Marketplace de Tecnología con estética Apple Human Interface Guidelines (HIG)

> Documento de especificación inicial (versión 1.0). Alcance: conceptual + diseño funcional + base de datos + wireframes.

---

1) Contexto general del proyecto

1.1 Descripción del problema

El comercio de productos tecnológicos (hardware, periféricos, accesorios, smart home, computación, audio, gaming) está fragmentado en múltiples tiendas con fichas técnicas incompletas, mala experiencia móvil y poca confianza para compras entre particulares o pymes. Los usuarios requieren **comparabilidad**, **compatibilidad** entre productos, **información técnica verificada** y **una experiencia fluida y confiable**. Los vendedores pequeños necesitan una vitrina verticalmente especializada y con reputación que simplifique pagos, despacho y atención postventa.

1.2 Solución propuesta

Un **marketplace vertical de tecnología** con diseño y experiencia inspirada en **Apple Human Interface Guidelines** (claridad, deferencia, profundidad), que ofrezca: - **Catálogo curado** por categorías técnicas (GPU, CPU, monitores, teclados, routers, SSD, etc.). - **Fichas ricas**: especificaciones normalizadas, compatibilidades, comparador, reseñas verificadas. - **Proceso de compra minimal y rápido** (carrito → checkout en 1–2 pasos, Apple Pay/Stripe/Transbank, envío). - **Confianza**: verificación de vendedores, SLA de despacho, garantía, resolución de disputas. - **Paneles dedicados** para vendedores (gestión de stock, precios, ofertas) y para administración (curación, categorías, moderación, métricas).

1.3 Principios de diseño (Apple HIG aplicado al web)

- **Claridad:** tipografía legible (sistemas tipo SF/SF Pro), jerarquía visual consistente, espacios generosos, microcopys claros.
- **Deferencia:** interfaz limpia que cede protagonismo al contenido (fotos, specs, compatibilidades), uso sobrio del color.
- **Profundidad:** orden visual, transiciones sutiles, señalización de navegación, foco y contexto.
- **Accesibilidad:** contraste adecuado, tamaños táctiles, soporte teclado/lector de pantalla.
- **Consistencia:** patrones comunes (cards, listas, filtros, tablas) y estados (hover/focus/loading/empty/error) definidos.

1.4 Alcance del MVP

- Autenticación (comprador, vendedor, admin).
- Catálogo por categorías con filtros.
- Búsqueda con sugerencias.
- Ficha de producto con variantes y reseñas.
- Carrito y checkout (pago, envío, confirmación).
- Panel de vendedor (productos, stock, órdenes).
- Panel admin (categorías, moderación, métricas básicas).

- Correo/Notificaciones transaccionales.

Fuera de alcance inicial: subastas, marketplace B2B, logística propia, devoluciones avanzadas multi-seller en una misma orden (se manejarán por ítem), bundles complejos.

---

2) Tecnologías del proyecto (propuesta)

2.1 Frontend

- **Framework:** Next.js 14+ (App Router), **TypeScript**.
- **Estilos:** Tailwind CSS con **design tokens** propios inspirados en HIG (espaciado 4/8/12/16, radios suaves, sombras sutiles, tipografía sistema).
- **UI Kit:** componentes propios + utilidades de accesibilidad (Radix primitives) respetando HIG.
- **Formularios y validación:** React Hook Form + Zod.
- **Estado y datos:** React Query/Server Actions (cuando aplique), cache por ruta.
- **SEO/Performance:** SSR/ISR, imágenes optimizadas (Next/Image), fuentes del sistema.
- **Accesibilidad:** aria-* y pruebas con axe.

2.2 Backend

- **Runtime:** Node.js 20+, **NestJS** (estructura modular) o Express con arquitectura limpia.
- **API:** REST con OpenAPI (Swagger). (Opcional GraphQL para agregaciones).
- **Autenticación:** NextAuth (OAuth/Email), **Sign in with Apple** opcional, 2FA (TOTP) futuro.
- **Pagos: Stripe** (global) y/o **Transbank Webpay** (Chile) según segmento.
- **Almacenamiento:** PostgreSQL 15+ con **Prisma ORM**; archivos en S3/R2.
- **Búsqueda:** Postgres Full-Text; (futuro: Meilisearch/Algolia).
- **Colas/eventos:** Redis/Upstash + BullMQ (emails, indexación, webhooks de pago).

2.3 Infraestructura y DevOps

- **Hosting:** Vercel (frontend), Fly.io/Railway/Render (backend) o contenedores en AWS/GCP.
- **CI/CD:** GitHub Actions (lint, test, build, deploy).
- **Observabilidad:** Sentry (FE/BE), Log aggregation, Health checks.
- **Seguridad:** Helmet, CORS estricto, Rate limiting, CSP, escaneo de dependencias.
- **Analítica:** PostHog/GA4.

---

3) Requerimientos

3.1 Requerimientos funcionales

Actor Visitante 1. Navegar por categorías y ver listado de productos. 2. Buscar productos con autocompletado/sugerencias. 3. Ver ficha de producto (fotos, specs, precio, stock, reseñas). 4. Ver compatibilidades/relacionados. 5. Crear cuenta e iniciar sesión.

Actor Comprador 6. Añadir productos/variantes al carrito (multi-seller). 7. Editar cantidades y eliminar ítems del carrito. 8. Ingresar dirección y método de envío. 9. Pagar (Stripe/Transbank) y ver confirmación. 10. Recibir correos/notificaciones de orden. 11. Ver historial de órdenes y estado (en preparación, enviado, entregado). 12. Dejar reseñas y calificaciones de productos. 13. Abrir tickets de soporte/disputa.

Actor Vendedor 14. Solicitar/crear perfil de vendedor (verificación). 15. Crear/editar productos, variantes, precios y stock. 16. Gestionar órdenes (preparación, despacho, número de seguimiento). 17. Ver panel con métricas de ventas y reseñas. 18. Gestionar políticas de envío y tiempos.

Actor Admin 19. Gestionar categorías/atributos técnicos. 20. Curar/moderar publicaciones y reseñas. 21. Gestionar usuarios/vendedores (estado, verificación). 22. Ver reportes (ventas, conversión, top categorías).

Actor Sistema 23. Enviar notificaciones transaccionales (pedido, envío, entrega). 24. Procesar webhooks de pago y actualizar estados. 25. Recalcular índices/búsquedas y cachés.

3.2 Requerimientos no funcionales

- **Usabilidad:** apego a HIG; Tareas críticas $\leq$ 3 clics.
- **Rendimiento:** LCP $\leq$ 2.5s en 4G; API P95 $\leq$ 300ms.
- **Disponibilidad:** $\geq$ 99.9% mensual.
- **Escalabilidad:** horizontal en FE/BE; colas para trabajos pesados.
- **Seguridad:** OWASP Top 10; cifrado en tránsito (TLS 1.2+); hashing de contraseñas (Argon2/BCrypt); protección CSRF/XSS/SQLi; roles RBAC.
- **Privacidad:** minimización de datos; cumplimiento legal local; logs de acceso.
- **Accesibilidad:** WCAG 2.2 AA.
- **Compatibilidad:** navegadores modernos; mobile-first responsive.
- **Mantenibilidad:** cobertura de tests $\geq$ 70%; lint/format; convenciones de commits.

---

4) WEB — Diagrama de casos de uso (visión general)

```
Actores: [Visitante] [Comprador] [Vendedor] [Admin]

[Visitante] ──> (Explorar catálogo)
[Visitante] ──> (Buscar productos)
[Visitante] ──> (Ver ficha de producto)
[Visitante] ──> (Crear cuenta / Iniciar sesión)

[Comprador] ──> (Gestionar carrito) ──> (Checkout) ──> (Pagar)
[Comprador] ──> (Rastrear orden)
[Comprador] ──> (Publicar reseña)
[Comprador] ──> (Abrir ticket soporte)

[Vendedor] ──> (Gestionar productos)
[Vendedor] ──> (Gestionar stock)
[Vendedor] ──> (Gestionar órdenes)
[Vendedor] ──> (Ver métricas)

[Admin] ──> (Gestionar categorías/atributos)
[Admin] ──> (Moderación)
[Admin] ──> (Reportes)
```

```
(Sistema) <── webhooks pago / emails / indexación
```

Alternativa (Mermaid pseudo-UML):

```
flowchart TD
    V[Visitante] -->|Explorar| C(Catálogo)
    V -->|Buscar| B(Búsqueda)
    V -->|Ficha| P(Producto)
    V -->|Cuenta| A(Auth)

    C1[Comprador] --> K(Carrito)
    K --> CH(Checkout)
    CH --> PG(Pago)
    C1 --> TR(Rastreo)
    C1 --> RV(Reseña)
    C1 --> ST(Soporte)

    S1[Vendedor] --> GP(Gestión Productos)
    S1 --> GS(Gestión Stock)
    S1 --> GO(Gestión Órdenes)
    S1 --> MX(Métricas)

    AD[Admin] --> GA(Gestión Atributos)
    AD --> MD(Moderación)
    AD --> RP(Reportes)

    SYS((Sistema)) -.-> PG
    SYS -.-> EM[Emails]
    SYS -.-> IX[Indexación]
```

---

5) WEB — Mockups (low-fi) de pantallas clave

5.1 Home / Catálogo

```
┌──────────────────────────────────────────────────────────────┐
│  Logo      Buscar… [⌘K]        Categorías   Ofertas  Cuenta  │
├──────────────────────────────────────────────────────────────┤
│  «Hero» limpio con 1 mensaje y CTA                           │
│  Filtros: Precio ▾  Marca ▾  Compatibilidad ▾  Envío ▾       │
│                                                              │
│  ▢ Producto Card      ▢ Producto Card      ▢ Producto Card   │
│  Foto grande          Foto grande          Foto grande       │
│  Título · Precio      Título · Precio      Título · Precio   │
│  ★★★★☆ (120)          ★★★★☆ (88)           ★★★☆☆ (23)        │
│                                                              │
```

```
│  Paginación simple / Cargar más                                  │
└──────────────────────────────────────────────────────────────────┘
```

5.2 Ficha de producto

```
┌──────────────────────────────────────────────────────────────────┐
│  ← Volver     Nombre Producto                                    │
├──────────────────────────────────────────────────────────────────┤
│ [Galería]            |  Precio   $XXX.XXX                      │
│                      |  Variante: [Capacidad ▾][Color ▾]       │
│                      |  Stock: En bodega ✓                     │
│                      |  Botón: Añadir al carrito               │
│                      |  Beneficios: garantía, despacho, etc    │
├──────────────────────────────────────────────────────────────────┤
│ Especificaciones (tabla limpia, legible; acordeón por secc.)    │
│ Compatibilidad (chips/puertos/so)                                │
│ Reseñas verificadas                                              │
└──────────────────────────────────────────────────────────────────┘
```

5.3 Carrito y Checkout

```
┌──────────────────────────────────────┐   ┌──────────────────────┐
│ Carrito                              │   │ Resumen              │
│  - Ítem (foto, nombre, variante)     │   │ Subtotal             │
│  - Cantidad [– 1 +]  Precio          │   │ Envío                │
│  - Eliminar                          │   │ Total                │
│                                      │   │ Botón: Pagar         │
└──────────────────────────────────────┘   └──────────────────────┘

Checkout (1–2 pasos): Dirección → Pago (Apple Pay/Stripe) → Confirmación
```

5.4 Panel de Vendedor — Productos

```
┌──────────────────────────────────────────────────────────────────┐
│  Sidebar: Dashboard · Productos · Órdenes · Reseñas · Ajustes    │
├──────────────────────────────────────────────────────────────────┤
│ [ + Nuevo producto ]  Buscar…                                    │
│ Tabla:  Nombre | SKU | Precio | Stock | Estado | Acciones        │
│         Teclado…| T123| 49.990 |  22   | Activo | Editar ⋮      │
└──────────────────────────────────────────────────────────────────┘
```

---

6) BD — Modelo Entidad-Relación (ER)

```
erDiagram
    USER ||--o{ ADDRESS : has
    USER ||--o{ ORDER : places
    USER ||--o{ REVIEW : writes
    USER ||--o| SELLER_PROFILE : owns
    SELLER_PROFILE ||--o{ PRODUCT : lists
    CATEGORY ||--o{ PRODUCT : groups
    PRODUCT ||--o{ PRODUCT_VARIANT : has
    PRODUCT_VARIANT ||--o{ INVENTORY : tracks
    ORDER ||--|{ ORDER_ITEM : contains
    PRODUCT_VARIANT ||--o{ ORDER_ITEM : referenced
    ORDER ||--o| PAYMENT : has
    PRODUCT ||--o{ REVIEW : receives
    USER ||--o{ CART : has
    CART ||--|{ CART_ITEM : contains
    PRODUCT_VARIANT ||--o{ CART_ITEM : referenced
    USER ||--o{ SUPPORT_TICKET : opens
```

Entidades mínimas cubiertas: **User, Product, Order, Payment** (y más para un modelo robusto).

---

7) Modelo relacional (tablas, claves y tipos)

Tipos sugeridos PostgreSQL. `PK` = clave primaria, `FK` = clave foránea, `UK` = única.

users - `id` UUID PK - `email` text UK not null - `password_hash` text null (si OAuth, puede ser null) - `name` text - `role` text check in ('buyer','seller','admin') default 'buyer' - `created_at` timestamptz default now()

seller_profiles - `id` UUID PK - `user_id` UUID FK → users(id) UK not null - `display_name` text not null - `status` text check in ('pending','verified','rejected') default 'pending' - `created_at` timestamptz default now()

addresses - `id` UUID PK - `user_id` UUID FK → users(id) not null - `line1` text not null; `line2` text; `city` text; `region` text; `zip` text; `country` text not null - `is_default` boolean default false

categories - `id` UUID PK - `name` text UK not null - `parent_id` UUID FK → categories(id) null

products - `id` UUID PK - `seller_id` UUID FK → seller_profiles(id) not null - `category_id` UUID FK → categories(id) not null - `title` text not null - `slug` text UK not null - `description` text - `specs_json` jsonb (atributos técnicos normalizados) - `status` text check in ('draft','active','paused') default 'draft' - `created_at` timestamptz default now()

product_variants - `id` UUID PK - `product_id` UUID FK → products(id) not null - `sku` text UK not null - `price_cents` integer not null check (price_cents>=0) - `currency` text default 'CLP' - `attributes_json` jsonb (p.ej., color, capacidad)

inventory - `id` UUID PK - `variant_id` UUID FK → product_variants(id) not null - `stock` integer not null check (stock>=0) - `reserved` integer not null default 0 check (reserved>=0)

carts - `id` UUID PK - `user_id` UUID FK → users(id) not null - `created_at` timestamptz default now()

cart_items - `id` UUID PK - `cart_id` UUID FK → carts(id) on delete cascade - `variant_id` UUID FK → product_variants(id) - `qty` integer check (qty>0) - UK(cart_id, variant_id)

orders - `id` UUID PK - `user_id` UUID FK → users(id) not null - `status` text check in ('pending','paid','preparing','shipped','delivered','cancelled') default 'pending' - `total_cents` integer not null - `currency` text default 'CLP' - `shipping_address_id` UUID FK → addresses(id) - `created_at` timestamptz default now()

order_items - `id` UUID PK - `order_id` UUID FK → orders(id) on delete cascade - `variant_id` UUID FK → product_variants(id) - `seller_id` UUID FK → seller_profiles(id) - `unit_price_cents` integer not null - `qty` integer not null check (qty>0)

payments - `id` UUID PK - `order_id` UUID FK → orders(id) UK not null - `provider` text check in ('stripe','webpay') not null - `provider_ref` text not null - `amount_cents` integer not null - `status` text check in ('requires_action','succeeded','failed','refunded') - `paid_at` timestamptz

reviews - `id` UUID PK - `user_id` UUID FK → users(id) - `product_id` UUID FK → products(id) - `rating` smallint check (rating between 1 and 5) - `comment` text - `created_at` timestamptz default now() - UK(user_id, product_id)

support_tickets - `id` UUID PK - `user_id` UUID FK → users(id) - `order_id` UUID FK → orders(id) - `subject` text not null - `status` text check in ('open','in_progress','resolved','closed') default 'open' - `created_at` timestamptz default now()

---

8) Diccionario de datos (columnas clave)

> Formato: **Tabla.Campo** — Tipo — Nulabilidad — Descripción

users - id — UUID — NN — Identificador del usuario. - email — text — NN, UK — Email único. - password_hash — text — N — Hash para login clásico. - role — text — NN — Rol del usuario (buyer/seller/admin). - created_at — timestamptz — NN — Fecha de creación.

seller_profiles - id — UUID — NN — Id de perfil de vendedor. - user_id — UUID — NN, UK, FK→users — Dueño del perfil. - display_name — text — NN — Nombre público de tienda. - status — text — NN — Estado de verificación.

categories - id — UUID — NN — Id de categoría. - name — text — NN, UK — Nombre único. - parent_id — UUID — N, FK→categories — Jerarquía opcional.

products - id — UUID — NN — Id del producto. - seller_id — UUID — NN, FK→seller_profiles — Vendedor. - category_id — UUID — NN, FK→categories — Categoría. - title — text — NN — Título visible. - slug — text — NN, UK — Identificador para URL. - description — text — N — Descripción extendida. - specs_json — jsonb — N — Atributos técnicos (clave/valor). - status — text — NN — Estado de publicación.

product_variants - id — UUID — NN — Id variante. - product_id — UUID — NN, FK→products — Producto padre. - sku — text — NN, UK — Código único. - price_cents — int — NN — Precio en centavos. - currency — text — NN — Moneda (CLP por defecto). - attributes_json — jsonb — N — Color, capacidad, etc.

inventory - id — UUID — NN — Id inventario. - variant_id — UUID — NN, FK→product_variants — Variante. - stock — int — NN — Unidades disponibles. - reserved — int — NN — Unidades reservadas.

orders - id — UUID — NN — Id de orden. - user_id — UUID — NN, FK→users — Comprador. - status — text — NN — Estado del flujo. - total_cents — int — NN — Total de la orden. - shipping_address_id — UUID — N, FK→addresses — Dirección de envío. - created_at — timestamptz — NN — Fecha de compra.

order_items - id — UUID — NN — Ítem de orden. - order_id — UUID — NN, FK→orders — Orden. - variant_id — UUID — NN, FK→product_variants — Variante. - seller_id — UUID — NN, FK→seller_profiles — Vendedor del ítem. - unit_price_cents — int — NN — Precio unitario al momento de compra. - qty — int — NN — Cantidad.

payments - id — UUID — NN — Pago. - order_id — UUID — NN, UK, FK→orders — Relación 1–1 con la orden. - provider — text — NN — stripe/webpay. - provider_ref — text — NN — ID transacción del proveedor. - amount_cents — int — NN — Monto pagado. - status — text — NN — Estado del pago. - paid_at — timestamptz — N — Fecha de éxito.

reviews - id — UUID — NN — Reseña. - user_id — UUID — NN, FK→users — Autor. - product_id — UUID — NN, FK→products — Producto. - rating — smallint — NN — 1..5. - comment — text — N — Comentario. - created_at — timestamptz — NN — Fecha.

addresses - id — UUID — NN — Dirección. - user_id — UUID — NN, FK→users — Dueño. - line1/line2/city/region/zip/country — text — según — Componentes postales. - is_default — boolean — NN — Marca principal.

support_tickets - id — UUID — NN — Ticket. - user_id — UUID — NN, FK→users — Creador. - order_id — UUID — N, FK→orders — Orden relacionada. - subject — text — NN — Asunto. - status — text — NN — open/in_progress/resolved/closed.

---

9) Restricciones de integridad referencial y supuestos

9.1 Integridad referencial (FKs y acciones)

- `seller_profiles.user_id` **UK+FK** garantiza un perfil por usuario vendedor (1–1). **ON DELETE RESTRICT**.
- `products.seller_id` **FK** → `seller_profiles.id`. **ON DELETE RESTRICT** (no borrar vendedor con productos activos).
- `products.category_id` **FK** → `categories.id`. **ON DELETE SET NULL** o **RESTRICT** (según política de curación).
- `product_variants.product_id` **FK** → `products.id`. **ON DELETE CASCADE** (si se elimina el producto, caen sus variantes).
- `inventory.variant_id` **FK** → `product_variants.id`. **ON DELETE CASCADE**`.
- `carts.user_id` **FK** → `users.id`. **ON DELETE CASCADE**.
- `cart_items.cart_id` **FK** → `carts.id` **ON DELETE CASCADE**; `cart_items.variant_id` **FK** → `product_variants.id` **ON DELETE RESTRICT**.
- `orders.user_id` **FK** → `users.id` **ON DELETE RESTRICT**; `orders.shipping_address_id` **FK** → `addresses.id` **ON DELETE SET NULL**.
- `order_items.order_id` **FK** → `orders.id` **ON DELETE CASCADE**; `order_items.variant_id` **FK** → `product_variants.id` **ON DELETE RESTRICT**; `order_items.seller_id` **FK** → `seller_profiles.id` **ON DELETE RESTRICT**.
- `payments.order_id` **UK+FK** → `orders.id` **ON DELETE CASCADE** (relación 1–1).
- `reviews.user_id` **FK** → `users.id`; `reviews.product_id` **FK** → `products.id`; **ON DELETE CASCADE** para limpieza si se borra un usuario/ producto.
- `addresses.user_id` **FK** → `users.id` **ON DELETE CASCADE**.
- `support_tickets.user_id` **FK** → `users.id` **ON DELETE SET NULL** (opcional); `support_tickets.order_id` **FK** → `orders.id` **ON DELETE SET NULL**`.

9.2 Restricciones/Checks y unicidades

- `users.email` **UK**.
- `seller_profiles.user_id` **UK** (un vendedor por usuario).
- `products.slug` **UK**; `product_variants.sku` **UK**.
- `cart_items` **UK(cart_id, variant_id)** (no duplicar variantes en el mismo carrito).
- `payments.order_id` **UK** (máximo un pago consolidado por orden; reintentos se reflejan en `status`).
- Checks: `price_cents>=0`, `stock>=0`, `qty>0`, `rating between 1 and 5`.

9.3 Supuestos

- El marketplace es **multi-seller por orden**; el despacho y la facturación pueden ser por ítem.
- Las reseñas solo pueden hacerlas usuarios que **compraron** el producto (enforzado a nivel de aplicación o trigger).
- El inventario se descuenta al **pago exitoso**; se reserva al crear la orden (campo `reserved`).
- Múltiples pasarelas de pago pueden coexistir; se prioriza Stripe para MVP internacional y Webpay para Chile.
- Categorías pueden ser jerárquicas (self-FK `parent_id`).

---

10) Notas de estilo HIG → tokens UI (sugerido)

- **Tipografía:** `font-sans` sistema (SF/SF Pro/Inter fallback), tamaños: 12/14/16/20/24/32.
- **Espaciado:** base 8 px; contenedores 16/24/32; secciones 48–64.
- **Color:** neutros (textos #111/#444), acento único (azul o cian), estados (verde éxito, rojo error) muy sobrios.
- **Componentes:** Cards con bordes suaves (`rounded-2xl`) y sombras ligeras, listas densas en paneles.
- **Feedback:** skeletons, spinners pequeños, toasts minimalistas.

---

11) Próximos pasos sugeridos

1. Alinear alcance MVP y pasarelas (Stripe/Webpay).
2. Definir taxonomía técnica y atributos normalizados por categoría.
3. Diseñar UI high-fi (Figma) basada en estos wireframes.
4. Preparar backlog y milestones; iniciar implementación FE/BE y esquema DB con Prisma.

Fin del documento (v1.0).