Apphive Mega Tutorial

None

None

None

Table of contents

1. Introducción 4
1.1 Objetivos del curso 4
2. Glosario esencial (ES → EN) 5
3. Fundamentos 6
3.1 Fundamentos 6
3.2 Data Manager (APIs REST y colecciones) 7
3.3 Navegación y rutas 8
4. UI 9
4.1 UI avanzada 9
4.2 Estilos y temas 10
4.3 Componentes reusables (design system) 11
5. Backend y APIs 12
5.1 APIs REST (backend propio) 12
5.2 Geo & Mapas (proxy backend) 13
6. Comunicación y pagos 14
6.1 Notificaciones push (intro) 14
6.2 Pagos (intro) 15
6.3 Stripe avanzado — Payment Intents, 3DS, tarjetas y suscripciones 16
6.4 Mercado Pago — Marketplace (split payments) 17
6.5 Pagos — buenas prácticas 18
7. Operación 19
7.1 Build & Deploy 19
7.2 Estado global y sincronización 20
7.3 Offline-first & Retry 21
7.4 Roles y permisos (RBAC) 22
7.5 Logging y auditoría 23
7.6 Modelado de datos 24
7.7 Performance y UX 25
7.8 Analítica de eventos 26
7.9 Internacionalización (i18n) 27
7.10 Testing & QA 28
7.11 Seguridad y hardening 29
7.12 Versionado y lanzamientos 30
7.13 SLO/SLA/SLI + alertas (burn rate) 31
7.14 Guardrails de costo 32

7.15 Catálogo de integraciones (facade) 33
7.16 Runbooks de incidentes 34
8. Índice maestro 35

1. Introducción

Bienvenido al **Apphive Mega Tutorial** (edición web). Fecha: 2025-10-02

Este portal está pensado para **neófitos totales**. Vas a aprender *clic por clic*, con capturas sugeridas, listas de verificación y diagramas ASCII neutros.

> 🔥 **Cómo usar este sitio**
>
> 1. Lee de arriba hacia abajo.
> 2. Usa la **búsqueda** (arriba a la derecha) para encontrar funciones o eventos.
> 3. Cuando veas *bloques de pasos*, síguelos literalmente.

1.1 Objetivos del curso

- Entender el ecosistema Apphive.
- Construir una app completa desde cero.
- Adoptar buenas prácticas para producción.

2. Glosario esencial (ES → EN)

- **Página (Page)**: Pantalla o vista.
- **App Process**: Bloque de lógica reusable (flow/acciones).
- **Data Manager**: Conector de datos / APIs.
- **Binding (enlace)**: Conectar un valor a un widget.
- **Request/Response**: Petición/Respuesta de API.
- **Rate limit**: Límite de peticiones.
- **Idempotencia (Idempotency)**: Reintentar sin duplicar efectos.
- **Tenant**: Cliente/espacio lógico aislado.
- **SLO/SLI/SLA**: Confiabilidad (objetivo/indicador/acuerdo).

3. Fundamentos

3.1 Fundamentos

3.1.1 Conceptos básicos

- **Page**: pantalla o vista principal.
- **App Process**: lógica reusable.
- **Data Manager**: acceso a datos/APIs.
- **Estado**: valores que cambian con la interacción (inputs, toggles...).

3.1.2 Primeros pasos (clic por clic)

1. Crea una **Page** de inicio.
2. Arrastra un **Button** y cambia su texto a `Comenzar`.
3. Crea un **App Process** llamado `goToWelcome`.
4. Enlaza el botón a `goToWelcome` → acción `Navigate` a la página `welcome`.
5. **Guarda** y **Prueba**.

3.2 Data Manager (APIs REST y colecciones)

3.2.1 Objetivo

Conectar tu app a datos externos sin exponer secretos.

3.2.2 Flujo (ASCII)

```
[Apphive UI] --(request)--> [Backend propio] --(secret/API Key)--> [Proveedor externo]
```

3.2.3 Pasos

1. En tu backend crea un endpoint `/api/items` (GET/POST).
2. En Apphive, en **Data Manager**, agrega una **fuente** llamada `Items` → URL de tu backend.
3. Define **acciones**: `listItems`, `createItem`.
4. En UI, enlaza un **Repeater** a `listItems` y un **Form** a `createItem`.
5. Maneja estados: **loading**, **error**, **empty**.

3.3 Navegación y rutas

3.3.1 Tipos

- **Navigate**: cambiar de página.
- **Back**: volver atrás.
- **Deep link**: abrir una sección con URL.

3.3.2 Buenas prácticas

- Nombra páginas con **snake-case** (ej: `order_detail`).
- Centraliza rutas en un **App Process** `goTo(name, params)`.

4. UI

4.1 UI avanzada

4.1.1 Validaciones

- Reglas por campo y por formulario.
- Marca campos inválidos en vivo.

4.1.2 Componentes reutilizables

- Crea un **App Process** `inputWithLabel(props)`.
- Reutiliza estilos básicos: padding, tipografías, espaciados.

4.1.3 Accesibilidad

- Contraste mínimo AA.
- Labels claros y estados de foco visibles.

4.2 Estilos y temas

- Define una paleta base (primario, éxito, alerta).
- Usa **variables de color** para mantener consistencia.
- Soporta **modo claro/oscuro** desde diseño.

4.3 Componentes reusables (design system)

4.3.1 Patrón

1. Crea una Page `kitchen_sink` para prototipos.
2. Define variantes de Button, Input, Card.
3. Documenta props esperadas y ejemplos.

5. Backend y APIs

5.1 APIs REST (backend propio)

5.1.1 Reglas básicas

- La app **no** guarda secretos.
- El backend firma/valida requests a terceros.
- Maneja **idempotencia** con `Idempotency-Key`.

5.1.2 Endpoints ejemplo (pseudocódigo)

```
GET  /orders?tenantId=...
POST /orders     (valida esquema, crea, audita)
GET  /orders/:id
```

5.2 Geo & Mapas (proxy backend)

5.2.1 Por qué proxy

Para no exponer API Keys.

5.2.2 Rutas de proxy

```
GET /maps/autocomplete?q=
GET /maps/details?id=
GET /maps/route?orig=...&dest=...
```

6. Comunicación y pagos

6.1 Notificaciones push (intro)

6.1.1 Conceptos

- **Token de dispositivo** por usuario.
- **Tópicos** para envíos segmentados.

6.1.2 Flujo mínimo

1. Registrar token en backend al iniciar sesión.
2. Enviar push desde backend según eventos.

6.2 Pagos (intro)

- Cobro directo con proveedor (Stripe/Mercado Pago).
- Webhooks para estado definitivo.
- Reembolsos y conciliación.

6.3 Stripe avanzado — Payment Intents, 3DS, tarjetas y suscripciones

6.3.1 Diagrama neutral

```
Apphive -> POST /payments/create-intent
<- clientSecret
Apphive -> confirm (3DS si aplica)
Webhook -> payment_intent.succeeded/failed
```

6.3.2 Pasos (clic por clic)

1. Backend crea **PaymentIntent**.
2. Apphive confirma y muestra loader.
3. Webhook actualiza la orden.
4. Guardar tarjeta: `customer + setup intent`.
5. Suscripciones: `priceId`, `trial`, `invoice.payment_*`.

6.4 Mercado Pago — Marketplace (split payments)

6.4.1 Flujo

```
Seller OAuth -> guardas token
Compra -> preference con application_fee + collector_id del seller
Webhook -> confirma pago, registra fee
```

6.5 Pagos — buenas prácticas

- Idempotencia en cobros y reembolsos.
- Logs y auditoría de transacciones.
- Manejo de fraudes/reintentos.

7. Operación

7.1 Build & Deploy

- Reglas para publicar versiones.
- Checklists previos (QA, accesibilidad, rendimiento).
- Estrategia de rollback simple.

7.2 Estado global y sincronización

- Fuente única de verdad por pantalla.
- Eventos para actualizar listas tras crear/editar.
- Evita estados duplicados.

7.3 Offline-first & Retry

- Cola local de operaciones (POST/PUT).
- Reintentos con backoff exponencial.
- Resolución de conflictos básica.

7.4 Roles y permisos (RBAC)

- Roles: admin, manager, user.
- Reglas de visibilidad/edición por rol.
- Verificación en **backend** y **UI**.

7.5 Logging y auditoría

- Log de acciones clave con `requestId` / `correlationId`.
- **Export** para investigación y soporte.

7.6 Modelado de datos

- Entidades principales, relaciones y claves.
- Campos auditables: `createdAt`, `createdBy`, `tenantId`.

7.7 Performance y UX

- Paginación y cargas perezosas.
- Caché de catálogos.
- Medición de p95 y p99.

7.8 Analítica de eventos

- Define eventos `track(name, props)`.
- Embudos y cohortes.
- Métricas accionables.

7.9 Internacionalización (i18n)

- Estructura de traducciones por módulo.
- Detección de locale y fallback.

7.10 Testing & QA

- Pruebas de flujo crítico.
- Listas de verificación previas a release.

7.11 Seguridad y hardening

- Validar entrada en backend.
- Evitar secretos en el cliente.
- Límites y protección contra abuso.

7.12 Versionado y lanzamientos

- SemVer para cambios.
- Flags para activación progresiva.

7.13 SLO/SLA/SLI + alertas (burn rate)

- Define SLI: disponibilidad API, p95 checkout, éxito notifs.
- Alertas por **burn rate** (rápidas y lentas).

7.14 Guardrails de costo

- Métricas proxy: requests, almacenamiento, jobs, notifs.
- Presupuestos por tenant/módulo con alertas y throttling.

7.15 Catálogo de integraciones (facade)

7.15.1 Endpoints plantilla

```
/facade/email    /facade/sms
/facade/maps     /facade/payments
/facade/storage
```

- Scopes, rate-limit, logs por proveedor.

7.16 Runbooks de incidentes

- SEV1..SEV4.
- Triage (Commander, Scribe).
- Comunicación y postmortem.

8. Índice maestro

- v0.1–0.9: Fundamentos a publicación avanzada.
- v1.0: cierre principiantes.
- v1.1–v1.7: intermedio/avanzado e integraciones.