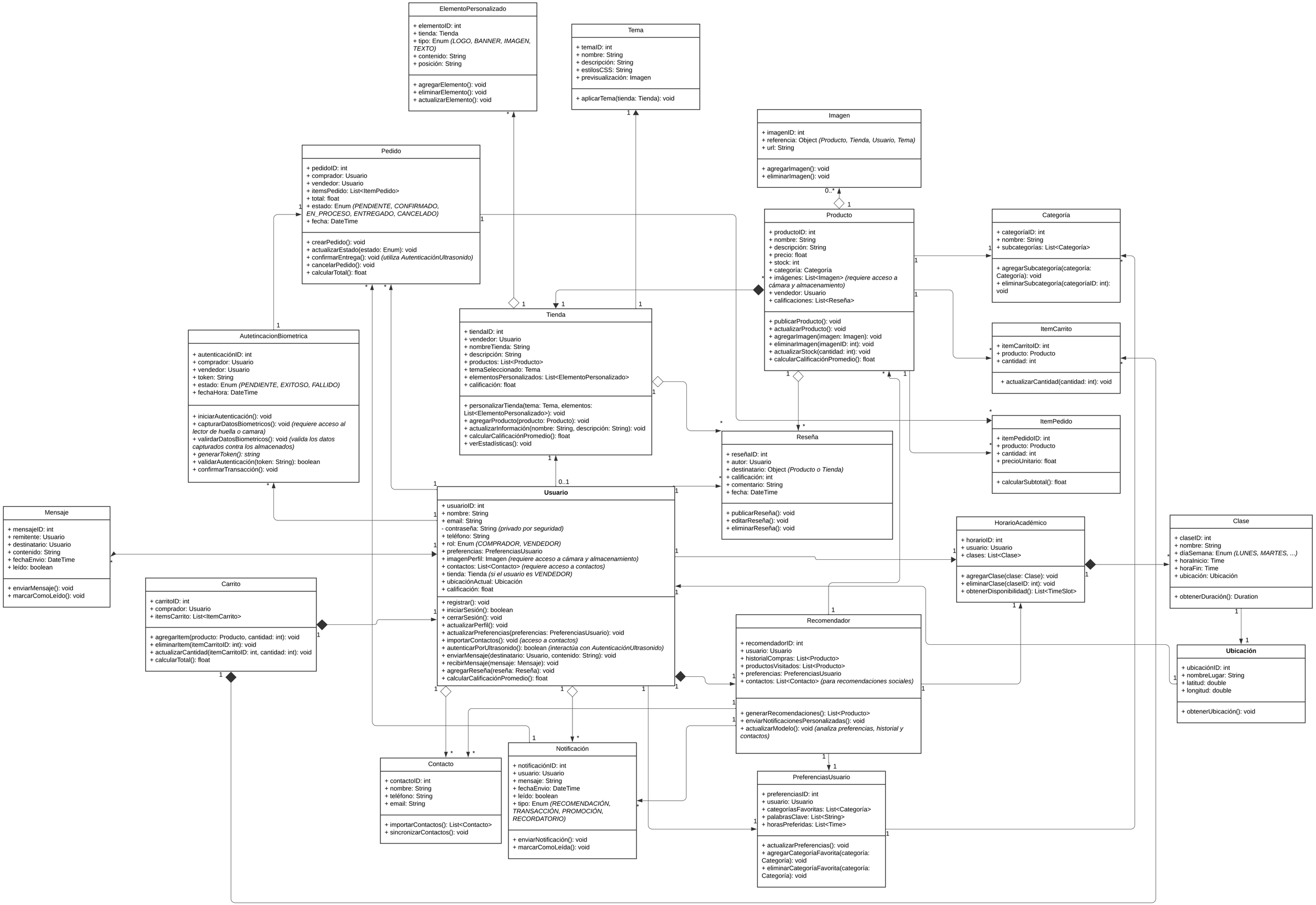
ElementoPersonalizado
+ elementoID: int
+ tienda: Tienda
+ tipo: Enum (LOGO, BANNER, IMAGEN, TEXTO)
+ contenido: String
+ posición: String
+ agregarElemento(): void
+ eliminarElemento(): void
+ actualizarElemento(): void
Tema
+ temaID: int
+ nombre: String
+ descripción: String
+ estilosCSS: String
+ previsualización: Imagen
+ aplicarTema(tienda: Tienda): void
Imagen
+ imagenID: int
+ referencia: Object (Producto, Tienda, Usuario, Tema)
+ url: String
+ agregarImagen(): void
+ eliminarImagen(): void
Pedido
+ pedidoID: int
+ comprador: Usuario
+ vendedor: Usuario
+ itemsPedido: List<ItemPedido>
+ total: float
+ estado: Enum (PENDIENTE, CONFIRMADO, EN_PROCESO, ENTREGADO, CANCELADO)
+ fecha: DateTime
+ crearPedido(): void
+ actualizarEstado(estado: Enum): void
+ confirmarEntrega(): void (utiliza AutenticaciónUltrasonido)
+ cancelarPedido(): void
+ calcularTotal(): float
Producto
+ productoID: int
+ nombre: String
+ descripción: String
+ precio: float
+ stock: int
+ categoría: Categoría
+ imágenes: List<Imagen> (requiere acceso a cámara y almacenamiento)
+ vendedor: Usuario
+ calificaciones: List<Reseña>
+ publicarProducto(): void
+ actualizarProducto(): void
+ agregarImagen(imagen: Imagen): void
+ eliminarImagen(imagenID: int): void
+ actualizarStock(cantidad: int): void
+ calcularCalificaciónPromedio(): float
Categoría
+ categoríaID: int
+ nombre: String
+ subcategorías: List<Categoría>
+ agregarSubcategoría(categoría: Categoría): void
+ eliminarSubcategoría(categoríaID: int): void
ItemCarrito
+ itemCarritoID: int
+ producto: Producto
+ cantidad: int
+ actualizarCantidad(cantidad: int): void
ItemPedido
+ itemPedidoID: int
+ producto: Producto
+ cantidad: int
+ precioUnitario: float
+ calcularSubtotal(): float
Tienda
+ tiendaID: int
+ vendedor: Usuario
+ nombreTienda: String
+ descripción: String
+ productos: List<Producto>
+ temaSeleccionado: Tema
+ elementosPersonalizados: List<ElementoPersonalizado>
+ calificación: float
+ personalizarTienda(tema: Tema, elementos: List<ElementoPersonalizado>): void
+ agregarProducto(producto: Producto): void
+ actualizarInformación(nombre: String, descripción: String): void
+ calcularCalificaciónPromedio(): float
+ verEstadísticas(): void
AutetincacionBiometrica
+ autenticaciónID: int
+ comprador: Usuario
+ vendedor: Usuario
+ token: String
+ estado: Enum (PENDIENTE, EXITOSO, FALLIDO)
+ fechaHora: DateTime
+ iniciarAutenticación(): void
+ capturarDatosBiometricos(): void (requiere acceso al lector de huella o camara)
+ valirdarDatosBiometricos(): void (valida los datos capturados contra los almacenados)
+ generarToken(): string
+ validarAutenticación(token: String): boolean
+ confirmarTransacción(): void
Reseña
+ reseñaID: int
+ autor: Usuario
+ destinatario: Object (Producto o Tienda)
+ calificación: int
+ comentario: String
+ fecha: DateTime
+ publicarReseña(): void
+ editarReseña(): void
+ eliminarReseña(): void
Usuario
+ usuarioID: int
+ nombre: String
+ email: String
- contraseña: String (privado por seguridad)
+ teléfono: String
+ rol: Enum (COMPRADOR, VENDEDOR)
+ preferencias: PreferenciasUsuario
+ imagenPerfil: Imagen (requiere acceso a cámara y almacenamiento)
+ contactos: List<Contacto> (requiere acceso a contactos)
+ tienda: Tienda (si el usuario es VENDEDOR)
+ ubicaciónActual: Ubicación
+ calificación: float
+ registrar(): void
+ iniciarSesión(): boolean
+ cerrarSesión(): void
+ actualizarPerfil(): void
+ actualizarPreferencias(preferencias: PreferenciasUsuario): void
+ importarContactos(): void (acceso a contactos)
+ autenticarPorUltrasonido(): boolean (interactúa con AutenticaciónUltrasonido)
+ enviarMensaje(destinatario: Usuario, contenido: String): void
+ recibirMensaje(mensaje: Mensaje): void
+ agregarReseña(reseña: Reseña): void
+ calcularCalificaciónPromedio(): float
Mensaje
+ mensajeID: int
+ remitente: Usuario
+ destinatario: Usuario
+ contenido: String
+ fechaEnvio: DateTime
+ leído: boolean
+ enviarMensaje(): void
+ marcarComoLeído(): void
HorarioAcadémico
+ horarioID: int
+ usuario: Usuario
+ clases: List<Clase>
+ agregarClase(clase: Clase): void
+ eliminarClase(claseID: int): void
+ obtenerDisponibilidad(): List<TimeSlot>
Clase
+ claseID: int
+ nombre: String
+ díaSemana: Enum (LUNES, MARTES, ...)
+ horaInicio: Time
+ horaFin: Time
+ ubicación: Ubicación
+ obtenerDuración(): Duration
Carrito
+ carritoID: int
+ comprador: Usuario
+ itemsCarrito: List<ItemCarrito>
+ agregarItem(producto: Producto, cantidad: int): void
+ eliminarItem(itemCarritoID: int): void
+ actualizarCantidad(itemCarritoID: int, cantidad: int): void
+ calcularTotal(): float
Recomendador
+ recomendadorID: int
+ usuario: Usuario
+ historialCompras: List<Producto>
+ productosVisitados: List<Producto>
+ preferencias: PreferenciasUsuario
+ contactos: List<Contacto> (para recomendaciones sociales)
+ generarRecomendaciones(): List<Producto>
+ enviarNotificacionesPersonalizadas(): void
+ actualizarModelo(): void (analiza preferencias, historial y contactos)
Ubicación
+ ubicaciónID: int
+ nombreLugar: String
+ latitud: double
+ longitud: double
+ obtenerUbicación(): void
Contacto
+ contactoID: int
+ nombre: String
+ teléfono: String
+ email: String
+ importarContactos(): List<Contacto>
+ sincronizarContactos(): void
Notificación
+ notificaciónID: int
+ usuario: Usuario
+ mensaje: String
+ fechaEnvio: DateTime
+ leído: boolean
+ tipo: Enum (RECOMENDACIÓN, TRANSACCIÓN, PROMOCIÓN, RECORDATORIO)
+ enviarNotificación(): void
+ marcarComoLeída(): void
PreferenciasUsuario
+ preferenciasID: int
+ usuario: Usuario
+ categoríasFavoritas: List<Categoría>
+ palabrasClave: List<String>
+ horasPreferidas: List<Time>
+ actualizarPreferencias(): void
+ agregarCategoríaFavorita(categoría: Categoría): void
+ eliminarCategoríaFavorita(categoría: Categoría): void