Logotipo

Descripción generada automáticamente





Contenido

[**1. Introducción 4**](#_heading=h.gjdgxs)

[**2. Propósito 4**](#_heading=h.30j0zll)

[**3. Alcance 4**](#_heading=h.1fob9te)

[**4. Equipo de Trabajo-Actores del Desarrollo 5**](#_heading=h.3znysh7)

[**5. Recomendaciones de conformidad con esta práctica. 6**](#_heading=h.2et92p0)

[**6. Referencias y estándares aplicables a este documento: 6**](#_heading=h.2s8eyo1)

[**7. Definiciones, acrónimos y abreviaciones. 7**](#_heading=h.17dp8vu)

[**8. DESCRIPCIÓN DE ARQUITECTURA:** 7](#_heading=h.3rdcrjn)

[8.1 VISTAS: 7](#_heading=h.26in1rg)

[8.2 TIPOS DE VISTAS: 7](#_heading=h.lnxbz9)

[8.3 Framework Conceptual 8](#_heading=h.35nkun2)

[8.3.1 Descripción de la arquitectura en contexto 8](#_heading=h.1ksv4uv)

[8.3.1 Actores Usuarios y sus roles: 9](#_heading=h.44sinio)

[8.3.2 Actividades de arquitectura en el ciclo de vida 9](#_heading=h.2jxsxqh)

[8.3.3 Descripciones prácticas de arquitectura. 10](#_heading=h.z337ya)

[8.4 Documentación de la arquitectura 10](#_heading=h.3j2qqm3)

[8.5 Selección de puntos de vista de la arquitectura 11](#_heading=h.1y810tw)

[8.6 Vistas de la arquitectura 12](#_heading=h.4i7ojhp)

[UML-VISTA LÓGICA-DIAGRAMA DE CLASES 13](#_heading=h.1ci93xb)

[UML-VISTA LÓGICA-DIAGRAMA DE BASE DE DATOS 15](#_heading=h.qsh70q)

[UML-VISTA DE PROCESO-DIAGRAMAS DE ACTIVIDAD 20](#_heading=h.1pxezwc)

[1. Inicio: El usuario inicia sesión en la app. 21](#_heading=h.a3eov0eedyfg)

[2. Acceso a la aplicación y escaneo: Muestra la vista de inicio, solicita acceso a la cámara, y escanea productos. Tras escanear, el usuario decide si añadir al carrito o descartar el producto. Al finalizar, se muestra el resumen de la orden. 21](#_heading=h.a3eov0eedyfg)

[3. Método de Pago: El usuario selecciona el método y confirma la compra. 21](#_heading=h.a3eov0eedyfg)

[4. Webpay: Procesa el pago. Si es exitoso, genera el ticket y código QR; si falla, muestra un mensaje de error. 21](#_heading=h.a3eov0eedyfg)

[5. Supervisor: Valida el ticket de compra presentado por el usuario para completar la transacción. 21](#_heading=h.a3eov0eedyfg)

[6. El proceso concluye con la generación de una orden de compra exitosa. 21](#_heading=h.ce0cdk9wev5r)

[UML-VISTA DE DESPLIEGUE 22](#_heading=h.49x2ik5)

[Usabilidad y Calidad de software- Front-end -Vistas Principales 25](#_heading=h.147n2zr)

[Usabilidad y Calidad de software – Vistas específicas de usuario 26](#_heading=h.3o7alnk)

[8.7 Consistencia en la cantidad de vistas de la arquitectura. 27](#_heading=h.23ckvvd)

[8.7.1 DESCRIPCION DE MODULOS 27](#_heading=h.ihv636)

[8.7.2 DESCRIPCIÓN DE COMPONENTES 27](#_heading=h.32hioqz)

[8.7.3 DESCRIPCIÓN DE CONECTORES 28](#_heading=h.1hmsyys)

[**9.1 Acceso a fuentes de información del proyecto** 29](#_heading=h.41mghml)

### Introducción

El presente informe tiene como objetivo describir en detalle la arquitectura de la aplicación móvil a realizar llamada “ScanBuy”, la cual está diseñada para automatizar el proceso de compra en las sucursales de AutoPlanet, permitiendo a los usuarios realizar sus compras de manera autónoma. El sistema propuesto busca mejorar la experiencia de los clientes al agilizar las transacciones y eliminar la necesidad de pasar por cajas físicas, optimizando así el flujo de compra.

El sistema desarrollado será implementado utilizando tecnologías como Ionic y Angular para su despliegue exclusivo en dispositivos Android. A través de esta aplicación, los usuarios podrán escanear productos mediante la cámara de sus teléfonos y procesar el pago directamente desde la aplicación. El sistema también proporcionará una interfaz intuitiva y accesible para todos los usuarios, asegurando que las gestiones de compra se realicen de manera eficiente y organizada, cumpliendo con los estándares internacionales de seguridad y calidad, como el ISO/IEC 27001 para la gestión de la seguridad de la información y el ISO 25010 para la calidad del software.

### Propósito

El propósito de este documento es servir como guía para el diseño e implementación de la arquitectura del sistema a realizar. En él se detallan las decisiones técnicas, los componentes principales y los puntos de integración que permitirán implementar una solución robusta y escalable, capaz de gestionar múltiples comunidades bajo un esquema de administración unificado.

Este informe es esencial para los desarrolladores y stakeholders del proyecto, ya que les proporcionará una visión detallada de la estructura y comportamiento del sistema, así como las relaciones entre las diferentes funcionalidades y componentes. Asimismo, establece las bases para futuras expansiones y mejoras del sistema, considerando factores de disponibilidad, seguridad y usabilidad.

### Alcance

El sistema gestionará las siguientes áreas:

* **Autonomía en la compra de productos**: Los usuarios podrán realizar sus compras de manera autónoma, escaneando los códigos de barra de los productos en una automotora y agregándolos a su carrito de compras a través de la aplicación.
* **Pago integrado dentro de la aplicación**: Los clientes podrán finalizar sus compras y realizar el pago directamente desde la aplicación mediante pasarelas de pago, eliminando la necesidad de pasar por una caja física.
* **Monitoreo en tiempo real por administradores**: Los administradores podrán supervisar las sesiones de compra activas y ver los productos que los usuarios tienen en sus carritos, lo que facilita el control y la seguridad.
* **Generación de reportes de ventas**: Los administradores tendrán la capacidad de generar reportes detallados de las ventas realizadas, lo que les permitirá llevar un registro más eficiente de las transacciones y el rendimiento de la tienda.

### Equipo de Trabajo-Actores del Desarrollo

El desarrollo del sistema requiere la participación de un equipo multidisciplinario con roles claramente definidos para asegurar que el proyecto se lleve a cabo de manera eficiente. A continuación, se presenta una tabla con los principales roles y sus responsabilidades:

| Rol | Responsabilidades |
| --- | --- |
| Cliente-Stakeholder | Definir los requisitos de alto nivel del sistema, establecer las expectativas del proyecto y proporcionar retroalimentación continua para asegurar que el desarrollo cumpla con sus necesidades |
| Product Owner | Definir los requisitos de alto nivel del sistema, establecer las expectativas del proyecto y proporcionar retroalimentación continua para asegurar que el desarrollo cumpla con sus necesidades |
| Desarrolladores Backend | Diseñar, desarrollar y mantener la lógica del servidor y la infraestructura del sistema. Asegurar la integración con la base de datos y las pasarelas de pago, y garantizar la seguridad de las transacciones. |
| Desarrolladores Frontend | Implementar la interfaz de usuario utilizando Ionic y Angular, asegurando que sea intuitiva y accesible para los usuarios. Colaborar con los desarrolladores backend para integrar la funcionalidad y asegurar la experiencia de usuario. |
| DBA | Gestionar la base de datos, asegurando la eficiencia, seguridad y disponibilidad de los datos. |
| Tester/QA | Desarrollar y ejecutar pruebas para identificar errores y garantizar la calidad del sistema. |
| Analista de Requisitos | Desarrollar y ejecutar pruebas para identificar errores y garantizar la calidad del sistema. |
| Administrador Plataformas | Gestionar la infraestructura técnica del proyecto, garantizar la disponibilidad de los entornos de desarrollo, pruebas y producción, y supervisar el despliegue del sistema. |

### Recomendaciones de conformidad con esta práctica.

Para asegurar que la arquitectura del sistema cumpla con los requisitos y estándares establecidos, se recomiendan las siguientes prácticas:

1. Implementar revisiones de arquitectura cada 2 semanas para asegurar que se ajusten a los cambios de requisitos.
2. Utilizar patrones de diseño desacoplados como MVC para modularidad y escalabilidad o por capas.
3. Adoptar metodologías ágiles como Scrum para la entrega continua de mejoras y funcionalidades.

### Referencias y estándares aplicables a este documento:

El documento sigue los siguientes estándares de calidad para asegurar la robustez y escalabilidad del sistema:

* **IEEE 830-1998 ST**: Estandarización de requisitos de software.
* **ISO 9126-2001**: Métricas de calidad del software.
* **Modelo 4+1 de Kruchten**: Modelo estructural basado en múltiples vistas de escenarios.
* **ATAM (Architecture Tradeoff Analysis Method)**: Metodología para evaluar y documentar decisiones arquitectónicas

### Definiciones, acrónimos y abreviaciones.

| Término | Definición |
| --- | --- |
| DAS | Documento de Arquitectura de Software |
| MVC | Modelo-Vista-Controlador, patrón de diseño utilizado para separar la lógica de negocio de la interfaz de usuario |
| UML | Unified Modeling Language, lenguaje gráfico para representar la arquitectura del sistema |
| CRUD | Operaciones básicas de Create, Read, Update, Delete para la gestión de datos. |

### DESCRIPCIÓN DE ARQUITECTURA:

La arquitectura del sistema se presenta mediante el uso de vistas basadas en el Modelo 4+1 de Kruchten, con el fin de proporcionar una visión completa y estructurada del sistema desde diferentes perspectivas. La arquitectura del sistema se presenta mediante el uso de vistas basadas en el Modelo 4+1 de Kruchten, con el fin de proporcionar una visión completa y estructurada del sistema desde diferentes perspectivas.

#### VISTAS:

Cada vista se representará en un diagrama UML para describir la estructura y comportamiento del sistema.

#### TIPOS DE VISTAS:

1. **Vista de Escenarios (Diagrama de Caso de Uso)**: Describe las funcionalidades del sistema y cómo interactúan los diferentes actores con él.
2. **Vista Lógica (Diagrama de Clases)**: Detalla la estructura de datos y las relaciones entre las principales entidades del sistema.
3. **Vista de Procesos (Diagrama de Actividad)**: Muestra el flujo de actividades dentro del sistema y cómo se coordinan los diferentes procesos.
4. **Vista de Despliegue (Diagrama de Componentes)**: Representa la distribución de componentes en la infraestructura física.
5. **Vista Física (Diagrama Topológico)**: Representa la infraestructura de red y la comunicación entre los diferentes servicios del sistema

#### Framework Conceptual

#### Descripción de la arquitectura en contexto

El sistema será diseñado y desarrollado utilizando un enfoque ágil con metodologías como Scrum, lo que permitirá la entrega incremental de funcionalidades, la retroalimentación constante y la capacidad de adaptarse a cambios en los requisitos a lo largo del ciclo de vida del proyecto. Esto asegurará que se puedan abordar necesidades emergentes y maximizar el valor del producto para los usuarios.

El diseño del sistema se basará en la separación de vistas, que ofrecerán diferentes perspectivas de la arquitectura para una comprensión integral. A continuación, se describen las principales vistas:

* **Vista de Escenarios**: Representada mediante diagramas de Casos de Uso, que describen las interacciones del usuario con el sistema y los flujos principales.
* **Vista Lógica**: Incluye el Modelo de Clases, que muestra las entidades y relaciones del sistema, así como el Modelo de Datos que detalla la estructura de la base de datos.
* **Vista de Procesos**: Utiliza Diagramas de Actividad para ilustrar los flujos internos del sistema y las interacciones entre los distintos módulos.
* **Vista de Despliegue**: Se implementará a través de Diagramas de Componentes o Modelos de Capas para reflejar la distribución de la lógica de negocio, la interfaz de usuario y la capa de acceso a datos.
* **Vista Física**: Utiliza Diagramas Topológicos o de Servicios para mostrar la distribución del sistema en el entorno físico, como servidores, redes y otros componentes de infraestructura.

Cada una de estas vistas está interconectada para garantizar la coherencia y la alineación con los requisitos de negocio. No hay ninguna vista separada de una implementación específica, sino que todas contribuyen a una visión holística de la arquitectura del sistema descrita en este documento.

Estas vistas se han creado utilizando el Lenguaje de Modelado Unificado (UML) en su versión 2.0 y han sido desarrolladas empleando herramientas de modelado estándar como StarUML, UMLetino y Lucidchart para facilitar la creación de diagramas visuales que respalden la documentación arquitectónica.

El estilo arquitectónico se centra en una combinación de patrones de diseño orientados a servicios y arquitectura por capas, en donde cada módulo del sistema se corresponde con un conjunto específico de casos de uso que resuelven los procesos definidos para la operación.

Estos procesos corresponden a:

* Autenticación y Gestión de Usuarios
* Gestión de Productos
* Gestión de Categorías
* Escaneo de Productos
* Gestión de Carrito de compras
* Gestión de Pago
* Generación de reportes

#### Actores Usuarios y sus roles:

Este documento representa la identificación de Actores/Usuarios Stakeholders y sus roles a partir de la interpretación de los casos de uso del Negocio asociados.

| ACTOR | ROL / CARGO | FUNCION |
| --- | --- | --- |
| Cliente | Usuario | Es el principal consumidor de la aplicación. Este usuario accede a la plataforma para realizar compras de productos dentro de una automotora de forma autónoma, utilizando la funcionalidad de escaneo de códigos de barras. El cliente tiene un rol limitado a interactuar con la interfaz de compra y realizar transacciones. |
| Supervisor | Admin | Es responsable de verificar los vouchers generados a través de la aplicación. Tiene acceso a herramientas de monitoreo que le permiten visualizar los carritos de compra y usuarios activos. |
| Gerente | SuperAdmin | Es responsable de verificar los vouchers generados a través de la aplicación. Tiene acceso a herramientas de monitoreo que le permiten visualizar los carritos de compra y usuarios activos. |

#### 8.3.2 Actividades de arquitectura en el ciclo de vida

El desarrollo de la arquitectura del sistema ScanBuy se abordó utilizando un ciclo de vida **iterativo incremental** basado en **Scrum**. Esta metodología fue seleccionada porque permite realizar entregas parciales y frecuentes de software funcional, facilitando la validación temprana de los requisitos por parte de los stakeholders y el Product Owner con la adaptación a cambios en el entorno de negocio.

El ciclo de vida se organizó en **sprints** de 2 a 4 semanas, con cada iteración centrada en la construcción y mejora de un módulo específico del sistema (por ejemplo, el módulo de autenticación y gestión de usuarios o el de gestión de productos). Al final de cada sprint, se realiza una revisión detallada de los resultados para asegurar la alineación con las expectativas del cliente y se priorizan las tareas para el siguiente sprint.

#### 8.3.3 Descripciones prácticas de arquitectura.

Durante el proceso de desarrollo, la arquitectura del sistema fue escalada desde un **escenario principal de caso de uso** hasta un diseño modular a través de los siguientes pasos:

1. **Análisis de Requerimientos**: Se analizaron los requisitos iniciales del sistema para identificar los casos de uso principales, tales como "Administrar Gastos Comunes", "Reservar Espacios Comunes" y "Generar Reportes de Cobros".
2. **Definición de la Vista Lógica**: A partir del escenario principal, se identificaron las entidades y relaciones claves en un diagrama de clases UML, representando el núcleo del sistema.
3. **Modelo de Datos**: Se desarrolló un diagrama de base de datos para definir la estructura de la información, manteniendo la coherencia con la vista lógica.
4. **Vista de Procesos**: Se modelaron los flujos de las actividades y transacciones principales del sistema mediante diagramas de actividad.
5. **Despliegue**: Se definió la infraestructura física y lógica para alojar el sistema en la nube, detallando los servicios necesarios, como balanceadores de carga, servidores de aplicaciones y bases de datos distribuidas.
6. **Definición de Planes de pruebas y técnicas de pruebas testing**.

#### Documentación de la arquitectura

* **ISO/IEC 25010**: Se utilizó para evaluar la calidad del sistema en términos de funcionalidad, seguridad y eficiencia.
* **IEEE 1471-2000**: Proporcionó la guía para la representación de vistas arquitectónicas y la identificación de stakeholders.
* **Modelo 4+1 de Kruchten**: Estructura la arquitectura en vistas lógicas, de desarrollo, de procesos y física, facilitando la visualización desde múltiples perspectivas.
* **Planes de prueba**: (Declarar los tipos de pruebas a realizar)

#### Selección de puntos de vista de la arquitectura

| VISTAS | UML | Cantidad |
| --- | --- | --- |
| Escenario Principal | Diagrama de Caso de uso | 1 (2-3 los necesarios) |
| Vista Lógica | Diagrama de Clases  MER | 1  1 |
| Vista de Proceso | Diagrama de Actividad:  Nombre 1.  Nombre 2  Nombre 3. | 1 (2-3, los necesarios) |
| Vista Física | Diagrama Topológico de Servicios | 1 |
| Vista de Despliegue | Diagrama de componentes  Diagrama de Capas (MVC) | 1  1 |

#### Vistas de la arquitectura

| UML-VISTA DE ESCENARIO |
| --- |
| Diagrama de Caso de uso extendido PROCESO DE COMPRA |
|  |

**Iniciar sesión**: Permite a usuarios y administradores acceder al sistema con sus credenciales, verificando el rol para acceder a las funcionalidades correspondientes.  
**Escanear Código de Barras:** Permite a los clientes escanear productos con la cámara del móvil para añadirlos al carrito de compras.  
**Gestión de Perfil de Usuario:** Permite a los usuarios editar su información personal como nombre, email y contraseña, asegurando la validez de los datos.  
**Carrito de Compras:** Facilita a los clientes agregar, eliminar o modificar productos en el carrito, mostrando el precio total antes del pago.  
**Gestión de Método de Pago:** Permite a los usuarios registrar y seleccionar métodos de pago como tarjetas o pasarelas de pago.  
**Gestionar y Monitorear Órdenes y Usuarios:** Los administradores monitorean en tiempo real las compras y carritos activos, además de gestionar usuarios y verificar vouchers.  
**Generar Reportes:** Administradores generan reportes de ventas, órdenes y usuarios, para analizar la actividad y ajustar la gestión de inventario.

| UML-VISTA LÓGICA-DIAGRAMA DE CLASES |
| --- |
| DIAGRAMA DE CLASES SCANBUY |
| A screenshot of a computer program  Description automatically generated |

El **Diagrama de Clases** para el sistema ScanBuy muestra las clases involucradas, sus atributos, métodos y relaciones. Este diagrama modela la estructura y el comportamiento del sistema de manera orientada a objetos.

**Elementos Principales:**

* **Usuario**
  + **Atributos: id\_usuario, nombre, apellido, email**
  + **Métodos:**
    - **iniciarSesion()**
    - **cerrarSesion()**
    - **modificarPerfil()**
    - **modificarPassword()**
* **Cliente** (hereda de Usuario)
  + **Métodos:**
    - **registrarse()**
    - **recuperarPassword()**
    - **getCategoriaCliente()**
* **Administrador** (hereda de Usuario)
  + **Métodos:**
    - **obtenerUsuariosActivos()**
* **SuperAdmin** (hereda de Usuario)
  + **Métodos:**
    - **obtenerUsuarios()**
    - **eliminarUsuario()**
* **Producto**
  + **Atributos: id\_producto, codigo\_barras, nombre, marca, annio, precio**
  + **Métodos:**
    - **agregarProducto()**
    - **obtenerProductos()**
    - **obtenerProductoEspecifico()**
    - **modificarProducto()**
    - **eliminarProducto()**
    - **escanearProducto()**
    - **getCategoria()**
* **Carrito**
  + **Atributos: id\_carrito, estado, cantidad, total**
  + **Métodos:**
    - **obtenerTotal()**
* **Ref\_Carrito**
  + **Atributos: id\_refcarrito, preciounitario, cantidad, total**
  + **Métodos:**
    - **limpiarCarrito()**
    - **actualizarCantidadCarrito()**
    - **getUsuario()**
    - **getProducto()**
    - **getCarrito()**
* **Compra**
  + **Atributos: id\_compra, estado, cantidad, total**
  + **Métodos:**
    - **generarBoleta()**
    - **obtenerCompras()**
    - **realizarCompra()**
* **Categoría**
  + **Atributos: nombre, descripcion**
  + **Métodos:**
    - **agregarCategoria()**
    - **obtenerCategorias()**
    - **modificarCategoria()**
    - **eliminarCategoria()**
* **Categoria\_Cliente**
  + **Atributos: id\_categoriacliente, descripcion**
  + **Métodos:**
    - **modificarCategoria()**
* **Voucher**
  + **Atributos: id\_voucher, cantidad, total, fecha\_emision, estado**
  + **Métodos:**
    - **generarVoucher()**
    - **getCompra()**

**Relaciones Principales:**

* Un Cliente tiene una Categoría\_Cliente
* Un Cliente tiene un Carrito
* Un Cliente puede tener múltiples Compras
* Un Carrito puede tener múltiples Ref\_Carrito
* Un Producto pertenece a una Categoría
* Un Administrador gestiona Productos y Categorías
* Una Compra genera un Voucher

| UML-VISTA LÓGICA-DIAGRAMA DE BASE DE DATOS |
| --- |
| MER SCANBUY |
|  |

El **Modelo Entidad-Relación (MER)** para ScanBuy representa las entidades involucradas, sus atributos y las relaciones que existen entre ellas. Este modelo permite visualizar cómo se organiza la información relacionada con la administración del condominio.

A continuación se presenta el diccionario de datos que le corresponde**:**

| Nombre tabla | CATEGORIA | | |
| --- | --- | --- | --- |
| Descripción | Tabla que almacena las categorías de los productos. | | |
| Atributos de la tabla | | | |
| Nombre de campo | Tipo de dato | Tamaño | Descripción |
| id\_categoria | int |  | Identificador único de la categoría de un producto |
| nombre | varchar |  | Nombre de la categoría |
| Campos claves | PK: id\_categoria | | |
| Relaciones | Relación 1:N con la tabla PRODUCTO. | | |

| Nombre tabla | PRODUCTO | | |
| --- | --- | --- | --- |
| Descripción | Contiene la información de los productos disponibles para su escaneo. | | |
| Atributos de la tabla | | | |
| Nombre de campo | Tipo de dato | Tamaño | Descripción |
| id\_producto | int |  | Identificador único de un producto |
| nombre | varchar |  | Nombre del producto |
| marca | varchar |  | Marca del producto |
| annio | int |  | Año de fabricación del producto |
| precio | int |  | Precio unitario del producto |
| id\_categoria | int |  | Identificador de la categoría del producto |
| Campos claves | PK: id\_producto  FK: id\_categoria (hace referencia a CATEGORIA) | | |
| Relaciones | Relación muchos a uno con la tabla CATEGORIA.  Relación 1:N con las tablas REF\_CARRITO y REF\_COMPRA. | | |

| Nombre tabla | CATEGORIA\_CLIENTE | | |
| --- | --- | --- | --- |
| Descripción | Clasifica a los clientes en diferentes categorías. | | |
| Atributos de la tabla | | | |
| Nombre de campo | Tipo de dato | Tamaño | Descripción |
| id\_categoriacliente | int |  | Identificador único de categoría cliente |
| descripcion | varchar |  | Descripción de la categoría |
| Campos claves | PK: id\_categoriacliente | | |
| Relaciones | Relación uno a muchos con la tabla USUARIO. | | |

| Nombre tabla | USUARIO | | |
| --- | --- | --- | --- |
| Descripción | Almacena información de los usuarios del sistema. | | |
| Atributos de la tabla | | | |
| Nombre de campo | Tipo de dato | Tamaño | Descripción |
| id\_usuario | int |  | Identificador único de usuario |
| nombre | varchar |  | Nombre del usuario |
| apellido | varchar |  | Apellido del usuario |
| email | varchar |  | Correo electrónico del usuario |
| id\_rol | int |  | Identificador del rol del usuario |
| id\_categoríacliente | int |  | Identificador de la categoría del cliente |
| Campos claves | PK: id\_usuario  FK: id\_rol (hace referencia a ROL), id\_categoriacliente (hace referencia a CATEGORIA\_CLIENTE) | | |
| Relaciones | Relación N:1 con la tabla ROL.   * Relación N:1 con la tabla CATEGORIA\_CLIENTE. * Relación 1:N con las tablas REF\_CARRITO y REF\_COMPRA. | | |

| Nombre tabla | ROL | | |
| --- | --- | --- | --- |
| Descripción | Define los roles disponibles para los usuarios, en este caso serían Cliente, Administrador y Superadministrador. | | |
| Atributos de la tabla | | | |
| Nombre de campo | Tipo de dato | Tamaño | Descripción |
| id\_rol | int |  | Identificador único del rol |
| nombre | varchar |  | Nombre del rol |
| Campos claves | PK: id\_rol | | |
| Relaciones | Relación 1:N con la tabla USUARIO. | | |

| Nombre tabla | CARRITO | | |
| --- | --- | --- | --- |
| Descripción | Almacena el carrito de compras de los usuarios. | | |
| Atributos de la tabla | | | |
| Nombre de campo | Tipo de dato | Tamaño | Descripción |
| id\_carrito | int |  | Identificador único del carrito |
| estado | varchar |  | Estado del carrito |
| cantidad | int |  | Cantidad total de productos |
| total | int |  | Total de la compra en el carrito |
| Campos claves | PK: id\_carrito | | |
| Relaciones | Relación 1:N con la tabla REF\_CARRITO. | | |

| Nombre tabla | REF\_CARRITO | | |
| --- | --- | --- | --- |
| Descripción | Representa los productos seleccionados en el carrito de compras. | | |
| Atributos de la tabla | | | |
| Nombre de campo | Tipo de dato | Tamaño | Descripción |
| id\_refcarrito | int |  | Identificador único de referencia del carrito |
| precio\_unitario | int |  | Precio unitario del producto |
| cantidad | int |  | Cantidad de unidades del producto |
| total | int |  | Total de la referencia del carrito |
| id\_usuario | int |  | Identificador del usuario |
| id\_producto | int |  | Identificador del producto |
| id\_carrito | int |  | Identificador del carrito |
| Campos claves | PK: id\_refcarrito  FK: id\_usuario (hace referencia a USUARIO), id\_producto (hace referencia a PRODUCTO), id\_carrito (hace referencia a CARRITO) | | |
| Relaciones | Relación N:1 con las tablas USUARIO, PRODUCTO, y CARRITO. | | |

| Nombre tabla | COMPRA | | |
| --- | --- | --- | --- |
| Descripción | Almacena la información de las compras finalizadas. | | |
| Atributos de la tabla | | | |
| Nombre de campo | Tipo de dato | Tamaño | Descripción |
| id\_compra | int |  | Identificador único de compra |
| estado | varchar |  | Estado de la compra |
| cantidad | int |  | Cantidad total de productos |
| total | int |  | Total de la compra |
| Campos claves | PK: id\_compra | | |
| Relaciones | Relación 1:N con las tablas REF\_COMPRA y VOUCHER. | | |

| Nombre tabla | VOUCHER | | |
| --- | --- | --- | --- |
| Descripción | Es el voucher final que hace referencia a la compra realizada | | |
| Atributos de la tabla | | | |
| Nombre de campo | Tipo de dato | Tamaño | Descripción |
| Id\_Voucher | int |  | Id del voucher |
| cantidad | int |  | cantidad de productos comprados |
| Total | int |  | total de la compra realizada |
| Fecha\_emision | timestamp |  | fecha de la compra |
| estado | varchar |  | estado actual de la compra (pagado, checkeado) |
| id\_compra | int |  | id de la compra que hace referencia |
| Campos claves | PK: Id\_voucher  FK: id\_compra = hace referencia a la compra que se acaba de realizar en la cual se encuentran todos los productos que están relacionados a la boleta | | |
| Relaciones | Relación 1:1 con la tabla “COMPRA” | | |

| Nombre tabla | REF\_COMPRA | | |
| --- | --- | --- | --- |
| Descripción | Representa los productos que van en la compra en tiempo real | | |
| Atributos de la tabla | | | |
| Nombre de campo | Tipo de dato | Tamaño | Descripción |
| id\_refcompra | int |  | Identificador único de referencia del compra |
| precio\_unitario | int |  | Precio unitario de la referencia de la compra |
| cantidad | int |  | Cantidad de productos |
| total | int |  | Total a pagar |
| id\_usuario | int |  | Id del usuario que está relacionado para obtener la referencia |
| id\_producto | int |  | Id del producto relacionado para obtener la referencia |
| id\_compra | int |  | Id de la compra relacionada para obtener la referencia |
| Campos claves | PK: id\_refcompra  FK: id\_usuario  FK: id\_compra  FK: id\_producto | | |
| Relaciones | Relación N:1 con tabla COMPRA  Relación 1:1 con tabla REF\_CARRITO | | |

| UML-VISTA DE PROCESO-DIAGRAMA DE ACTIVIDAD |
| --- |
| DIAGRAMA DE ACTIVIDAD PROCESO DE COMPRA |
|  |

El **diagrama de actividad** representa el flujo de actividades que se llevan a cabo en un proceso específico, en este caso, el proceso de compra del usuario. Este diagrama es útil para visualizar los pasos secuenciales y las decisiones que un residente debe tomar al realizar un pago.

**Componentes**:

#### Inicio: El usuario inicia sesión en la app.

#### Acceso a la aplicación y escaneo: Muestra la vista de inicio, solicita acceso a la cámara, y escanea productos. Tras escanear, el usuario decide si añadir al carrito o descartar el producto. Al finalizar, se muestra el resumen de la orden.

#### Método de Pago: El usuario selecciona el método y confirma la compra.

#### Webpay: Procesa el pago. Si es exitoso, genera el ticket y código QR; si falla, muestra un mensaje de error.

#### Supervisor: Valida el ticket de compra presentado por el usuario para completar la transacción.

#### El proceso concluye con la generación de una orden de compra exitosa.

#### 

| UML-VISTA DE DESPLIEGUE |
| --- |
| DIAGRAMA DE DESPLIEGUE SCANBUY |
|  |

**Servidor de Aplicaciones**:

* **Aplicación Móvil Scanbuy**: Proporciona una interfaz amigable para todos los actores del sistema (Usuarios (cliente), administrador, superadministrador). Permite la consulta de productos, compras, gestión de pagos y generación de reportes.

**Servidor de Base de Datos**:

* **Base de Datos de Scanbuy**: Almacena toda la información relevante, Almacena toda la información relevante, incluyendo a los usuarios, productos, carritos en real-time, compras realizadas Este servidor es fundamental para que la aplicación funcione correctamente, ya que procesa las consultas y actualizaciones de datos.

**Servidor de Correo**:

* **Sistema de Notificación**: Se encarga de enviar correos electrónicos automáticos a los usuarios para confirmar el correo electrónico o restablecer la contraseña en caso de necesitarlo

| UML-VISTA FÍSICA – DIAGRAMA INFRAESTRUCTURA |
| --- |
| DIAGRAMA INFRAESTRUCTURA SCANBUY |
|  |

**Centro de Datos Principal - ScanBuy**: Es el núcleo central del sistema que contiene dos componentes principales:

**1. Proveedores y Red**:

* **ISP**: Proveedor de Servicios de Internet que suministra la conectividad
* **Router Principal**: Gestiona el tráfico de red entre todos los componentes
* **Firewall de Seguridad**: Protege la infraestructura contra amenazas externas

**2. Servidores**:

* **Servidor de Backup**: Encargado de realizar y mantener las copias de seguridad del sistema
* **Supabase**: Base de datos en la nube basada en PostgreSQL donde se almacena toda la información del sistema
* **Servidor de Correo**: Maneja las notificaciones y comunicaciones por email
* **Servicio de Aplicaciones**: Aloja la aplicación web de ScanBuy y conecta todos los servicios

**Estación de AutoPlanet**: Representa la red local de cada sucursal, que incluye:

* **Estación de Trabajo: Supervisor**
* **Estación de Trabajo: Gerente**
* **Red WiFi Cliente**
* **Cámara de Seguridad**
* **Sistema de Control de Acceso**

| Usabilidad y Calidad de software- Front-end -Vistas Principales |
| --- |
| Vistas principales del sistema – Vista Home Principal de entrada |
|  |

La **Vista Front-End** del sistema de ScanBuy ha sido diseñada para ofrecer una experiencia de usuario intuitiva, moderna y fácil de navegar. La interfaz está dividida en diferentes secciones con menús y paneles dinámicos que se adaptan a cada tipo de usuario, brindando acceso a funcionalidades específicas según su rol (Administrador, Superadmin, Cliente). Se presenta un diseño limpio y ordenado que resalta las opciones más importantes, permitiendo a los usuarios interactuar con el sistema de manera eficiente

**Estilo Moderno y Profesional**: Colores vivos y elementos gráficos que se destacan sin distraer, utilizando una paleta de colores asociada a la identidad visual de ScanBuy.

**Interactividad y Usabilidad**: Uso de gráficos animados, transiciones suaves y elementos interactivos que facilitan la navegación.

**Iconografía Clara**: Uso de iconos intuitivos para representar cada módulo y función, ayudando a los usuarios a ubicar rápidamente la opción deseada

| Usabilidad y Calidad de software – Vistas específicas de usuario |
| --- |
| Vistas principales del sistema – Vista Principal de Usuario a escritorio de trabajo principal de su componente de sistema |
|  |

#### Consistencia en la cantidad de vistas de la arquitectura.

Se garantiza que cada vista esté alineada con los requisitos funcionales y no funcionales del sistema, evitando redundancias y manteniendo la coherencia en toda la arquitectura

#### DESCRIPCION DE COMPONENTES

Ejemplos:

| Nombre del componente | Descripción | Artefactos incluidos |
| --- | --- | --- |
| Autenticación y Gestión de Usuarios | Maneja el registro, inicio de sesión, y gestión de perfiles de usuarios | Vista de inicio de sesión, Vista de registro de usuario, vista de perfil, vista de recuperación de contraseña. |
| Gestión de Productos | Permite crear, editar y eliminar productos en la base de datos. | Vista de gestión de productos |
| Gestión de Categorías | Facilita la creación y organización de categorías para clasificar los productos. | Vista de gestión de categorias |
| Escaneo de Productos | Implementa la funcionalidad de escanear códigos de barras para obtener información del producto y agregarlo al carrito de compras. | Vista de agregar producto al carrito mediante escaneo, Vista de agregar producto a la BD mediante escaneo |
| Gestión de Carrito de Compras | Permite agregar, eliminar y modificar productos antes de proceder al pago. | Vista de carrito de compras |
| Gestión de Pago | Procesa los pagos de los usuarios de forma segura a través de Webpay | Vista de pago (WebPay), Vista de voucher |
| Generación de reportes | Genera reportes de ventas y otras estadísticas importantes para administradores y gerentes. | Vista de reportes |

#### DESCRIPCIÓN DE ARTEFACTOS

| Nombre del Artefacto | Descripción | Componente Relacionado |
| --- | --- | --- |
| Vista de inicio de sesión |  |  |
| Vista de registro de usuario |  |  |
| Vista de perfil |  |  |
| Vista de recuperación de contraseña |  |  |
| Vista de gestión de productos |  |  |
| Vista de gestión de categorías |  |  |
| Vista de gestión de categorías |  |  |
| Vista de agregar producto al carrito mediante escaneo |  |  |
| Vista de agregar producto a la BD mediante escaneo |  |  |
| Vista de carrito de compras |  |  |
| Vista de pago (WebPay) |  |  |
| Vista de voucher |  |  |
| Vista de reportes |  |  |

* + - 1. **Arquitectura lógica**
      2. Performances

El sistema ScanBuy está diseñado para gestionar un volumen medio de usuarios en simultáneo (estimado en cientos entre todas las sucursales), sin comprometer el rendimiento, especialmente en momentos de alta demanda, como horas punta de compras. Para asegurar tiempos de respuesta rápidos (inferiores a 2 segundos por transacción), se implementarán las siguientes estrategias:

1. **Escalabilidad en el Backend**: *Supabase* se usará como backend para la base de datos y autenticación, aprovechando su infraestructura serverless para escalar automáticamente en función de la demanda. Esto permite añadir recursos de forma dinámica y flexible en momentos de alta concurrencia, sin necesidad de administración manual.
2. **Optimización de Peticiones a la Base de Datos**: Se implementarán índices y consultas optimizadas en la base de datos de *Supabase* (basada en PostgreSQL). Esto incluye el uso de índices en campos de búsqueda frecuentes como el código de barras del producto y optimización en el manejo de lecturas y escrituras.
3. **Balanceo de Carga en la API**: Se configurará un balanceador de carga en *Supabase* (o en la infraestructura de API Gateway) para distribuir el tráfico equitativamente entre los endpoints, evitando la saturación de un único punto de acceso.
4. **Caché y Almacenamiento Local**: La aplicación móvil almacenará temporalmente en caché información crítica, como datos de usuario y productos escaneados, lo cual reducirá la cantidad de peticiones a *Supabase* y mejorará el tiempo de respuesta en situaciones de baja conectividad.
   * + 1. Criterios de Calidad

**Funcionalidad**

* **Correctitud**: Cada módulo del sistema (autenticación, carrito, pagos) debe realizar correctamente su función específica.
* **Adecuación**: Las funcionalidades deben alinearse con las necesidades del cliente (como el flujo sencillo para escanear productos y proseguir al pago directamente).

**Fiabilidad**

* **Disponibilidad**: El sistema debe estar disponible el 99% de las veces. Si bien no se le pondrá fin al tipo de compra tradicional en las sucursales de Autoplanet, es ideal que la aplicación se muestre disponible en todo momento para que los usuarios la puedan utilizar y de a poco se vaya haciendo adopción del nuevo flujo de compra hasta que se convierta en el estándar.
* **Tolerancia a Fallos**: Implementación de mecanismos de recuperación ante fallos en los servicios backend de Supabase y bases de datos.

**Portabilidad**

* **Adaptabilidad**: La aplicación está diseñada específicamente para Android, pero debe garantizarse que el código pueda ser adaptado en el futuro a iOS con mínimas modificaciones, utilizando las capacidades híbridas de Ionic.
* **Instalabilidad**: El proceso de instalación de la aplicación en dispositivos Android debe ser sencillo, a través de la Play Store.

**Escalabilidad**

* **Crecimiento en Demanda**: Debe poder soportar un aumento del 200% en la cantidad de usuarios activos simultáneamente en un plazo de 2 años, sin comprometer la estabilidad.
  + - 1. Detalles de la implementación

La implementación del sistema se basará en una arquitectura de microservicios, utilizando las siguientes tecnologías:

* **Backend**: Supabase para la gestión de la base de datos, autenticación, almacenamiento, servicios en tiempo real, etc.
* **Frontend**: HTML, CSS, Ionic y Angular.
* **Base de Datos**: PostgreSQL dentro del entorno de Supabase para la gestión eficiente de datos.
* **Infraestructura**: AWS utilizando contenedores Docker para una gestión flexible y escalable de los microservicios.
  + - 1. Lenguajes y plataformas

Las tecnologías que se utilizarán en el desarrollo d el sistema incluyen:

* **Ionic**
* **Angular**
* **Node.js**
* **PostgreSQL en Supabase**
* **Typescript**
* **Android Studio**
* **Visual Studio Code**

### Acceso a fuentes de información del proyecto

Para acceder a la carpeta de evidencias acumuladas y otros documentos relevantes del proyecto, puede utilizar el siguiente enlace:  
  
Insertar link: [Carpeta de Evidencia del Proyecto](https://drive.google.com/drive/folders/1zE5gFhWYLF_TTF4MvaJOGdvCZZGXNxFp?usp=sharing)  
  
Esta carpeta incluye documentación técnica, diagramas, pruebas de rendimiento y cualquier otra evidencia relacionada con el desarrollo y la implementación del sistema