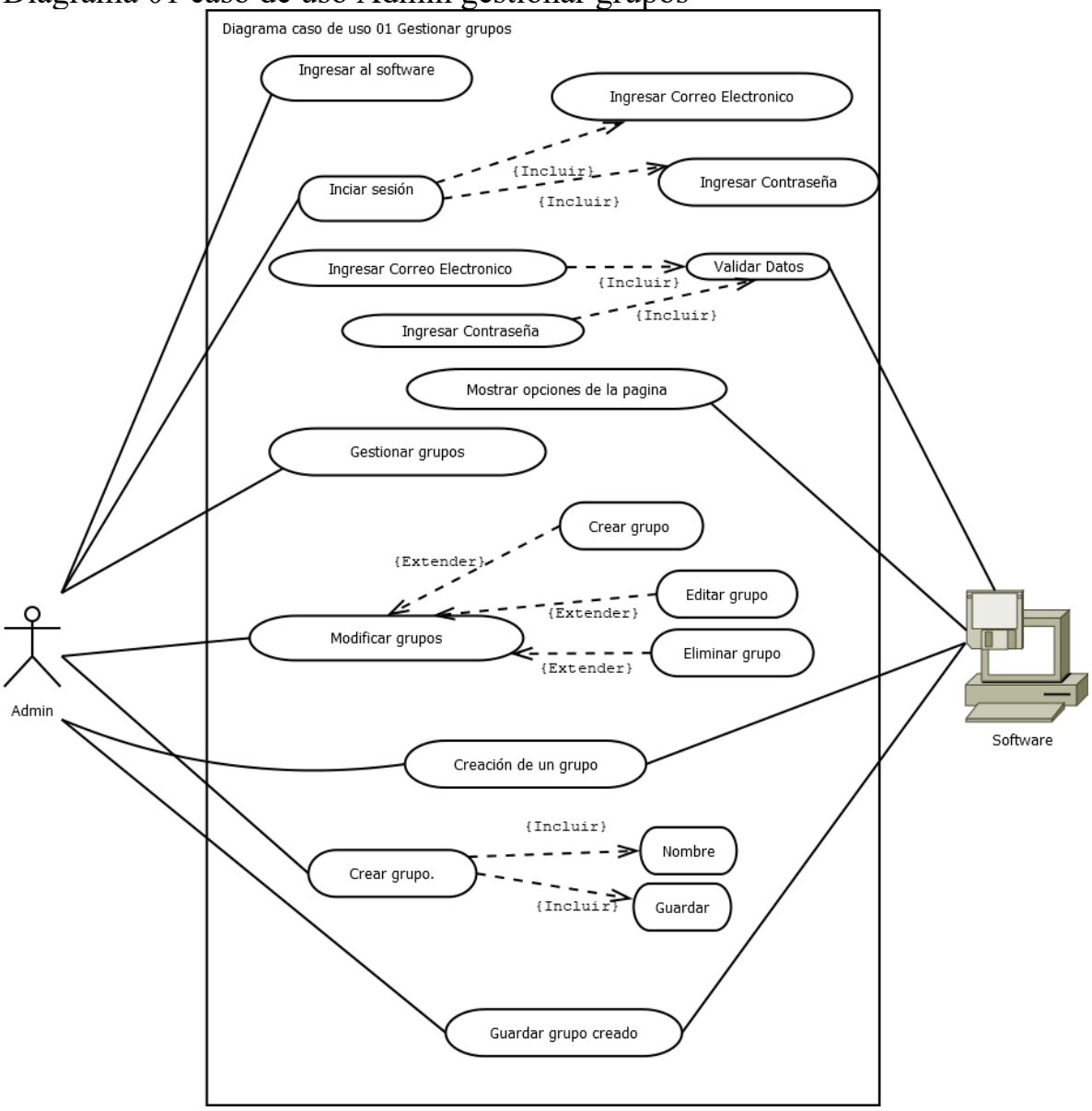


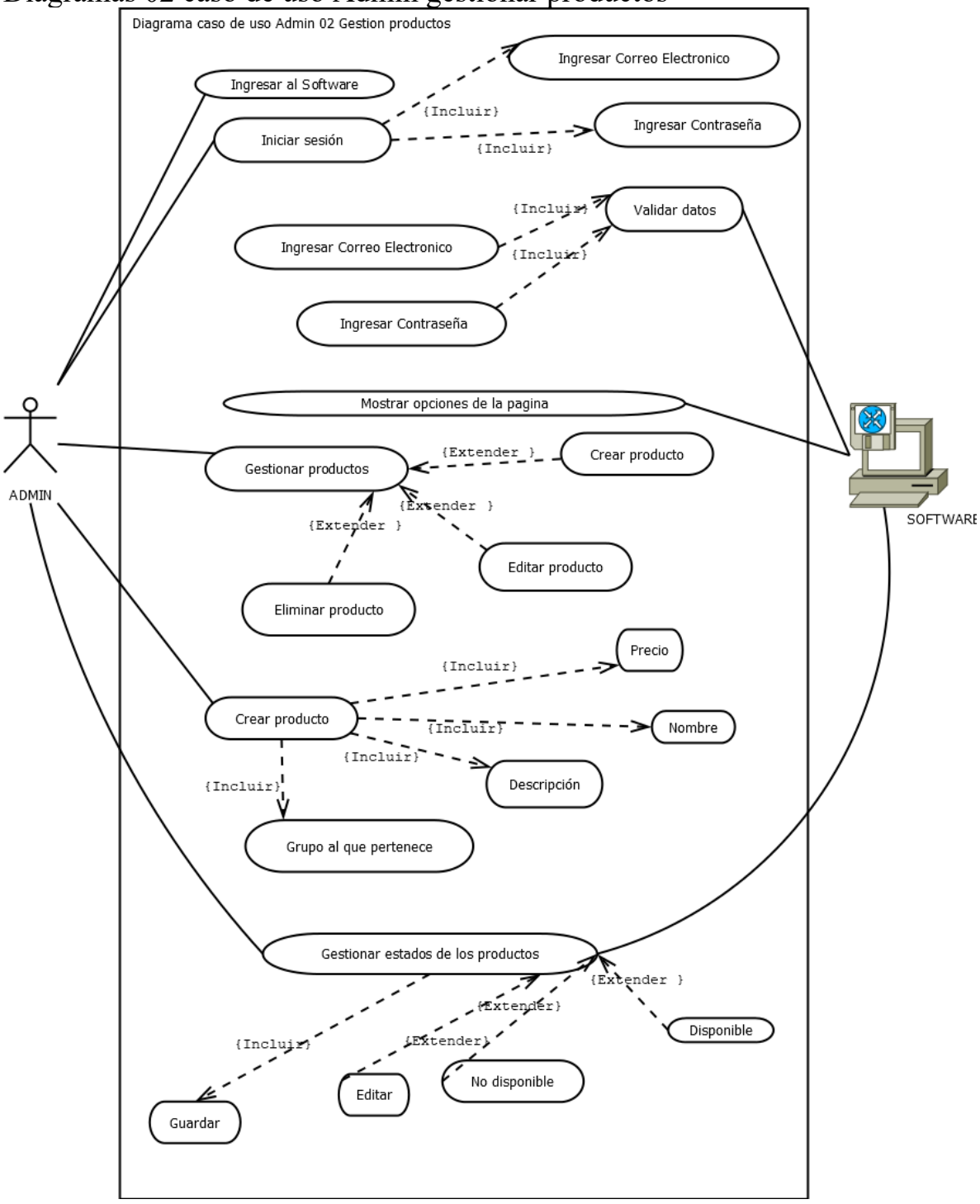
Diagrama 01 caso de uso Admin gestionar grupos



Ficha caso de uso 01 ADMIN Gestionar Grupos

Nombre: Ficha caso de uso ADMIN 01 Gestionar Grupos
Autor: Juan Caballero, Santiago Zambrano, Andrw Barrera, Miguel Sarabia y Wilber Robles
Actor: Administrador
Precondiciónal
1. El Administrador debe de estar registrado en el software.
2. El Administrador debe de saberse sus credenciales.
3. El Administrador debe de tener una computadora.
Flujo normal
1. El Administrador ingresa al software.
2. El Administrador ingresa sus datos de inicio Correo Electronico y contraseña.
3. El Software valida los datos Correo electronico y contraseña.
4. El Software muestra las diferentes opciones.
5. El Administrador Selecciona Visualizar Grupos.
6. El Software muestra las diferentes funciones que realiza en los grupos Crear grupo, Editar grupo, Eliminar grupo
7. El Administrador cierra el software .
Flujo alternativo
2.1 El administrador no recuerda sus credenciales
3.1 El Software no valida los datos del administrador
3.1 El Software no muestra las diferentes opciones en el sistema
5. Perdida de luz del Administrador

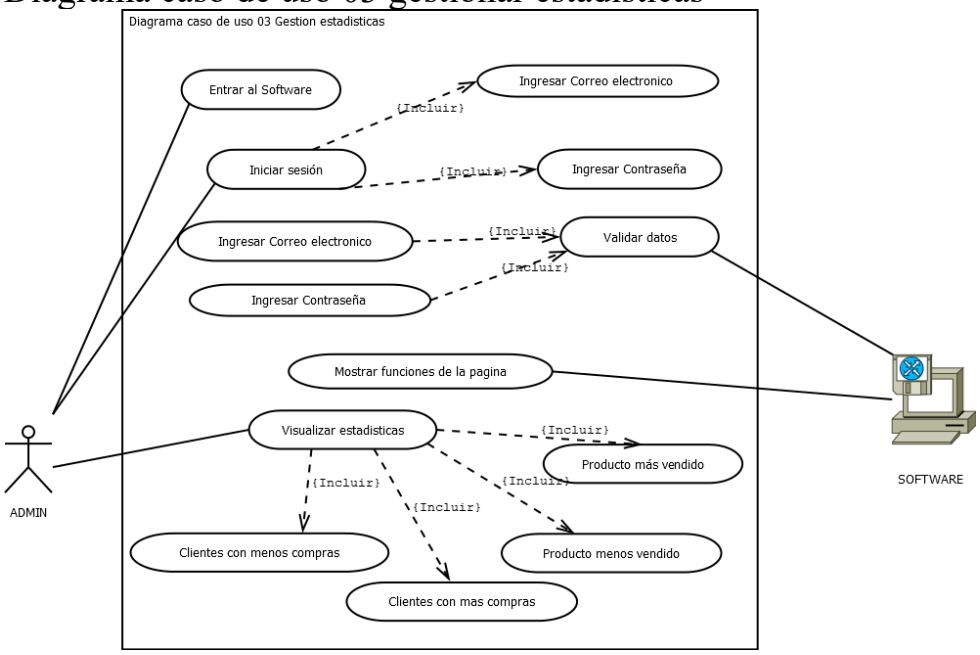
Diagramas 02 caso de uso Admin gestionar productos



Ficha caso de uso 02 Gestionar productos

Nombre: Ficha caso de uso ADMIN 02 Gestionar productos
Autor: Juan Caballero, Santiago Zambrano, Andrw Barrera, Miguel Sarabia y Wilber Robles
Actor: Administrador
Precondiciónal
1. El Administrador debe de estar registrado en el software.
2. El Administrador debe de saberse sus credenciales.
3. El Administrador debe de tener una computadora.
4. El software debe de estar funcionando correctamente.
Flujo normal
1. El Administrador ingresa al software.
2. El Administrador ingresa sus datos de inicio Correo Electronico y contraseña.
3. El Software valida los datos Correo electronico y contraseña.
4. El Software muestra las diferentes opciones.
5. El Administrador selecciona gestionar productos.
6. El Software muestra las opciones de uso del producto. Crear producto, editar producto, eliminar producto
7. El Administrador seleccionar crear producto.
8. El software los campos a completar de la creación del producto. Nombre, precio, descripción, grupo al que pertenece.
9. El Administrador guarda el nuevo producto.
10. El software se encarga del guardado del nuevo producto.
Flujo alternativo
2.1 El Administrador no se acuerde de sus credenciales.
3.1 El Software no responda para verificar los datos.
4.1 El Software no muestra las diferentes opciones del sistema
10.1 El Software no guarde el nuevo producto guardado

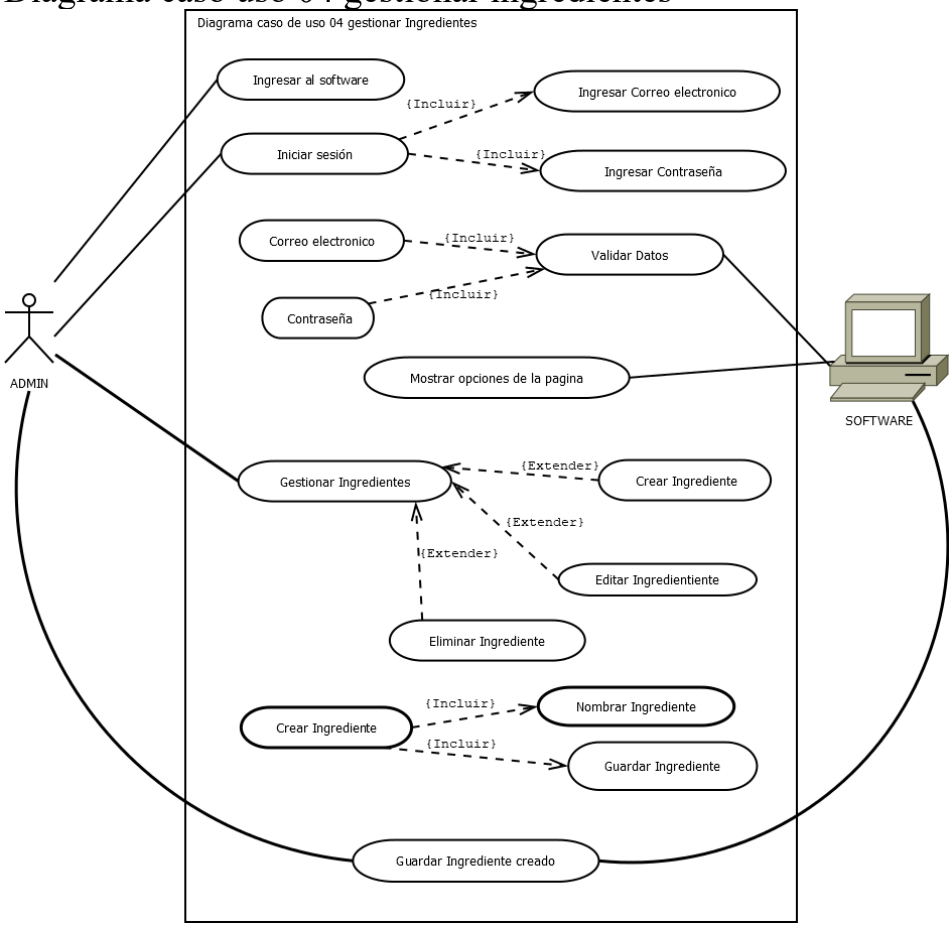
Diagrama caso de uso 03 gestionar estadísticas



Ficha caso de uso 03 gestionar estadísticas

Ficha Caso de uso
Nombre: Ficha caso de uso ADMIN 03 gestion estadisticas
Autor: Juan Caballero, Santiago Zambrano, Andrw Barrera, Miguel Sarabia y Wilber Robles
Actor: Administrador
Precondiciónal
1. El Administrador debe de estar registrado en el software.
2. El Administrador debe de saberse sus credenciales.
3. El Administrador debe de tener una computadora.
Flujo normal
1. El Administrador ingresa al software.
2. El Administrador ingresa sus datos de inicio Correo Electronico y contraseña.
3. El Software valida los datos Correo electronico y contraseña.
4. El Software muestra las diferentes opciones.
5. El Administrador Selecciona visualizar las estadisticas.
6. El Software muestra las diferentes tablas de Producto mas vendido, producto menos vendido, clientes con mas compras y clientes con menos compras.
7. El Administrador cierra el software .
Flujo alternativo
2.1 El Administrador no se acuerde de sus credenciales.
3.1 El Software no responda para verificar los datos.
3.1 Repentina perdida de señal del Software con el Administrador.
4.1 El Software no logra renderizar las tablas.

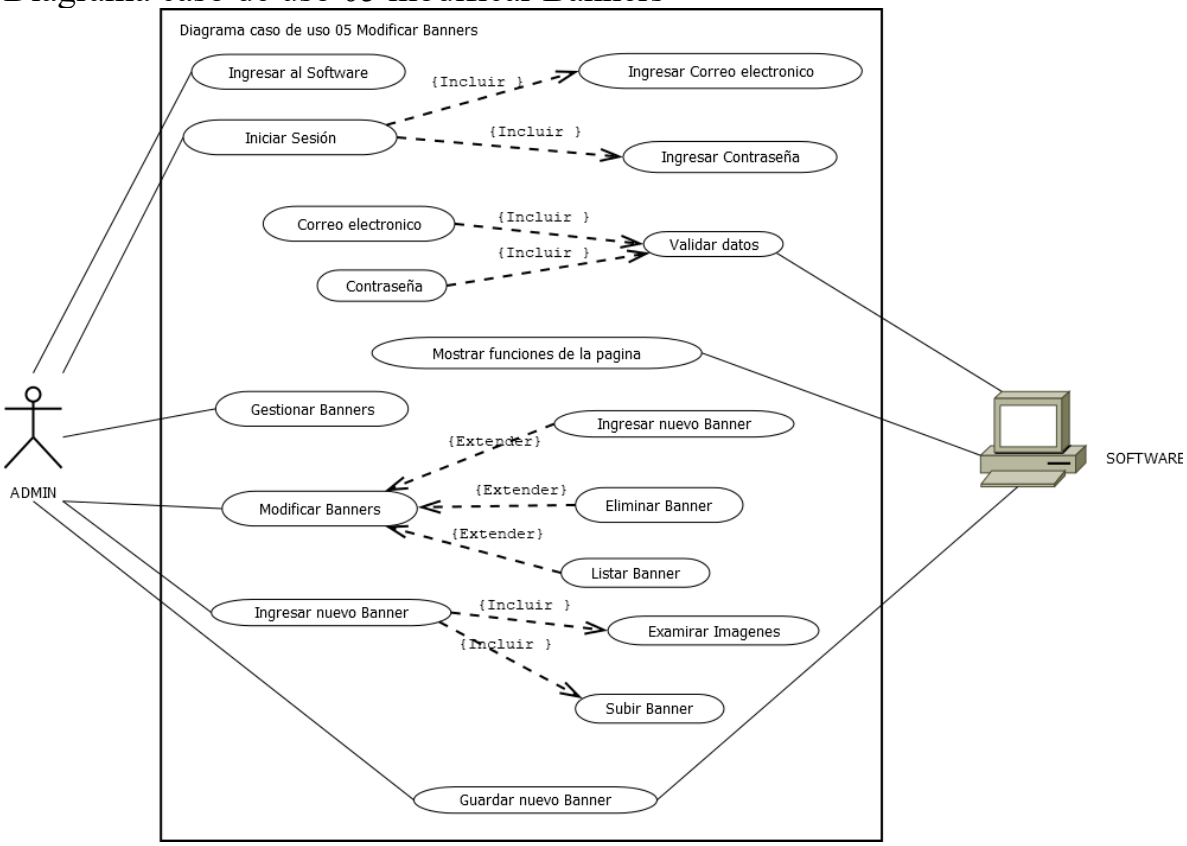
Diagrama caso uso 04 gestionar ingredientes



Ficha caso de uso 04 gestionar ingredientes

Nombre: Ficha caso de uso ADMIN 04 gestionar ingredientes
Autor: Juan Caballero, Santiago Zambrano, Andrw Barrera, Miguel Sarabia y Wilber Robles
Actor: Administrador
Precondiciónal
1. El Administrador debe de estar registrado en el software.
2. El Administrador debe de saberse sus credenciales.
3. El Administrador debe de tener una computadora.
4. El software debe de estar funcionando
Flujo normal
1. El Administrador ingresa al software.
2. El Administrador ingresa sus datos de inicio Correo Electronico y contraseña.
3. El Software valida los datos Correo electronico y contraseña.
4. El Software muestra las diferentes opciones.
5. El Administrador Selecciona gestionar ingredientes.
6. El Software muestra las opciones de uso de los ingredientes. Crear ingrediente, editar ingrediente, eliminar ingrediente
7. El Administrador selecciona crear ingrediente.
8. El software muestra los campos a llenar para crear un ingrediente. Nombre
9. El Administrador guarda el nuevo ingrediente.
10. El software se encarga del guardado del nuevo ingrediente
Flujo alternativo
1.1 El Administrador no se acuerde de sus credenciales.
2.1 El Software no responda para verificar los datos.
4.1 El Software no muestra las vistas seleccionada por el Administrador
5.1 El Software no guarda el nuevo ingrediente creado

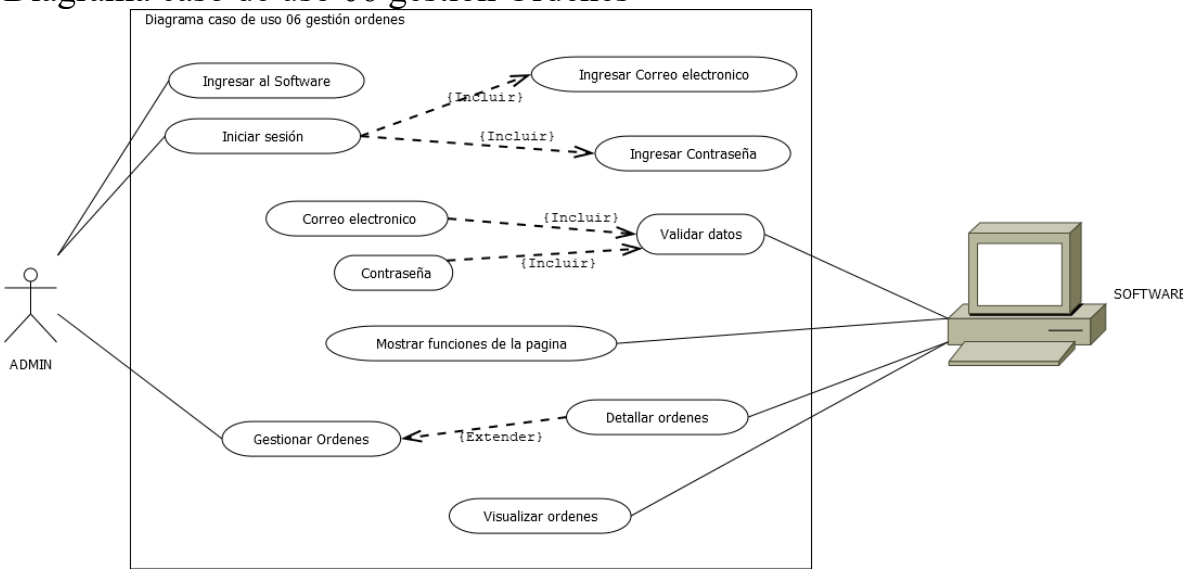
Diagrama caso de uso 05 modificar Banners



Ficha caso de uso 05 modificar banners

Nombre: Ficha caso de uso ADMIN 05 Gestionar banners
Autor: Juan Caballero, Santiago Zambrano, Andrw Barrera, Miguel Sarabia y Wilber Robles
Actor: Administrador
Precondiciónal
1. El Administrador debe de estar registrado en el software.
2. El Administrador debe de saberse sus credenciales.
3. El Administrador debe de tener una computadora.
4. El software debe de estar funcionando
Flujo normal
1. El Administrador ingresa al software.
2. El Administrador ingresa sus datos de inicio Correo Electronico y contraseña.
3. El Software valida los datos Correo electronico y contraseña.
4. El Software muestra las diferentes opciones.
5. El Administrador Selecciona gestionar Banners.
6. El Software muestra las opciones de Banner. Nuevo Banner, listar Banners y Eliminar Banner
7. El Administrador Ingresa un nuevo Banner.
8. El software permite seleccionar un nueva imagen para Banner
9. El Administrador selecciona Imagen y la guarda.
10. El software se encarga del guardado de la imagen y la renderización de la imagen en el Home de la pagina web.
Flujo alternativo
2.1 El administrador no recuerda sus datos
3.1 El Sofware no valida los credenciales del administrador
4.1 El Software no muestra las vistas seleccionada por el Administrador
10.1 El Software no realiza el guardado de los banners y no renderiza en la vista principal

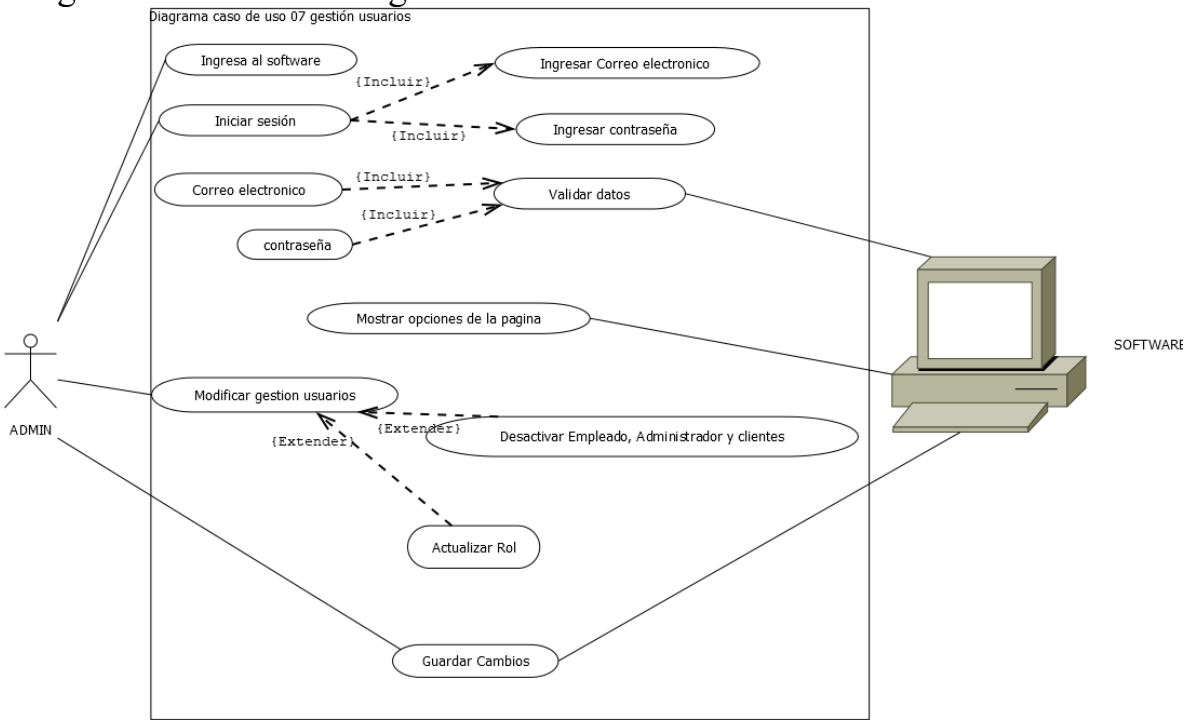
Diagrama caso de uso 06 gestión Ordenes



Ficha caso de uso 06 gestionar ordenes

Nombre: Ficha caso de uso ADMIN 06 Gestionar Ordenes
Autor: Juan Caballero, Santiago Zambrano, Andrw Barrera, Miguel Sarabia y Wilber Robles
Actor: Administrador
Precondiciónal
1. El Administrador debe de estar registrado en el software.
2. El Administrador debe de saberse sus credenciales.
3. El Administrador debe de tener una computadora.
4. El software debe de estar funcionando
Flujo normal
1. El Administrador ingresa al software.
2. El Administrador ingresa sus datos de inicio Correo Electronico y contraseña.
3. El Software valida los datos Correo electronico y contraseña.
4. El Software muestra las diferentes opciones.
5. El Administrador Selecciona gestionar ordenes
6. El Software muestra las opciones de la vista ordenes. Visualizar ordenes en el sistema.
7. El Administrador supervisa las ordenes actuales en el Software.
Flujo alternativo
2.1 El administrador no recuerda sus credenciales
3.1 El software falla al validar los datos
4.1 El Software no muestra las vistas seleccionada por el Administrador
7.1 Falla al cargar las ordenes actuales en el sistema

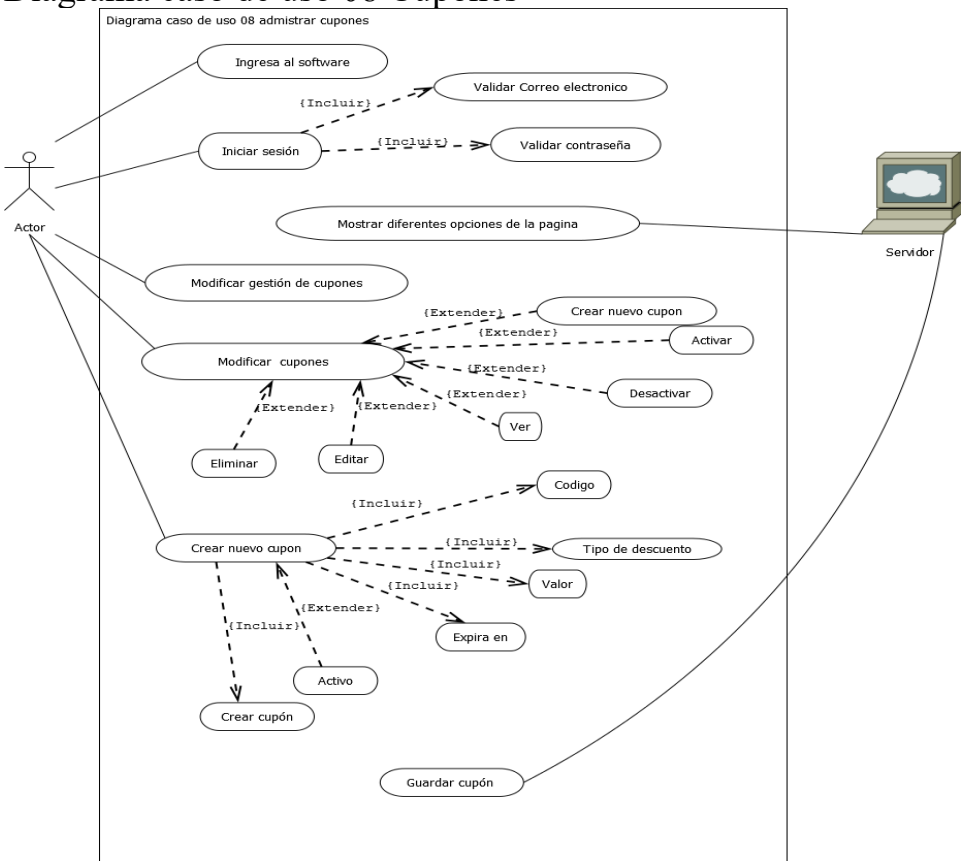
Diagrama caso de uso 07 gestión usuarios



Ficha caso de uso 07 gestionar usuarios

Nombre: Ficha caso de uso ADMIN 07 Gestionar Usuarios
Autor: Juan Caballero, Santiago Zambrano, Andrw Barrera, Miguel Sarabia y Wilber Robles
Actor: Administrador
Precondiciónal
1. El Administrador debe de estar registrado en el software.
2. El Administrador debe de saberse sus credenciales.
3. El Administrador debe de tener una computadora.
4. El software debe de estar funcionando
Flujo normal
1. El Administrador ingresa al software.
2. El Administrador ingresa sus datos de inicio Correo Electronico y contraseña.
3. El Software valida los datos Correo electronico y contraseña.
4. El Software muestra las diferentes opciones.
5. El Administrador Selecciona gestionar Usuarios
6. El Software muestra las opciones de la vista de usuarios. Buscar usuarios, actualizar rol y desactivar usuarios (Clientes, Administradores y Empleados)
7. El Administrador puede hacer una actualización a un Usuario a su antojo.
Flujo alternativo
1.1 El Administrador no se acuerde de sus credenciales.
3.1 El Software no responda para verificar los datos.
4.1 El Software no muestra las vistas seleccionada por el Administrador
7.1 El software no realiza las funciones realizadas por el administrador

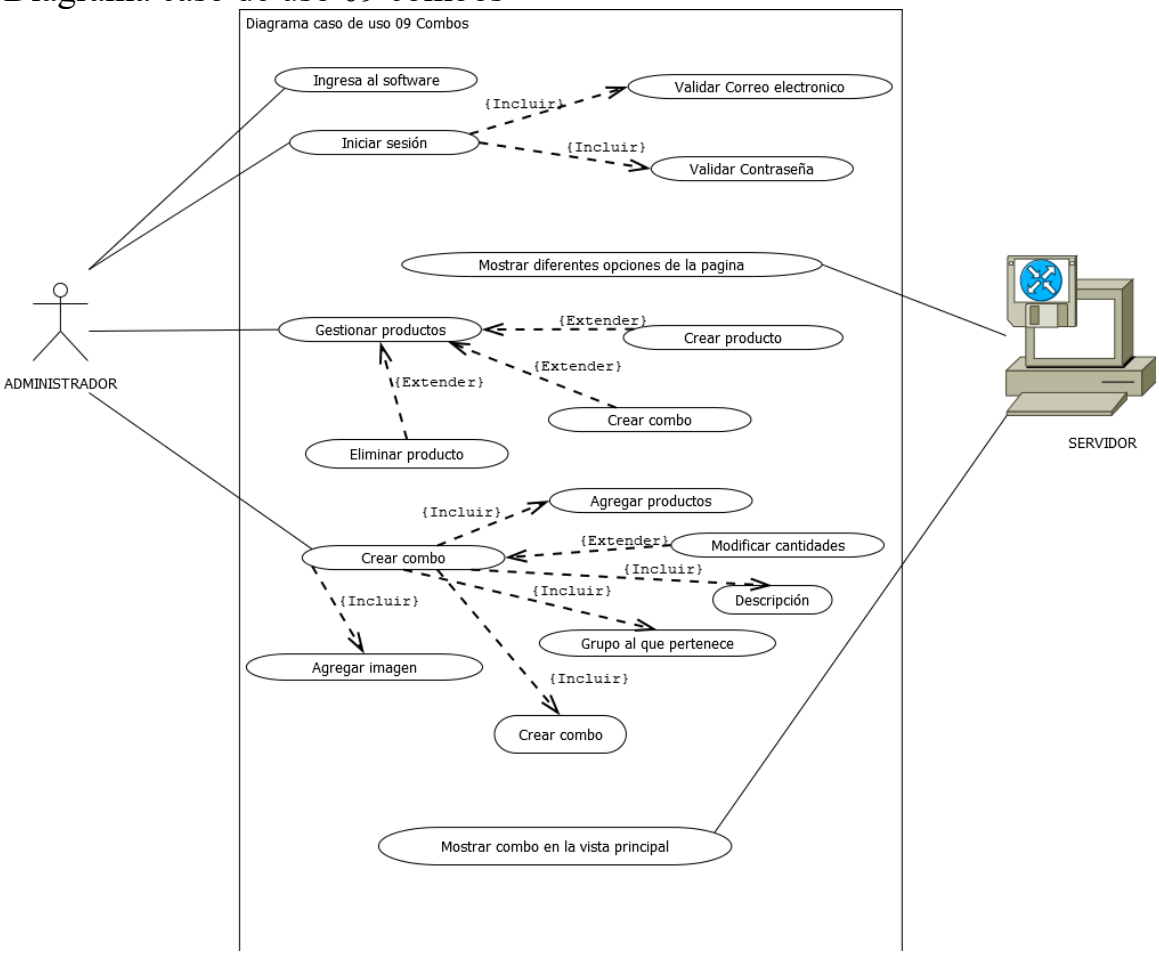
Diagrama caso de uso 08 Cupones



Ficha caso de uso 08 gestionar cupones

Nombre: Ficha caso de uso ADMIN 08 Gestionar Cupones
Autor: Juan Caballero, Santiago Zambrano, Andrw Barrera, Miguel Sarabia y Wilber Robles
Actor: Administrador
Precondicióna l
1. El Administrador debe de estar registrado en el software.
2. El Administrador debe de saberse sus credenciales.
3. El Administrador debe de tener una computadora.
4. El software debe de estar funcionando
Flujo normal
1. El Administrador ingresa al software.
2. El Administrador ingresa sus datos de inicio Correo Electronico y contraseña.
3. El Software valida los datos Correo electronico y contraseña.
4. El Software muestra las diferentes opciones.
5. El Administrador Selecciona gestionar cupones
6. El Software muestra las opciones de la vista de cupones(Crear cupon, activar, desactivar, ver , editar y eliminar)
7. El Administrador selecciona crear cupon
8. El software muestra campos a llenar (Codigo, tipo de descuento, valor, expira en y activar)
9. El Administrador crea el nuevo cupón
10. El software guarda el nuevo cupon.
Flujo alternativo
1.1 Caida del software
2.1 El Software no responda para verificar los datos.
4.1 El Software no muestra las vistas seleccionada por el Administrador
10.1 El Software no guarda el nuevo cupón creado

Diagrama caso de uso 09 combos

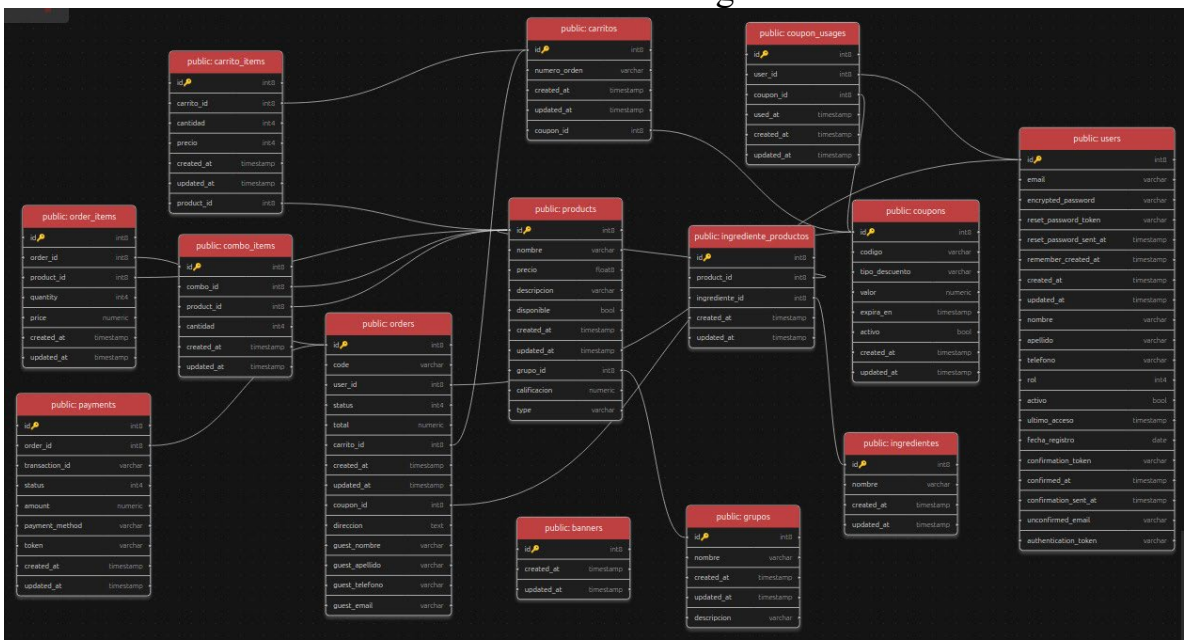


Ficha caso de uso 09 gestionar combos

Nombre: Ficha caso de uso ADMIN 09 Gestionar Combo
Autor: Juan Caballero, Santiago Zambrano, Andrw Barrera, Miguel Sarabia y Wilber Robles
Actor: Administrador
Precondiciónal
1. El Administrador debe de estar registrado en el software.
2. El Administrador debe de saberse sus credenciales.
3. El Administrador debe de tener una computadora.
4. El software debe de estar funcionando.
Flujo normal
1. El Administrador ingresa al software.
2. El Administrador ingresa sus datos de inicio Correo Electronico y contraseña.
3. El Software valida los datos Correo electronico y contraseña.
4. El Software muestra las diferentes opciones.
5. El Administrador Selecciona gestionar productos(Crear producto, eliminar producto y crear combo)
6. El Software muestra las campos a llenar para crear un combo(Agregar productos, modificar cantidades, eliminar productos, descripción, grupo que pertenece y agregar imagen)
7. El Administrador llena los campos y guarda el combo.
8. El software guarda el combo y lo renderiza en la vista principal.
Flujo alternativo
1.1 El software no carga
2.1 El administrador no recuerda sus credenciales
3.1 El software no valida los credenciales del administrador
4.1 El Software no muestra las vistas seleccionada por el Administrador
8.1 El Software no guarda el combo creado por el administrador

Modelo relacional en la base de datos.

El modelo relacional de la base de datos es el siguiente.



Diccionario de datos

Tabla public_users

Campo	Tipo Datos	Descripción
Id	Integer(Pk)	Identificador de usuario
Email	Varchar	Dirección de correo electronico
encrypted_password	Varchar	Contraseña cifrada del usuario
reset_password_token	Varchar	Token de seguridad para restablecer la contraseña
reset_password_sent_at	Timestamp	Fecha y hora de envio de token
remember_created_at	Timestamp	Fecha y hora de creación para recordar al usuario
created_at	Timestamp	Fecha y hora de la creación del usuario
updated_at	Timestamp	Fecha y hora de ultima actualización del registro
Nombre	Varchar	Nombre del usuario
Apellido	Varchar	Apellido del usuario
Teléfono	Varchar	Numero telefónico del usuario
Rol	Varchar	Rol o tipo de usuario
Activo	Bool	Indica si la cuenta del usuario esta activa
Ultimo_acceso	Timestamp	Fecha especifica del ultimo acceso del usuario
Fecha_registro	Date	Fecha especifica del registro del usuario
Confirmation_token	Varchar	Token usado para confirmar la cuenta del usuario
Confirmed_at	Timestamp	Fecha y hora de la cuenta que fue confirmada
Confirmation_sent_at	Timestamp	Fecha y hora del envio del correo

Unconfirmed_email	Varchar	Correo electrónico no confirmado
Authentication_token	Varchar	Token de autenticación para la api o sesiones

Tabla public_orders

Campo	Tipo dato	Descripción
Id	Integer(PK)	Identificador único de la orden
Code	Varchar	Codigo único de referencia del pedido
User_id	Integer(FK)	Referencia al usuario que realizo el pedido
Status	Varchar	Estado actual del pedido
Total	Numeric	Monto de la orden
Carrito_id	Integer(FK)	Referencia al carrito del pedido
Created_at	Timestamp	Fecha y hora de creación de la orden
Updated_at	Timestamp	Fecha y hora de la orden
Coupon_id	Integer(FK)	Referencia al cupon aplicado
Dirección	Text	Dirección del envio del pedido
Guest_nombre	Varchar	Nombre del comprador si es un invitado
Guest_apellido	Varchar	Apellido del comprador si es un invitado
Guest_telefono	Varchar	Teléfono del comprador si es un invitado
Guest_email	Varchar	Email del comprador si es un invitado

Tabla public_carritos

Campo	Tipo Dato	Descripción
Id	Integer(Pk)	Identificador único del carrito

Numero_orden	Varchar	Numero de orden asociado
User_id	Integer(Fk)	Referencia al usuario al que pertenece
Created_at	Timestamp	Fecha y hora de la creación del carrito
Updated_at	Timestamp	Fecha y hora de la actualización del carrito
Coupon_id	Integer(FK)	Referencia al cupon usado

Tabla public_products

Campo	Tipo dato	Descripción
Id	Integer(Pk)	Identificador del producto
Nombre	Varchar	Nombre del producto
Código	Varchar	Código de referencia
Precio	Float	Precio unitario
Descripción	Varchar	Descripción del producto
Disponible	Bool	Indica si el producto esta en stock y disponible
Created_at	Timestamp	Fecha y hora de la creación del producto
Updated_at	Timestamp	Fecha y hora de la actualización del producto
Grupo_id	Integer(FK)	Referencia al grupo o categoría al que pertenece
Calificación	Numeric	Calificación del producto
Activo	Bool	Producto disponible o no

Type	Varchar	Tipo de producto

Tabla Public_coupons

Campo	Tipo dato	Descripción
Id	Integer(Pk)	Identificador del cupon
Código	Varchar	Código alfanumerico que el usuario
Tipo_descuento	Varchar	Tipo de descuento
Expira_en	Numeric	Numero días o fechas de expiración
Created_at	Timestamp	Fecha y hora de creación del cupon
Updated_at	Timestamp	Fecha y hora de la ultima actualización del cupon

Tabla public_ingredients

Campo	Tipo dato	Descripción
Id	Integer(PK)	Identificador único del producto
Nombre	Varchar	Nombre del ingrediente
Created_at	Timestamp	Fecha y hora de la creación del registro del ingrediente
Updated_at	Timestamp	Fecha y hora de la ultima actualización del ingrediente

Tabla public_grupos

Campo	Tipo Dato	Descripción
Id	Integer(Pk)	Identificador único del grupo
Nombre	Varchar	Nombre del grupo o categoría

Created_at	Timestamp	Fecha y hora de creación del grupo
Updated_at	Timestamp	Fecha y hora de la actualización del grupo
Descripción	Varchar	Descripción del grupo o categoría

Tabla public_banners

Campo	Tipo Dato	Descripción
Id	Integer(Pk)	Identificador del banner
Created_at	Timestamp	Fecha y hora de la creación del banner
Updated_at	Timestamp	Fecha y hora de la actualización del banner

Tabla public_order_items

Campo	Tipo Dato	Descripción
Id	Integer(Pk)	Identificador único del item
Order_id	Integer(PK, FK)	Referencia a la orden a que pertenece el item
Producto_id	Integer(PK, FK)	Referencia al producto especifico
Quantity	Integer	Cantidad del producto en la orden
Price	Numeric	Cantidad del producto al momento de la orden
Created_at	Timestamp	Fecha y hora de la creación de la orden
Updated_at	Timestamp	Fecha y hora de la actualización del item de la orden

Tabla public_carritos_items

Campo	Tipo dato	Descripción
Id	Integer(Pk)	Identificador único del item dentro de un carrito
Carrito_id	Integer(PK, FK)	Referencia al carrito al que pertenece el item
Cantidad	integer	Cantidad del producto en el carrito
Precio	Float	Precio unitario del producto en el carrito
Product_id	Integer(Fk)	Referencia al producto especifico en la línea del carrito
Created_at	Timestamp	Fecha y hora de la creación del carrito
Updated_at	Timestamp	Fecha y hora de la ultima actualización del item en el carrito

Tabla public_combo_item

Campo	Tipo dato	Descripción
Id	Integer(PK)	Identificador único de la relación entre el combo y el producto
Combo_id	Integer(FK)	Referencia al producto que es un combo
Product_id	Integer(FK)	Referencia al producto que forma parte del combo
Cantidad	Integer	Cantidad de productos en un combo
Created_at	Timestamp	Fecha y hora de la creación del item del combo
Updated_at	Timestamp	Fecha y hora de la ultima actualización

Tabla public_payments

Campo	Tipo dato	Descripción
Id	Integer(PK)	Identificador único del pago.
Order_id	Integer(FK)	Referencia a la orden a la que corresponde el pago
Transaction_id	Varchar	Identificador único de la transación generado por el pago
Status	Varchar	Estado del pago
Amount	Numeric	Monto pagado
Payment_method	Varchar	Método de pago usado
Token	Varchar	Token de referencia de la transacción de pago
Created_at	Timestamp	Fecha y hora de la creación de pago
Updated_at	Timestamp	Fecha y hora de la última actualización del pago

Tabla public_coupon_usages

Campo	Tipo dato	Descripción
Id	Integer(Pk)	Identificador único del registro del cupon
User_id	Integer(FK)	Referencia al usuario que utilizo el cupon
Coupon_id	Integer(FK)	Referencia al cupon usado
Used_at	Timestamp	Fecha y hora del cupon usado
Created_at	Timestamp	Fecha y hora de la creación del registro de uso
Updated_at	Timestamp	Fecha y hora de la actualización del cupon

Tabla public_ingrediente_productos

Campo	Tipo dato	Descripción

Id	Integer(PK)	Identificador único de la relación
Product_id	Integer(FK)	Referencia al producto que contiene el ingrediente
Ingrediente_id	Integer(FK)	Referencia al ingrediente que esta en el producto
Created_at	Timestamp	Fecha y hora de la creación del registro
Updated_at	Timestamp	Fecha y hora de la ultima actualización del registro

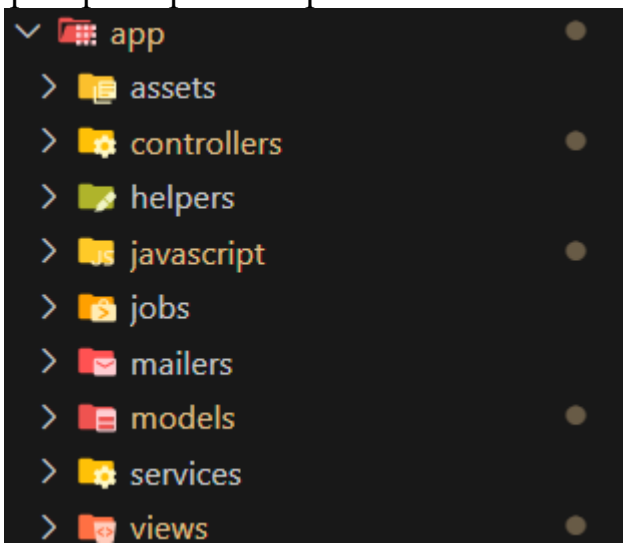
Relaciones principales

El sistema se estructura alrededor del **Cliente** (public_users), que es la entidad central, ya que cada uno puede poseer un Carrito (public_carritos) y realizar múltiples Pedidos/Órdenes (public_orders). Tanto los Carritos como los **Pedidos** están compuestos por **Ítems** (a través de public_carritos_items y public_order_items, respectivamente) que referencian los Productos (public_products) que se están comprando. Finalmente, cada Pedido se registra con un Pago (public_payments), y cada Producto pertenece a una Categoría/Grupo (public_grupos) para su clasificación.

Estructura de las carpetas

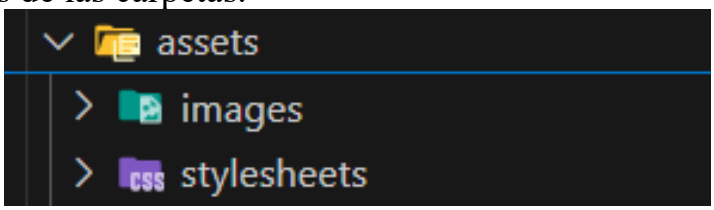
La estructura de las carpetas del Aplicativo web son las siguientes.

Carpeta principal del Aplicativo web.

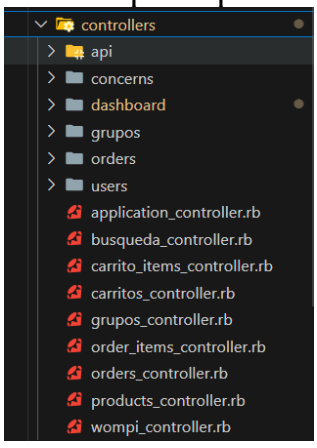


En esta carpeta “app” se encuentran carpetas importantes como “controllers”, “helpers”, “assets”, “models”, “views” y “JavaScript”

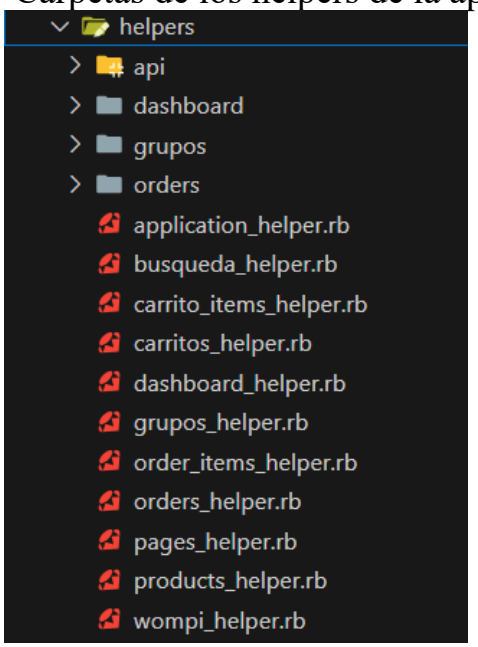
Estilos de las carpetas.



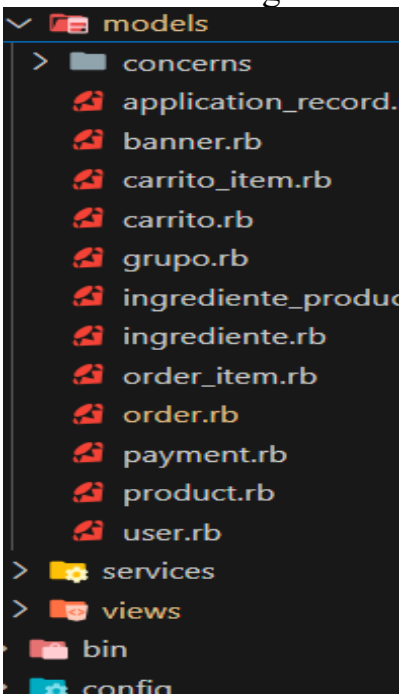
Controladores principales de la página web



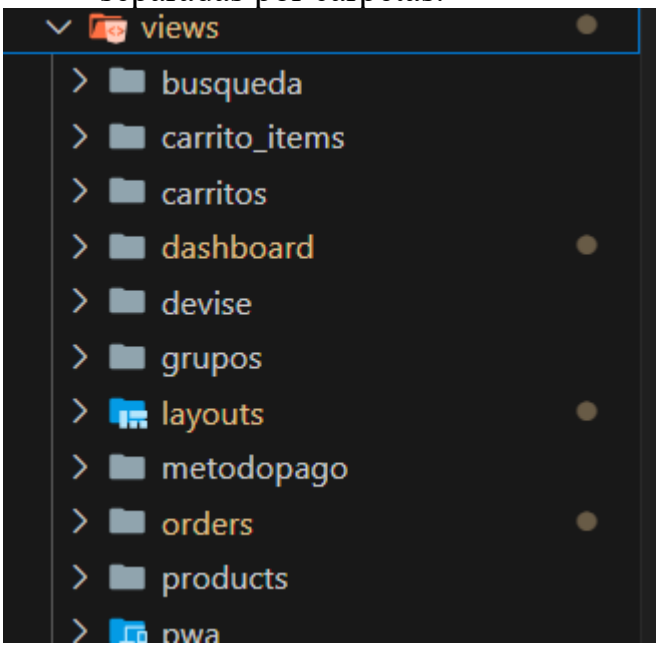
- Carpetas de los helpers de la aplicación



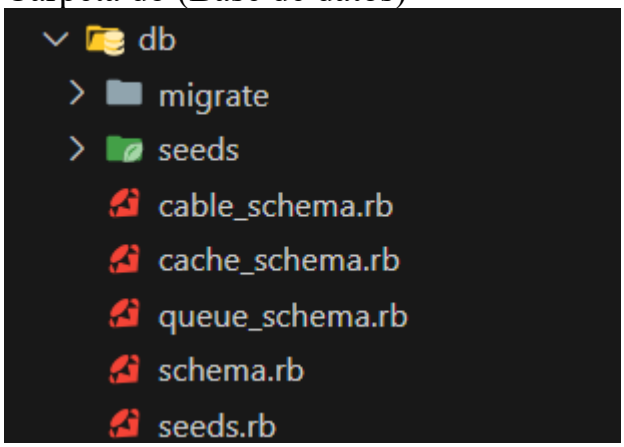
Modelos de la Pagina web



- Views de las Pagina principal, en el cual las vistas se encuentran separadas por carpetas.

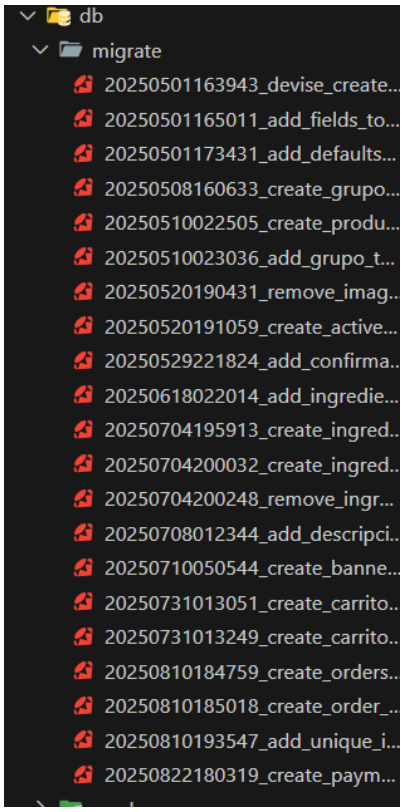


- Carpeta db (Base de datos)

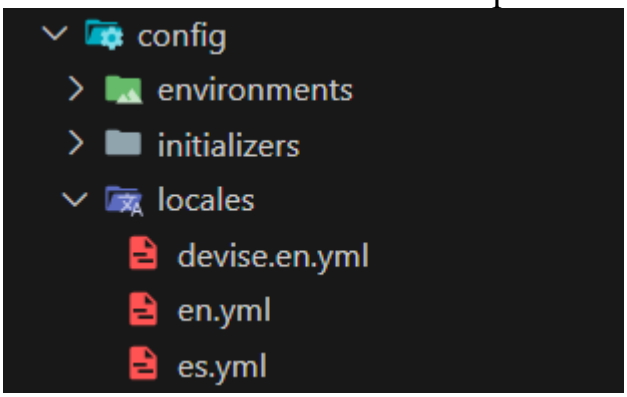


En el archivo seeds.rb se encuentran los elementos incrustados en la Base de datos.

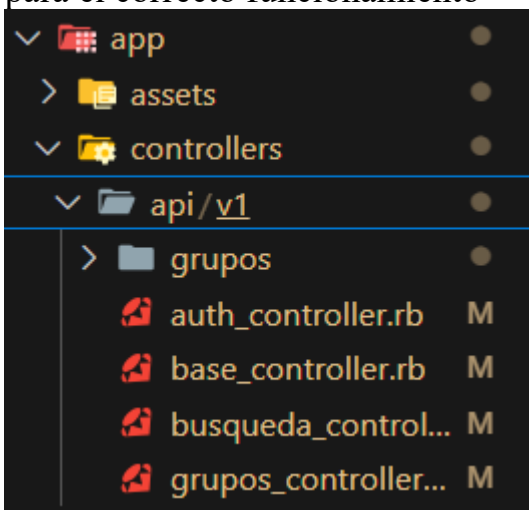
- Carpeta migrate se encuentran todas las migraciones que sean realizado dentro del Aplicativo web.



- Config/locale se encuentran los archivos que se necesitan para hacer el Cambio de idioma en el Aplicativo web.



- Api para conectar el aplicativo móvil con el backend
  Dirección en las cuales se pueden encontrar todas las APIS necesarias para el correcto funcionamiento



Descripción de la plataforma.

En la creación del software se manejaron varias herramientas las cuales son Las siguientes.

Ruby on Rails

Flutter

Chart.js

Bootstrap

JavaScript

Documentación del código fuente.

Por esta parte damos a entender partes del código fuente del software y Explicar cuáles son sus rutas dentro de las carpetas y explicación detallada Del código para un mayor entendimiento.

Código fuente Pagina web
Primero por el controlador principal de la pagina se encuentra en la siguiente rutas

```ruby
class ApplicationController < ActionController::Base
  allow_browser versions: :modern
  before_action :configure_permitted_parameters, if: :devise_controller?
  before_action :store_user_location!, if: :storable_location?
  before_action :set_locale

  layout :layout_by_resource

  private

  def storable_location?
    request.get? && is_navigational_format? && !devise_controller? &&
!request.xhr?
  end

  def store_user_location!
    store_location_for(:user, request.fullpath)
  end

  def layout_by_resource
    if devise_controller?
      "devise" # usa layouts/devise.html.erb
    else
      "application"
    end
  end

  def set_locale
    I18n.locale = params[:locale] || session[:locale] || I18n.default_locale
  end

  def default_url_options
    { locale: I18n.locale }
  end

  protected

  def configure_permitted_parameters
    devise_parameter_sanitizer.permit(:sign_up, keys: [ :nombre, :apellido,
:telefono, :email, :password, :password_confirmation ])
    devise_parameter_sanitizer.permit(:account_update, keys: [ :nombre,
:apellido, :telefono, :password, :password_confirmation, :current_password
])
  end
```

ta.

App/controllers/application_controllers.rb

Este archivo cuentas con los principales controladores de la pagina web los Cuales hay controladores, para validaciones de usuarios, manejos de Internalización.

Un ejemplo de controlador de autenticación de usuarios es el siguiente.
def storable_location?

Encargado de si se debe de guardar la dirección url después que se haya Ingresado en el sistema.
Otro controlador que se tiene en el código es del manejo del layout por defecto en el cual se mostrara en la página.

def layout_by_resource.
Con la parte de manejo de internalización se tiene estos diferentes controladores.

def set_locale
Encargo de mostrar cual es el idioma por defecto que se usara en la página web
def default_url_options

Controlador encargo del cambio de idiomas dentro de la pagina web por Medio de la url de la página, al realizar un cambio de idioma, por ejemplo: Se maneja el que está en la url puede ser es/en

Y por la parte de controladores protegidos, tenemos los que son encargados de Parámetros del inicio de sesión y registros de una cuenta.

Por otra parte, también se dará a explicar sobre algunos fragmentos del layout principal de la aplicación, que se encuentran en la siguiente ruta. App/views/layouts/application.html.rb.

```
<head>
      <title><%= content_for(:title) || "Bitevia" %></title>
      <meta name="viewport" content="width=device-width,initial-scale=1">
      <meta name="apple-mobile-web-app-capable" content="yes">
      <meta name="mobile-web-app-capable" content="yes">
      <%= csrf_meta_tags %>
      <%= csp_meta_tag %>

      <%= yield :head %>
```

Tenemos por este lado etiquetas de plantilla de HTML de las cuales definen estructuras principales de la plantilla de Home en el software.

Código de JS y CSS en el código del layout

```
      <%# Includes all stylesheet files in app/assets/stylesheets %>
      <%= stylesheet_link_tag "bootstrap.min", "data-turbo-track": "reload"
%>
      <%= stylesheet_link_tag "application", "data-turbo-track": "reload" %>
      <%= javascript_importmap_tags %>
```

Encargado el manejo de archivos JS, CSS y Bootstrap en el layout, en vez de colocar varias líneas de código para hacer conexiones entre los diferentes archivos, se maneja el uso de esos códigos para que ya sea predeterminado la conexión.
Modelos en el aplicativo se mostrará solo un modelo principal que hace parte en el aplicativo, como el cual es el modelo User.rb.
Se encuentra en la siguiente ruta App/models/user.rb

Enumeración de roles.

```
 # Enum roles
  enum :rol, { cliente: 0, empleado: 1, admin: 2 }
```

Validaciones de usuarios.

```ruby
# Validaciones
  validates :nombre, presence: true
  validates :apellido, presence: true
  validates :telefono, presence: true
```

Validaciones con el fin de aseguramiento de que esos atributos del modelo no se encuentren vacíos.

Confirmación de usuario.

```ruby
 # Permitir login si está confirmado o es admin
  def active_for_authentication?
    super && (confirmed? || admin?)
  end
```

Confirmar si un usuario esta registrado o si es un admin del sistema

Registro de acceso de empleados

```ruby
  # Registra ultimo acceso
  def after_database_authentication
    self.update_column(:ultimo_acceso, Time.current)
  end

  # Logica para usuario inactivos por mas de un mes
  def inactive_for_a_month?
    ultimo_acceso.present? && ultimo_acceso < 1.month.ago
  end
```

Campos encargados de la autenticación de empleados al hacer uso del sistema.

Controlador de validaciones de inicio sesión

```
class Api::V1::AuthController < ApplicationController
  skip_before_action :verify_authenticity_token # Para API no necesitamos
CSRF

  def login
    user = User.find_by(email: params[:email])

    if user&.valid_password?(params[:password])
      if user.active_for_authentication?
        render json: {
          token: user.authentication_token,
          user: {
            id: user.id,
            nombre: user.nombre,
            apellido: user.apellido,
            email: user.email,
            rol: user.rol,
            telefono: user.telefono
          }
        }, status: :ok
      else
        render json: { error: "Cuenta no confirmada o inactiva" }, status:
:unauthorized
      end
    else
      render json: { error: "Contraseña incorrecta" }, status: :unauthorized
    end
  end

  def logout
    user = User.find_by(authentication_token:
request.headers["Authorization"])

    if user
      user.update(authentication_token: SecureRandom.hex(20)) # invalidamos
token actual
      render json: { message: "Sesión cerrada correctamente" }, status: :ok
    else
      render json: { error: "Token no encontrado o inválido" }, status:
:unauthorized
    end
  end
end
```

Controlador para una Api diseñada para manejar las peticiones de inicio de sesión Gestionando los inicios de sesión y cierre de sesión por medio de tokens de autenticación.

Controlador de búsqueda global para el servicio de una api en el que ofrece arrays de productos y grupos que coincidan con dicho termino.

```ruby
# frozen_string_literal: true

module Api
  module V1
    class BusquedaController < ApplicationController
      skip_before_action :verify_authenticity_token
      before_action :authorize_request, except: [:index]

      def index
        if params[:q].present?
          raw_query = params[:q].downcase.strip
          query = I18n.transliterate(raw_query)
          terms = query.split

          # Buscar grupos que coincidan
          @grupos = Grupo.all.select do |g|
            normalizado = I18n.transliterate(g.nombre.downcase.strip)
            terms.all? { |t| normalizado.include?(t) }
          end.sort_by(&:id)

          # Buscar productos que coincidan (en nombre del producto o nombre
del grupo)
          @productos = Product.all.select do |p|
            normalizado_producto =
I18n.transliterate(p.nombre.downcase.strip)
            normalizado_grupo = p.grupo ?
I18n.transliterate(p.grupo.nombre.downcase.strip) : ""

            # Buscar en el nombre del producto O en el nombre del grupo
            terms.all? { |t| normalizado_producto.include?(t) ||
normalizado_grupo.include?(t) }
          end.sort_by(&:id)

          render json: {
            productos: @productos.as_json(
              only: [:id, :nombre, :descripcion, :precio, :grupo_id],
              methods: [:imagen_url, :ingredientes]
            ),
            grupos: @grupos.as_json(
              only: [:id, :nombre, :descripcion],
              methods: [:imagen_url]
            ),
            total: @productos.size + @grupos.size
          }
        else
```

Acuerdos de nivel de servicio (ANS)

El personal del soporte técnico es responsable de manejar los requerimientos Finales utilizando la herramienta del sistema Bitevia software

Todos los técnicos deben de estar relacionados con los niveles de prioridad y sus respectivas tiempo de respuesta, los cuales son los siguientes.

- Bajo: problemas no críticos que no dañen la funcionalidad
  Tiempo de respuesta: 24 horas laborales.

- Normal: Problemas que afectan la funcionalidad principal, pero no impiden el trabajo. (ejemplo. un formulario no enviar dato)
  Tiempo de respuesta: 8 horas laborales.

- Alto: problemas que impiden a un usuario realizar una tarea (ejemplo. Acceso denegado a un usuario)
  Tiempo de respuesta: 4 horas laborales.

- Critico: Problemas que afectan a múltiples usuarios o detienen varios Procesos del negocio (ejemplo. Caída del servidor)
  Tiempo de respuesta: 1 hora laboral

Procedimiento de escalado

El escalado se debe de aplicar cuando un problema no puede ser resuelto por El técnico encargado del primer nivel. El requerimiento debe ser Escalado si:

- El problema requiere de intervención de un supervisor o gerente de aérea.
- Se necesita un reembolso o una decisión que el empleado no está autorizado a tomar.

Instrucciones de escalado

Todo problema que no se pueda resolver de una manera inmediata deber ser reportado al supervisor de aérea. El técnico debe de dar detalles del problema Y las acciones que se tomaron antes de escalarlo.

Contactos de personas encargadas del desarrollo del Software.

- Juan Esteban Caballero Goenaga – Desarrollador en Pagina web y Aplicativo.
  Git Hub https://github.com/JuanCaballero09
  Correo Electrónico juanes0921200@gmail.com

- Santiago David Zambrano Izaquita – Desarrollador en Pagina Web
  Git Hub https://github.com/San5472
  Correo Electrónico Santiagozambrano751@gmail.com

- Andrw Stiven Barrera Poveda – Desarrollador en el Aplicativo móvil.
  Git Hub https://github.com/andrw790
  Correo Electrónico andrwbarrera79@gmail.com

- Miguel Sarabia Soto - Desarrollador en el Aplicativo móvil.
  Git Hub https://github.com/MiguelSarabiaSoto
  Correo Electrónico Miguelsarabia0812@gmail.com

- Wilber Robles Mercado - Desarrollador en el Aplicativo móvil.
  Git Hub https://github.com/Tribalsoft
  Correo Electrónico wilrm8sena@gmail.com

Manual técnico Bitevia software Móvil

Componentes

Alcance

El programa Móvil fue creado para dar soluciones a problemas encontrados en el local "La Terraza del Pri", ubicado en la Calle 56 #34-6, barrio Ciudadela Metropolitana, Soledad, Atlántico.

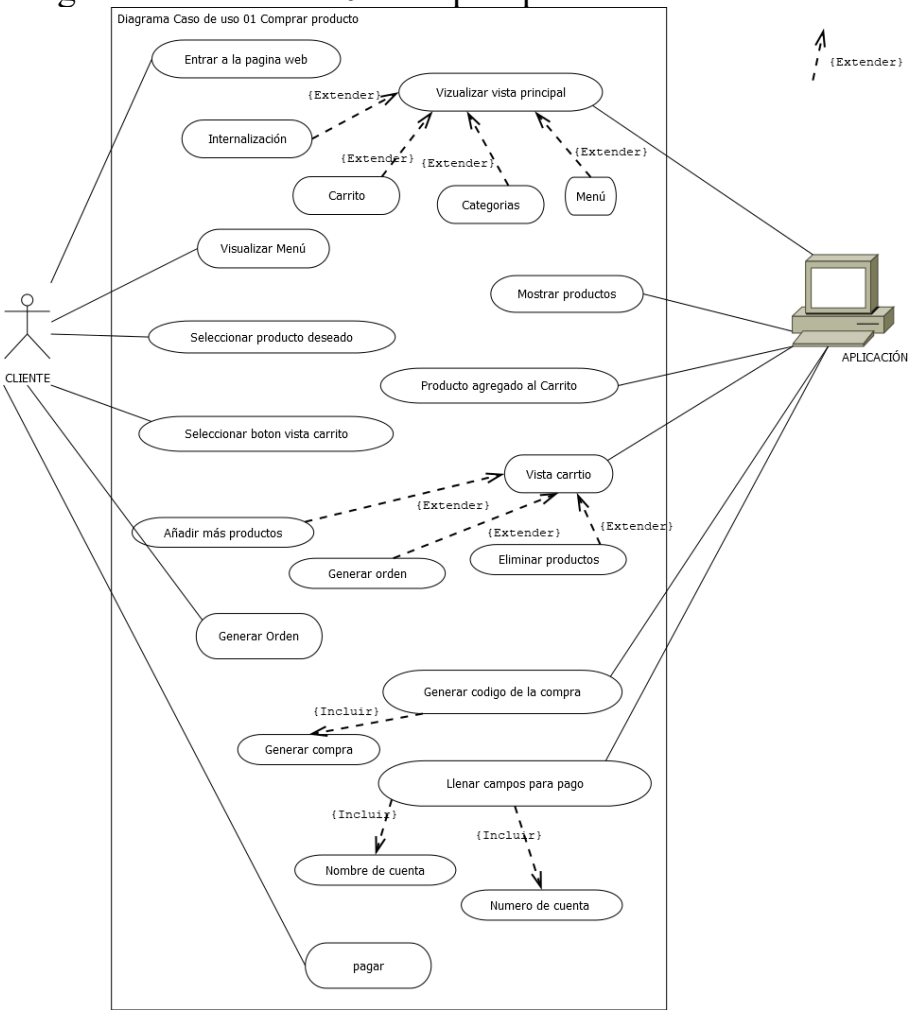
El programa realizara las funciones visualización de diferentes productos del Local, cuenta con la funcionalidad de poder hacer compras desde la Aplicación, a la realización de las compras contara con un sistema de Recolección de la dirección de las personas, para que las personas encargadas De la entrega del producto, sepa en donde se realizara la entrega y por último Contara con sistema notificaciones Push y fidelización de clientes.

Descripción de los procesos.

La aplicación móvil está diseñada para brindar una experiencia optimizada y accesible desde dispositivos Android/iOS. El proceso inicia con la instalación de la app desde la tienda correspondiente. Al abrirla, el usuario accede a una interfaz simplificada e intuitiva, donde también debe autenticarse. Posteriormente, puede realizar las principales operaciones del sistema, como registrar información, consultar datos y recibir notificaciones en tiempo real. La app se comunica con el backend mediante servicios API, lo que asegura sincronización constante con la base de datos utilizada en la página web.

Diagramas

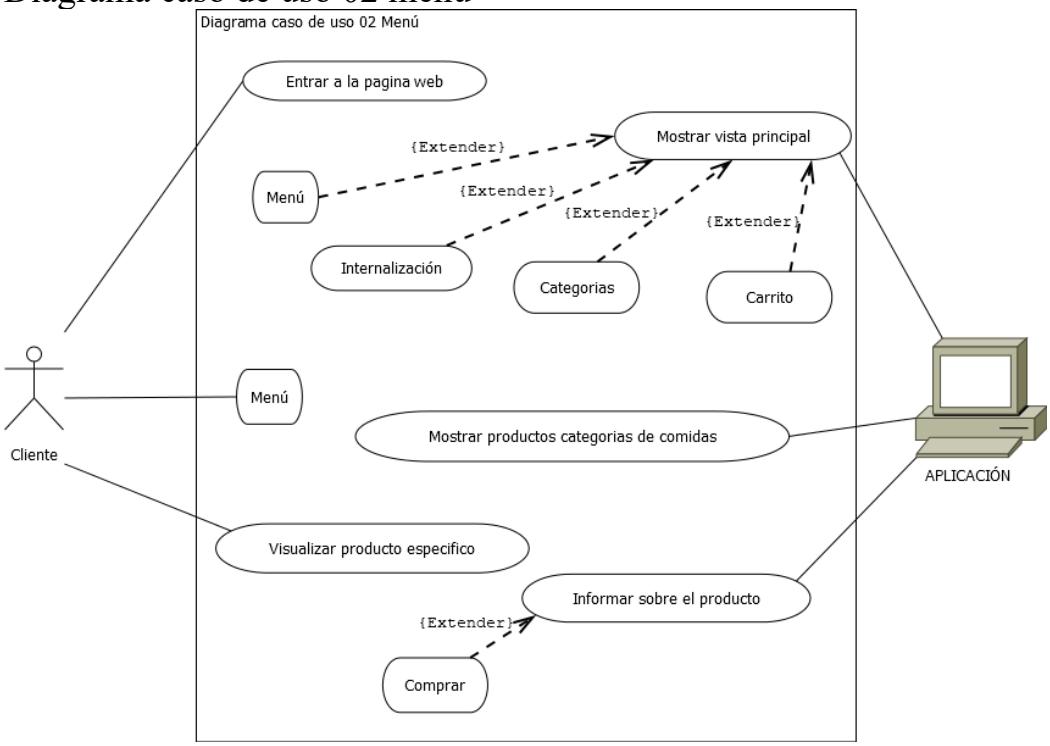
Diagrama caso de uso 01 comprar producto



Ficha caso de uso 01 comprar producto

Ficha caso de uso cliente
Nombre: ficha caso de uso Clientes 01 Comprar producto
Autor: Juan caballero, Santiago zambrano, Andrw Barrera, Miguel Sarabia y Wilber Robles
Actor: Cliente
Precondicional
1. El Cliente debe tener una computadora
2. El Cliente debe tener un telefono movil
3. El Cliente debe de tener conexión a internet
Flujo Normal
1. El Cliente ingresa a la pagina web
2. La pagina web muestra la vista principal y sus diferentes opciones (Menú, categorias, carrito y internalización)
3. El Cliente selecciona Menú.
4. La pagina web muestra los productos dentro del menú
5. El Cliente selecciona su producto preferido
6. La pagina web le da una alerta de su producto agregado al carrito
7. El Cliente se dirige al boton de la vista carrito
8. La pagina web le muestra la vista y las diferentes opciones del carrito (Generar orden, añadir productos, eliminar productos).
9. El Cliente oprime para generar la orden del pago
10. La pagina web le muestra una vista en donde se realiza el codigo de la orden y confirmación de la compra.
11. El Cliente procede con generar orden.
12. La pagina web, muestra un formulario de llenado para realizar el pago
13 La pagina web, muestra los campos del formulario (Nombre de la cuenta y Numero de la cuenta)
14. El cliente realiza el pago.
Flujo alternativo
1.1 El cliente no cuenta con conexión a internet.
2.1 Caida de la pagina web
3.1 La pagina demora en realizar las solicitudes del cliente.
4.1 El cliente no cuente con dinero en su cuenta

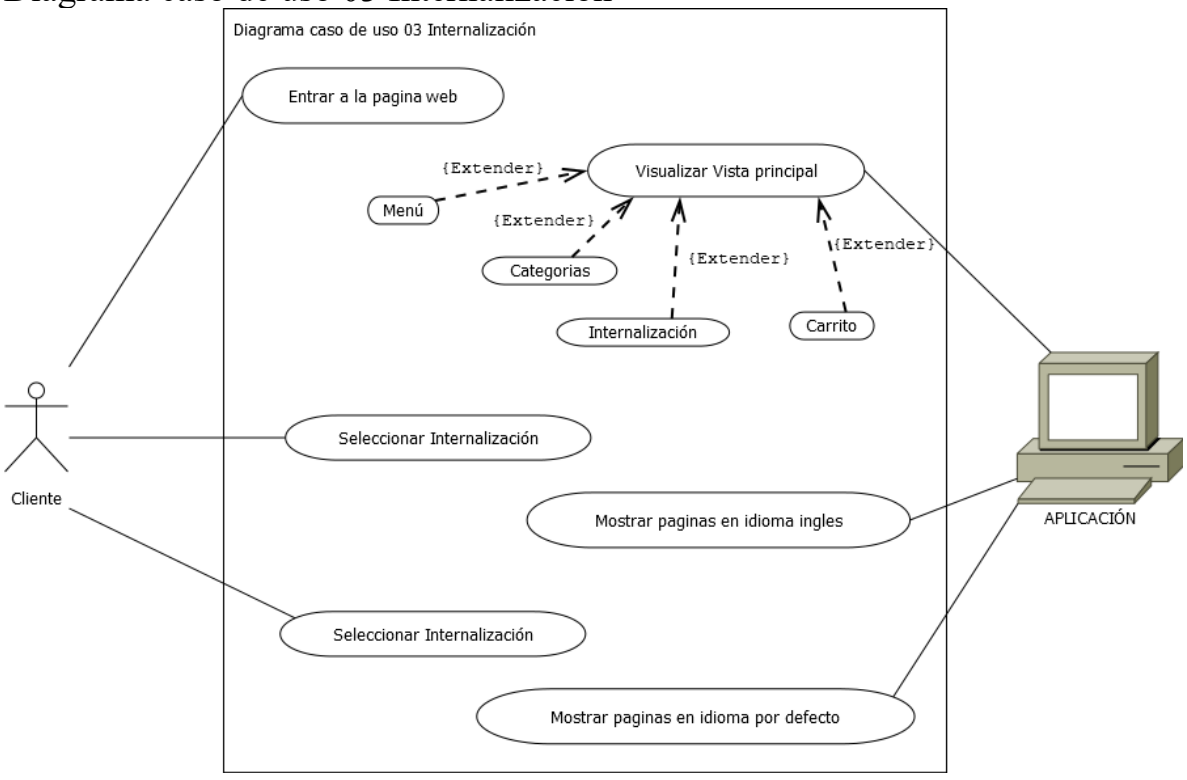
Diagrama caso de uso 02 menú



Ficha caso de uso 02 menú

Nombre: ficha caso de uso Clientes 02 Menú
Autor: Juan caballero, Santiago zambrano, Andrw Barrera, Miguel Sarabia y Wilber Robles
Actor: Cliente
Precondicional
1. El Cliente debe tener una computadora
2. El Cliente debe tener un telefono movil
3. El Cliente debe de tener conexión a internet
Flujo Normal
1. El Cliente ingresa a la pagina web
2. La pagina web muestra la vista principal y sus diferentes opciones (Menú, categorias, carrito y internalización)
3. El Cliente selecciona Menú.
4. La pagina web muestra los productos dentro del menú
5. El Cliente selecciona su producto y lo visualiza
6. La pagina web le muestra toda la información al respecto del producto
7. La pagina web muestra la opción de comprar el producto
Flujo alternativo
1.1 El cliente no cuenta con conexión a internet.
2.1 Caida de la pagina web
3.1 La pagina demora en realizar las solicitudes del cliente.
4.1 La pagina no muestra los productos

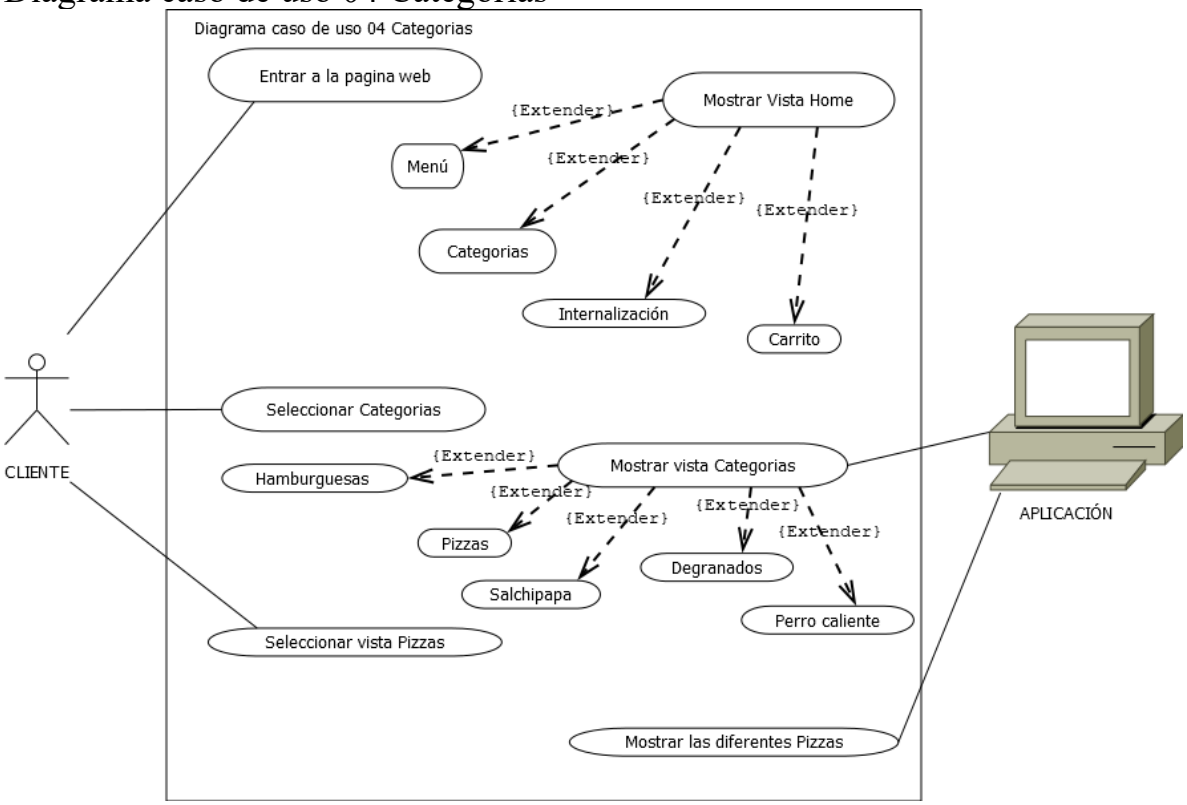
Diagrama caso de uso 03 Internalización



Ficha caso de uso 03

Nombre: ficha caso de uso Clientes 03 Internalización
Autor: Juan caballero, Santiago zambrano, Andrw Barrera, Miguel Sarabia y Wilber Robles
Actor: Cliente
Precondicional
1. El Cliente debe tener una computadora
2. El Cliente debe tener un telefono movil
3. El Cliente debe de tener conexión a internet
Flujo Normal
1. El Cliente ingresa a la pagina web
2. La pagina web muestra la vista principal y sus diferentes opciones (Menú, categorias, carrito y internalización)
3. El Cliente selecciona El boton de Internalización.
4. La pagina web muestra las diferentes sus diferentes paginas, en el idioma ingles
5. El Cliente vuelve a seleccionar el boton de Internalización
6. La pagina web le vuelve a mostrar todas las paginas en el idioma por defecto
Flujo alternativo
1.1 El cliente no cuenta con conexión a internet.
2.1 Caida de la pagina web
3.1 La pagina demora en realizar las solicitudes del cliente.

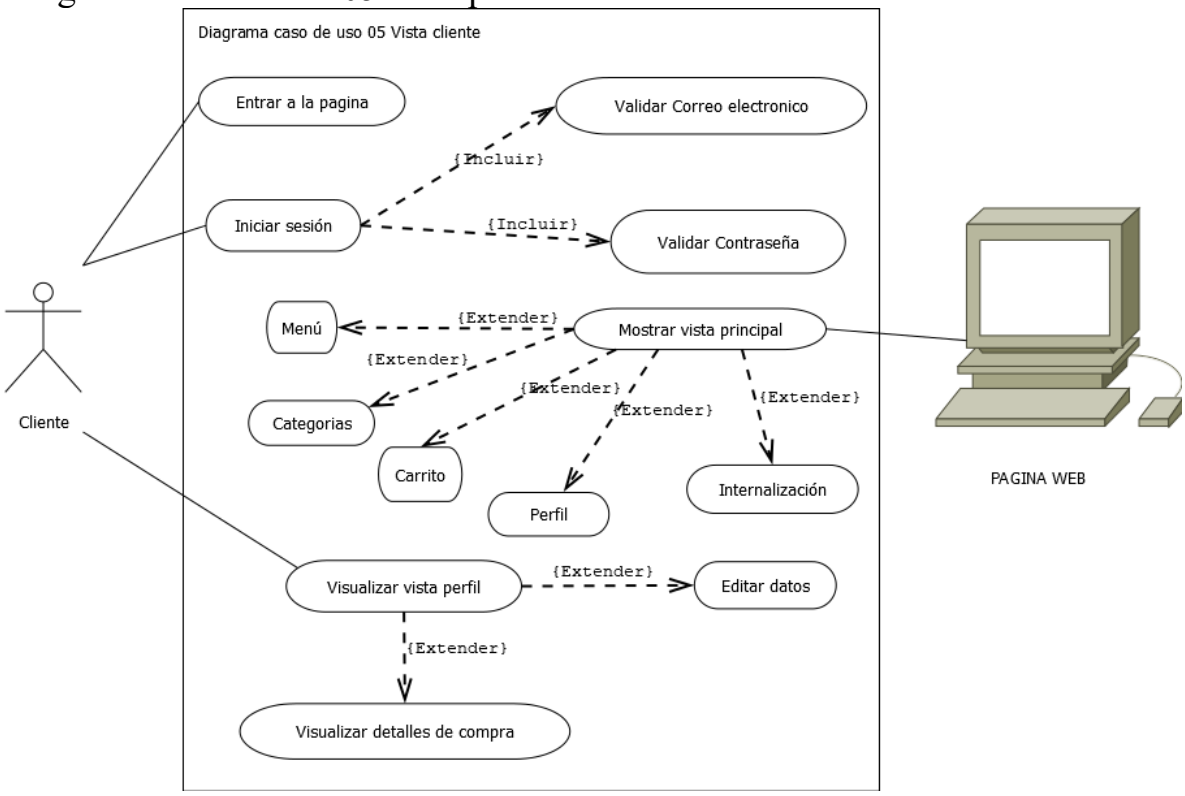
Diagrama caso de uso 04 Categorías



Ficha caso de uso 04 categorías

Nombre: ficha caso de uso Clientes 04 Categorias
Autor: Juan caballero, Santiago zambrano, Andrw Barrera, Miguel Sarabia y Wilber Robles
Actor: Cliente
Precondicional
1. El Cliente debe tener una computadora
2. El Cliente debe tener un telefono movil
3. El Cliente debe de tener conexión a internet
Flujo Normal
1. El Cliente ingresa a la pagina web
2. La pagina web muestra la vista principal y sus diferentes opciones (Menú, categorias, carrito y internalización)
3. El Cliente selecciona Categorias.
4. La pagina web muestra las diferentes categorias(Hamburguesa, pizza, salchipapa, degranado,perro caliente)
5. El Cliente selecciona la categoria de las pizzas
6. La pagina web le muestra las diferentes pizzas que pertenecen a esa categoria
Flujo alternativo
1.1 El cliente no cuenta con conexión a internet.
2.1 Caida de la pagina web
3.1 La pagina demora en realizar las solicitudes del cliente.

Diagrama caso de uso 05 vista perfil

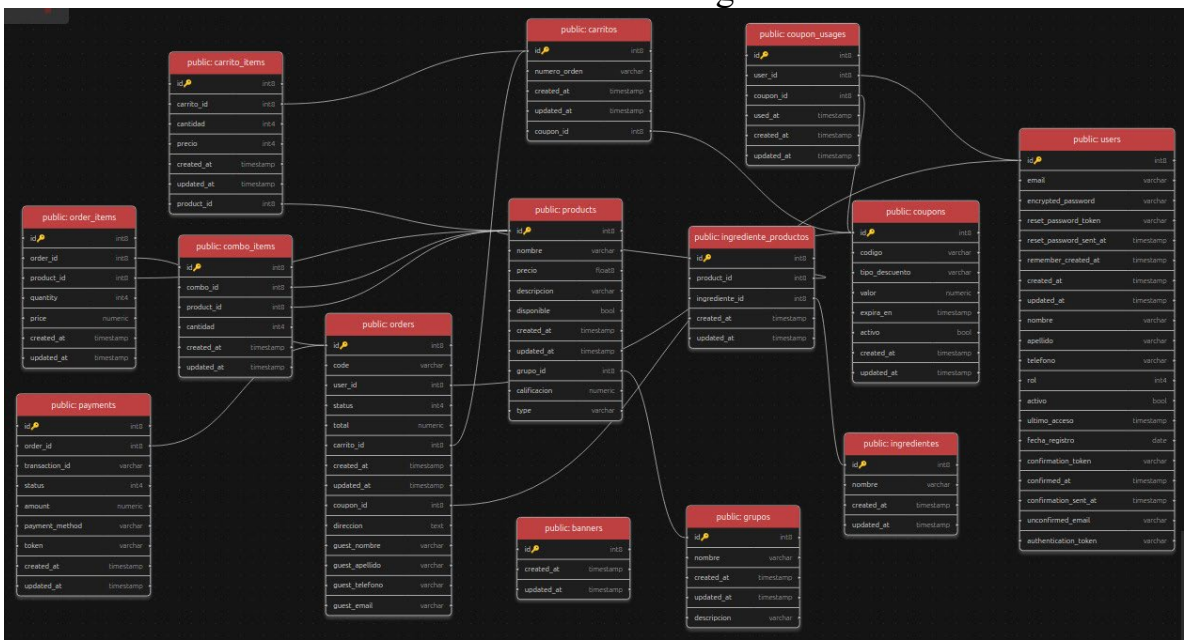


Ficha caso de uso 05 vista perfil

Ficha caso de uso cliente
Nombre: ficha caso de uso Clientes 05 Vista perfil
Autor: Juan caballero, Santiago zambrano, Andrw Barrera, Miguel Sarabia y Wilber Robles
Actor: Cliente
Precondicional
1. El Cliente debe tener una computadora
2. El Cliente debe tener un telefono movil
3. El Cliente debe de tener conexión a internet
Flujo Normal
1. El Cliente ingresa a la pagina web
2. El Cliente inicia sesión
3. La pagina Web valida los datos de la persona como correo electronico y contraseña
4. La pagina web muestra las diferentes opciones (Menú, caategorias, carrito, perfil e internalización)
5. El Cliente selecciona vista perfil
6. La pagina web le muestra las diferentes opciones. (Editar datos, eliminar cuenta y visualizar detalles de la compra)
Flujo alternativo
1.1 El cliente no cuenta con conexión a internet.
2.1 El cliente inicia como invitado
3.1 Aviso de error, datos incorrectos
4.1 Caida del servidor de la pagina

Modelo relacional en la base de datos.

El modelo relacional de la base de datos es el siguiente.



En caso de alguna confusión el Aplicativo móvil trabaja junto a la pagina Web por lo tanto manejan el mismo modelo relacional de base de datos.

Descripción de la plataforma.

En la creación del software se manejaron varias herramientas de las cuales son Las siguientes.

Como lenguaje de programación principal se uso Dart con su framework de Flutter. Para hacer las pruebas de las conexiones con el backend, principal se Manejó ngrok. Para conectar el backend con ngrok, se manejó uso de APIS Para hacer uso de la conexión.

Para uso de las vistas y mejor presentación se manejó.

Bootstrap.

Documentación del código fuente
Código aplicación flutter

Primero se definen parámetros de los cuales se trabajarán desde otros archivos Del Softtware.
Dirección del código : first_flutter/lib/models/producto.dart

```dart
import 'package:equatable/equatable.dart';
import 'package:flutter/foundation.dart';
import '../core/config/api_config.dart';

class Product extends Equatable {
  final int id;
  final String name;
  final String category;
  final double price;
  final String description;
  final String image;
  final List<String> ingredients;

  const Product({
    required this.id,
    required this.name,
    required this.category,
    required this.price,
    required this.description,
    required this.image,
    this.ingredients = const [],
  });
```

```dart
factory Product.fromJson(Map<String, dynamic> json) {
  // Debug: Ver qué datos vienen del JSON
  debugPrint('=== DEBUG: Product.fromJson ===');
  debugPrint('JSON recibido: $json');

  List<String> ingredientsList = [];

  // Check for ingredients in both languages and formats
  final ingredientsField = json['ingredientes'] ?? json['ingredients'];
  if (ingredientsField != null) {
    if (ingredientsField is String) {
      // Si es un String, dividir por comas
      ingredientsList = ingredientsField
          .split(',')
          .map((e) => e.trim())
          .where((e) => e.isNotEmpty)
          .toList();
    } else if (ingredientsField is List) {
      // Si es una lista, puede contener Strings o Maps
      ingredientsList = ingredientsField.map((item) {
        if (item is String) {
          return item;
        } else if (item is Map) {
          // Si es un Map, extraer el campo 'nombre'
          return (item['nombre'] ?? item['name'] ?? '').toString();
        }
        return item.toString();
      }).where((e) => e.isNotEmpty).toList();
    }
  }
```

```dart
int productId;
if (json['id'] is String) {
  productId = int.tryParse(json['id']) ?? 0;
} else {
  productId = json['id'] ?? 0;
}

// Handle price conversion
double productPrice;
final priceField = json['price'] ?? json['precio'] ?? 0;
if (priceField is String) {
  productPrice = double.tryParse(priceField) ?? 0.0;
} else {
  productPrice = (priceField as num).toDouble();
}

// Map categoria_id to category names (transformado desde grupo_id)
String category = json['category'] ?? json['categoria'] ?? 'General';
if (category == 'General' && json['categoria_id'] != null) {
  switch (json['categoria_id']) {
    case 1:
      category = 'hamburguesas';
      break;
    case 2:
      category = 'salchipapas';
      break;
    case 3:
      category = 'pizzas';
      break;
    case 4:
      category = 'bebidas';
      break;
    case 5:
      category = 'postres';
      break;
    default:
      category = 'General';
  }
}
```

Siguiendo con demás códigos del software, se realiza la importación de los Métodos, que pueden ser heredados y en el siguiente se obtiene un atributo Abstracto que fue heredado y que posteriormente fue sobrescrito.
Dirección del archivo: firts_flutter/lib/bloc/producto_event.dart

```dart
import 'package:flutter_bloc/flutter_bloc.dart';
import '../../core/errors/exceptions.dart';
import '../../repository/api_product_repository.dart';
import '../base_state.dart';
import 'product_event.dart';
import '../../repository/product_repository.dart';

/// BLoC que maneja el estado de los productos
/// Soporta tanto repositorios locales como de API
class ProductBloc extends Bloc<ProductEvent, BaseState> {
  final ProductRepository repository;

  ProductBloc(this.repository) : super(const InitialState()) {

    // Manejo del evento para obtener todos los productos
    on<FetchProducts>((event, emit) async {
      emit(const LoadingState(message: 'Loading products...'));
      try {
        final products = await repository.getProducts();
        emit(SuccessState(products));
      } on NetworkException catch (e) {
        emit(ErrorState(
          message: e.message,
          code: e.code,
          onRetry: () => add(FetchProducts()),
        )); // ErrorState
      } on DataException catch (e) {
        emit(ErrorState(
          message: e.message,
          code: e.code,
          onRetry: () => add(FetchProducts()),
        )); // ErrorState
      } catch (e) {
        emit(ErrorState(
          message: 'Unexpected error loading products',
          onRetry: () => add(FetchProducts()),
        )); // ErrorState
      }
    });
```

Api que conecta el backend. Ruby on rails, con flutter, es la siguiente con su ruta.

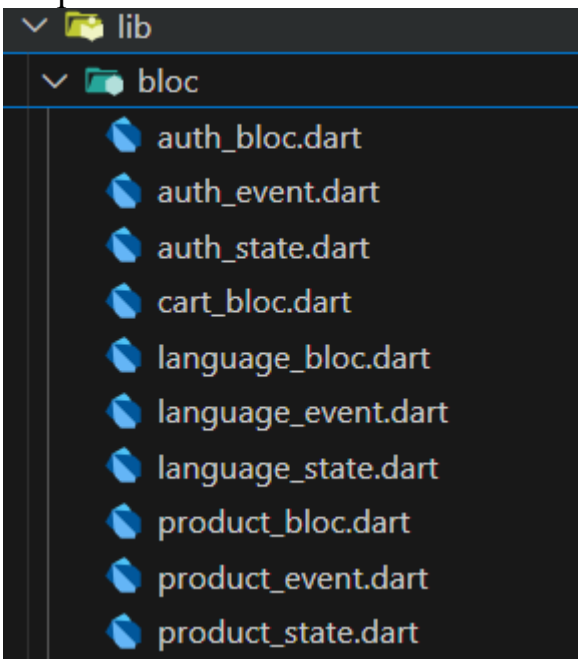
First flutter/lib/service/api_service.dart

```dart
import 'dart:convert';
import 'dart:async';
import 'package:flutter/foundation.dart';
import 'package:flutter_secure_storage/flutter_secure_storage.dart';
import 'package:http/http.dart' as http;
import '../models/product.dart';
import '../core/errors/exceptions.dart';
import '../core/config/api_config.dart';

/// Servicio centralizado para el consumo de APIs
/// Maneja autenticación, productos y categorías
class ApiService {
  /// URL base configurable de la API
  final String baseUrl;
  final storage = const FlutterSecureStorage();

  /// Constructor que permite configurar la URL base
  /// [baseUrl] - URL base de la API (opcional, usa configuración por defecto)
  ApiService({String? baseUrl}) : baseUrl = baseUrl ?? ApiConfig.currentBaseUrl;

  /// Obtiene el token de autenticación almacenado
  Future<String?> _getAuthToken() async {
    return await storage.read(key: 'token');
  }

  /// Headers comunes para las peticiones autenticadas
  Future<Map<String, String>> _getAuthHeaders() async {
    final token = await _getAuthToken();
    return {
      ...ApiConfig.defaultHeaders,
      if (token != null) 'Authorization': token,
    };
  }
```

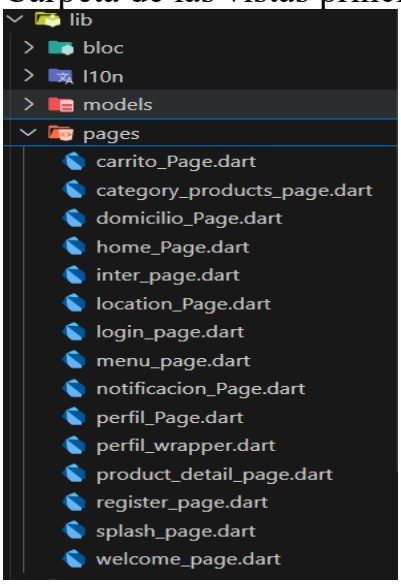
```dart
/// 🔄 Función auxiliar para transformar JSON: grupo → categoria
/// Transforma grupo_id → categoria_id y grupo → categoria
Map<String, dynamic> _transformJsonToCategoria(Map<String, dynamic> json) {
  final transformed = Map<String, dynamic>.from(json);

  // Transformar grupo_id → categoria_id
  if (json.containsKey('grupo_id')) {
    transformed['categoria_id'] = json['grupo_id'];
    transformed.remove('grupo_id');
  }

  // Transformar grupo → categoria
  if (json.containsKey('grupo')) {
    transformed['categoria'] = json['grupo'];
    transformed.remove('grupo');
  }

  return transformed;
}

/// 🔄 Transformar lista de productos/categorías
List<dynamic> _transformList(List<dynamic> items) {
  return items.map((item) {
    if (item is Map<String, dynamic>) {
      return _transformJsonToCategoria(item);
    }
    return item;
  }).toList();
}
```

Estructura de las carpetas

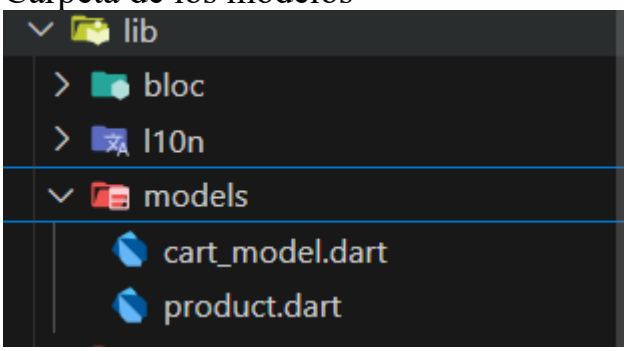
Carpeta bloc



Carpeta de las vistas principales



Carpeta de los modelos



Acuerdos de los niveles de (ANS)

El personal del soporte técnico es responsable de manejar los requerimientos Finales utilizando la herramienta del sistema Bitevia software

Todos los técnicos deben de estar relacionados con los niveles de prioridad y sus respectivas tiempo de respuesta, los cuales son los siguientes.

- Bajo: problemas no críticos que no dañen la funcionalidad
  Tiempo de respuesta: 24 horas laborales.

- Normal: Problemas que afectan la funcionalidad principal, pero no impiden el trabajo. (ejemplo. un formulario no enviar dato)
  Tiempo de respuesta: 8 horas laborales.

- Alto: problemas que impiden a un usuario realizar una tarea (ejemplo. Acceso denegado a un usuario)
  Tiempo de respuesta: 4 horas laborales.

- Critico: Problemas que afectan a múltiples usuarios o detienen varios Procesos del negocio (ejemplo. Caída del servidor)
  Tiempo de respuesta: 1 hora laboral

Procedimiento de escalado

El escalado se debe de aplicar cuando un problema no puede ser resuelto por El técnico encargado encargado del primer nivel. El requerimiento debe ser Escalado si:

- El problema requiere de intervención de un supervisor o gerente de aérea.
- Se necesita un reembolso o una decisión que el empleado no está autorizado a tomar

Instrucciones de escalado

Todo problema que no se pueda resolver de una manera inmediata deber ser reportado al supervisor de aérea. El técnico debe de dar detalles del problema Y las acciones que se tomaron antes de escalarlo.

Contactos de personas encargadas del desarrollo del Software.

- Juan Esteban Caballero Goenaga – Desarrollador en Pagina web y Aplicativo.
  Git Hub https://github.com/JuanCaballero09
  Correo Electrónico juanes0921200@gmail.com

- Santiago David Zambrano Izaquita – Desarrollador en Pagina Web
  Git Hub https://github.com/San5472
  Correo Electrónico Santiagozambrano751@gmail.com

- Andrw Stiven Barrera Poveda – Desarrollador en el Aplicativo móvil.
  Git Hub https://github.com/andrw790
  Correo Electrónico andrwbarrera79@gmail.com

- Miguel Sarabia Soto - Desarrollador en el Aplicativo móvil.
  Git Hub https://github.com/MiguelSarabiaSoto
  Correo Electrónico Miguelsarabia0812@gmail.com

- Wilber Robles Mercado - Desarrollador en el Aplicativo móvil.
  Git Hub https://github.com/Tribalsoft
  Correo Electrónico wilrm8sena@gmail.com