METODOLOGÍA QA · CÓDICE DE REFERENCIA

El Códice de Nomenclatura

Cada artefacto de prueba en este framework tiene una forma de nombre exacta. Esta es la única fuente de verdad — desde un test case en el TMS hasta un componente KATA, una rama, una carpeta. Aprende la forma una vez; lee la intención de cualquier artefacto de un vistazo.

- 7 CAPAS
- 30+ ARTEFACTOS
- 12 CONVENCIONES RATIFICADAS
- 0 CONTRADICCIONES

Vive en **.claude/skills/agentic-qa-core/** — el skill base global.

#naming #conventions #single-source-of-truth

ANTES DE LAS FORMAS — POR QUÉ EXISTE ESTE SISTEMA

¿Por qué un códice de nombres?

Un nombre no es decoración: es el primer dato de cada artefacto. Cuando cada nombre sigue una forma estricta, su **capa**, su **alcance** y su **intención** se leen sin abrir nada — por una persona o por una máquina.

Legible de un vistazo

El prefijo y el primer token dicen qué es y a qué capa pertenece antes de leer la frase.

Trazable

La clave Jira viaja en el nombre, así Story ↔ test ↔ código ↔ rama quedan atados.

Legible por máquina

Lint, sync y el `kata-manifest` dependen de formas predecibles para automatizar.

Anti-duplicación

Una sola forma por artefacto evita dos nombres para la misma cosa.

Aprende la forma una vez; lee la intención de cualquier artefacto para siempre.

Conceptos base

Todo el códice usa estos términos. Léelos una vez; el resto del deck asume que ya los conoces.

TMS

Test Management System — dónde viven tests, planes y corridas (aquí, Jira + Xray).

Epic / Story / PBI

Epic = feature; Story = unidad de trabajo con criterios de aceptación; PBI = el ítem del backlog.

Test Case (TC)

Un caso individual — un comportamiento afirmado. La hoja del árbol.

Test Set

Agrupación lógica de TCs por feature. No planifica ni ejecuta — solo agrupa.

Test Plan

Qué se va a probar y su alcance (Story / Feature / Sprint / Producto).

Test Execution

Una corrida de ese plan — el registro de qué pasó o falló.

Precondition (PRC)

El estado requerido antes de ejecutar (ej. usuario autenticado).

ATC

Acceptance Test Case — el método KATA que implementa un TC en código.

KATA

La arquitectura de pruebas en capas (L1–L4) del framework. Ver la capa CODE.

Fixture

Inyección de dependencias de KATA — entrega api/ui/test ya listos al test.

Xray

El plugin de TMS sobre Jira que aporta Test, Test Set, Test Plan, Test Execution.

Modalidad

Cómo guarda el proyecto los TCs: Xray (issues dedicados) o Jira-nativo (campos).

LA CLAVE DE SIGLAS — CADA ACRÓNIMO DEL CÓDICE, EXPANDIDO

Acrónimos de un vistazo

Cada **Plan (P)** se empareja con sus **Results (R)** en la misma altitud — lee cada fila de lado a lado.

ALTITUD	PLAN (P)	RESULTS (R)
Producto	**MTP** — Master Test Plan	—
Feature	**FTP** — Feature Test Plan	**FTR** — Feature Test Results
Sprint	**STP** — Sprint Test Plan	**STR** — Sprint Test Results
Story	**ATP** — Acceptance Test Plan	**ATR** — Acceptance Test Results

SIGLA	SIGNIFICA	CAPA
TC	Test Case	CASE
ATC	Acceptance Test Case	CODE
TS	Test Set	GROUP
PRC	Precondition	CONTAINER
PBI	Product Backlog Item	Jira / FS

Primera letra = altitud: **M**aster · **F**eature · **S**print · **A**cceptance (Story). **Última letra**: **P** = Plan (qué se probará) · **R** = **Results** (el paraguas: la ejecución y sus runs). Un "run" = una iteración de la ejecución.

#naming #conventions #single-source-of-truth

Siete Capas de Nomenclatura

La **capa** de un nombre te dice qué tipo de cosa es antes de leer una sola palabra. Cada capa tiene su color en este códice.

CASE	El Test Case (TC) individual — lo que se afirma	`should <outcome>`
GROUP	El contenedor de casos — la feature bajo prueba	`Validate <feature>`
CONTAINER	Vasijas de planificación y ejecución — ATP/ATR/FTP/STP + Test Set/PRC	`{ACRONYM}: {scope}: …`
CODE	KATA — componentes, ATCs, archivos, tags	`@atc · {Resource}Api`
JIRA	Estados de workflow, labels, títulos de issues de calidad	`Epic: Component: …`
GIT	Ramas, commits, títulos de PR	`type(KEY): …`
FILESYSTEM	Árbol PBI, archivos spec/plan, carpetas KATA	`EPIC-<KEY>-<slug>`

GROUP vs CONTAINER — ambos "contienen", pero **GROUP** solo agrupa casos por feature (`describe` / Test Set), mientras **CONTAINER** son las vasijas de planificación y ejecución (Plan / Run). GROUP organiza; CONTAINER planifica + ejecuta.

CÓMO ENCAJAN LAS CAPAS — UNA IMAGEN ANTES DE LOS DETALLES

El modelo mental

Cada capa anida en la siguiente. Una Story se planifica, agrupa sus casos, se ejecuta, y el código la espeja — todo atado por la misma clave Jira.

```
STORY PROJ-101 — la unidad de trabajo
 ├─ ATP: PROJ-101: … PLAN: qué se probará · CONTAINER
 ├─ TS: Validate login GROUP: agrupa los casos
 |   ├─ PROJ-101: TC1: should grant access… CASE
 |   └─ PROJ-101: TC2: should reject login… CASE
 ├─ ATR: PROJ-101: Story Testing RESULTS: qué pasó/falló · CONTAINER
 └─ @atc('PROJ-101') loginWithValidCredentials() CODE espeja el caso
```

- MISMA CLAVE JIRA EN TODO
- PLAN → RUN
- GROUP CONTIENE CASES
- CODE ESPEJA CASE VÍA @ATC

#naming #conventions #single-source-of-truth

Anatomía del título de un Test Case

{US_ID}: TC#: should <expected outcome> *[<connector> <condition>] [given <precondition>]*

- **{US_ID}** — clave de la User Story, siempre
- **TC#** — secuencial (TC1, TC2…)
- **should** — fijo, verbo primero
- **<outcome>** — verbo + objeto afirmado
- **[connector]** — when / if · opcional
- **[given]** — alcance / permiso · opcional

EJEMPLOS

PROJ-101: TC1: should grant access when credentials are valid

PROJ-101: TC2: should reject login if password is incorrect

PROJ-101: TC3: should cap input when exceeding 50 chars

PROJ-101: TC4: should block checkout when cart is empty given a logged-in user

RECHAZAR — vago, no por la palabra "should"

~~Login test~~ — sin ID, sin TC#, sin outcome

~~Login - error~~ — demasiado vago

~~TC1: Test form~~ — falta el prefijo US_ID

~~should work correctly~~ — sin outcome concreto

Regla de mayúsculas — should va en minúscula porque **continúa** el prefijo {US_ID}: TC#: (frase corrida). Validate va en mayúscula porque **inicia** el summary del grupo. La capa decide el casing.

tms-conventions.md · jira-test-management.md · tms-architecture.md

#naming #conventions #single-source-of-truth

should vs Validate

CASE · LA HOJA

should <outcome>

Un comportamiento concreto, afirmado. Lo que pasa o falla.

```
should grant access when credentials valid
```

Aplica a: TC del TMS · intención del método ATC · test() · Gherkin Scenario

GROUP · LA RAMA

Validate <feature>

El conjunto de casos de una capacidad. Nombra la caja, no un caso.

```
Validate user login
```

Aplica a: summary de Xray Test Set · describe() en código

Juntos en código

```
describe('PROJ-101: Validate user login', () => {
  test('PROJ-101: should grant access when credentials are valid')
  test('PROJ-101: should reject login when password is wrong')
})
```

Agrupar los casos

describe() en código

```
'{TICKET-ID}: Validate {feature}'
```

'UPEX-411: Validate discount codes'

automation-standards.md

Test Set · TS

```
TS: {EPIC-KEY|module}: Validate {feature}
```

TS: GX-101: Validate credit card payment

La membresía es un link — nunca el prefijo del TC.

Test Plan · ATP (Story)

```
ATP: {STORY-KEY}: {story title}
```

ATP: PROJ-123: Apply discount at checkout

Test Plan · FTP (Feature)

```
FTP: {EPIC-KEY}: {feature}
```

FTP: PROJ-42: Checkout & Payments

Test Plan · STP (Sprint)

```
STP: Sprint#{N}: Regression
```

STP: Sprint#30: Regression

#naming #conventions #single-source-of-truth

tms-conventions.md §3 · jira-test-management.md §5 · planning-ladder

CAPA · CONTAINER — VASIJAS DE EJECUCIÓN Y SETUP

Vasijas de ejecución y setup

Vasija (ing. *vessel*) = el subconjunto de artefactos que **contiene** una corrida o un plan. Todo en el códice es un artefacto; la vasija es la que se ejecuta o se planifica — por eso no la llamamos solo "artefacto".

ATR · Acceptance Test Results (Story Testing · issue Test Execution)

```
ATR: {STORY-KEY}: Story Testing
```

ATR: GX-123: Story Testing

STR · Sprint Test Results (Regression Testing · issue Test Execution)

```
STR: Sprint#{N}: Regression Testing
```

STR: Sprint#30: Regression Testing

ReTest (re-test de bug)

```
ReTest: {BUG-KEY}: {summary}
```

ReTest: GX-202: Wrong error on invalid password

Precondition · PRC (título = el estado)

```
PRC: {COMPONENT}: {required state}
```

PRC: Payment: Authenticated user with a saved card

El título indica el **estado**; los pasos de setup viven en el contenido.

tms-conventions.md §3 · xray-platform.md · planning-ladder

#naming #conventions #single-source-of-truth

La Escalera de Planificación

Cada Plan y cada Run lleva un prefijo acrónimo de 3 letras, así la altitud y el plan-vs-run se leen en el primer token. Cuatro Epics de proceso QA le dan a cada tipo de artefacto un único hogar de gobierno.

- QA MASTER TEST PLAN · TODOS LOS TEST PLANS
- QA TEST REPOSITORY · TODOS LOS TESTS
- QA TEST ARTIFACTS · TODOS LOS RUNS + PRC + TEST SETS
- QA DEFECT MANAGEMENT · TODOS LOS BUG/DEFECT/IMPROVEMENT

PRODUCT	MTP — Master Test Plan	Epic QA Master Test Plan
FEATURE	FTP → FTR — "Feature Testing", ≥1 run/sprint	{EPIC-KEY}
SPRINT	STP → STR — "Regression Testing", 1/sprint	Sprint#{N}
STORY	ATP → ATR — "Story Testing", 1 run	{STORY-KEY}

ALTITUD	PLAN	RESULTS	PATRÓN
Product	MTP	—	`Epic QA Master Test Plan (+ master-test-plan.md local)`
Feature	FTP	FTR	`FTP/FTR: {EPIC-KEY}: … — término FTR "Feature Testing", ≥1 run/sprint`
Sprint	STP	STR	`STP/STR: Sprint#{N}: … — término STR "Regression Testing", 1/sprint`
Story	ATP	ATR	`ATP/ATR: {STORY-KEY}: … — término ATR "Story Testing", 1 run`

Items sobre campos — cada Plan es un issue Test Plan, cada Run es un issue Test Execution (ambas modalidades); el custom field de la Story es sólo un fallback. 3 ejes: **parent** = epic QA · **link** = alcance · **components** = módulo.

#naming #conventions #single-source-of-truth

CAPA · CODE

Nombres de código KATA

Componentes, ATCs, archivos, fixtures, tags — los nombres que viven en TypeScript. La clave del decorador ata cada línea de vuelta a Jira.

KATA = *Component Action Test Architecture* — no es solo naming: es la estrategia arquitectónica del framework (inyección de dependencias, capas L1–L4, fixtures y nomenclatura). Aquí cubrimos solo sus **nombres**; la arquitectura completa vive en `kata-architecture.md`.

#naming #conventions #single-source-of-truth

ATC, decorador, archivo, test()

Nombre de método ATC

```
{verb}{Resource}{Scenario} // camelCase
```

loginWithValidCredentials()

getUserWithNonExistentId()

Describe acción + outcome esperado. Nunca un solo click.

Decorador @atc

```
@atc('{JIRA_KEY}')
```

@atc('PROJ-101')

✗ sin template literal: @atc(`TC-${id}`)

Archivo de test

```
{verb}{Feature}.test.ts
```

applyDiscount.test.ts

authenticateUser.test.ts

El verbo = acción del usuario (apply, create) — no "verify/check/test".

Bloque test() CASE

```
'{TICKET-ID}: should {behavior} when {cond}'
```

'UPEX-411: should apply percentage discount when code is valid'

automation-standards.md · SKILL.md · kata-manifest.json

CAPA · CODE — CLASES KATA (PASCALCASE, ARCHIVO = CLASE)

Componentes y fixtures

TIPO	PATRÓN	EJEMPLO	CAPA KATA
Componente API	{Resource}Api	UsersApi.ts · OrdersApi.ts	L3
Componente UI (Page)	{Page}Page	LoginPage.ts · CheckoutPage.ts	L3
Módulo Steps	{Domain}Steps	AuthSteps.ts · CheckoutSteps.ts	cadenas L3
Fixture	{Type}Fixture	ApiFixture.ts · UiFixture.ts · TestFixture.ts	L4
Bases	ApiBase · UiBase	ApiBase.ts · UiBase.ts	L2
Context	TestContext	TestContext.ts	L1

automation-standards.md §10 · kata-architecture.md

CAPA · CODE — METADATA DE SELECCIÓN

Tags y Gherkin

`@critical` `@smoke` `@regression` `@e2e` `@integration` `@flaky`

@integration es el tag de pruebas API (reemplaza a `@api`). No confundir con el alias de import `@api/` — es otra cosa, intacto.

Gherkin Feature

```
Feature: <Entity> <Action>
```

```
Feature: User Login
```

Gherkin Scenario CASE

```
Scenario Outline: should <outcome>
<conn> <cond>
```

```
should grant access when credentials are valid
```

automation-standards.md · tms-conventions.md §6

Nombres en el Issue Tracker (Jira / TMS)

Título de issue de calidad bug / defect / improvement

`<EPIC>: <COMPONENT>: <summary>`

CheckoutFlow: Payment: Error not shown for incorrect password

Clasifica por la etapa del ciclo de vida de la feature — no por dónde se encontró.

Clave de User Story

`{{PROJECT_KEY}}-{n}`

PROJ-123 · GX-101

Test Status ciclo de workflow

Draft → In Design → Ready → Manual / In Review → Candidate → In Automation → Pull Request → Automated · Deprecated

Execution Status por corrida — campo distinto

TODO · EXECUTING · PASS · FAIL · ABORTED · BLOCKED

labels: `smoke` `regression` `e2e` `integration` `functional` `automation-candidate` `manual-only` `automated`

Esta capa es **agnóstica del gestor** — Jira es el ejemplo de referencia; las mismas formas aplican a cualquier issue tracker (estados, labels, títulos de calidad).

tms-conventions.md · reporting-templates.md · defect-management-doctrine.md

CAPA · GIT — RAMAS, COMMITS, PRS

Nombres de control de versiones

Rama

`{prefix}/{KEY}-{slug}`

`test/UPEX-123-bulk-assign`

feat · fix · test · docs · refactor · chore

Commit

`{type}({scope}): {desc}`

`feat(UPEX-123): add bulk-assign`

Imperativo · ≤72 chars · sin atribución AI

Título de PR (tests)

`{type}({KEY}): {desc}`

`test(UPEX-123): discount coverage`

conventional-commits.md · git-flow-master/SKILL.md

#naming #conventions #single-source-of-truth

CAPA · FILESYSTEM — LOS ÁRBOLES PBI Y KATA

Nombres de archivos y carpetas

QUÉ	PATRÓN	EJEMPLO
Carpeta de Epic	EPIC-<KEY>-<slug>	EPIC-GX-101-user-auth
Carpeta de Story	STORY-<KEY>-<slug>	STORY-UPEX-110-discount-code
Carpetas coverable	BUG / IMPROVEMENT / TECHSTORY-<KEY>-<slug>	BUG-GX-202-login-error
Scope de test-spec	{PREFIX}-T{NN}-{name}	OD-T01-discount-validation
Spec por ATC	atc/{TICKET-ID}-{brief}.md	atc/UPEX-411-discount-20pct.md
Archivos canónicos	spec.md · automation-plan.md · ROADMAP.md · PROGRESS.md	nombres fijos
Archivos de test KATA	tests/{e2e,integration}/{module}/{verbFeature}.test.ts	tests/e2e/orders/applyDiscount.test.ts

CLAUDE.md §9 · planning-playbook.md

AUDITORÍA DE COBERTURA · LAS 12 CONVENCIONES AHORA RATIFICADAS

Las convenciones ratificadas

Doce artefactos que solo tenían un nombre implícito o indefinido ahora son convenciones ratificadas — escritas en los references de los skills y aplicadas por convención.

- 12 RATIFICADAS
- AHORA CANON
- LINT A FUTURO

#naming #conventions #single-source-of-truth

HUECOS · DATOS, EVIDENCIA Y ARQUITECTURA

Ratificadas · 1–6

1 · Archivos de test-data

Los fixtures en tests/data/ no tienen forma.

`{resource}-{variant}.json → users-valid.json`

2 · Evidencia / screenshots

Los archivos en evidence/ son ad-hoc.

`{KEY}-step{NN}-{action}.png`

3 · Mocks / stubs

Sin hogar ni nombre para mocks de API.

`data/mocks/{endpoint}/{method}.{status}.json`

4 · Numeración de ADR

ADR-NNNN colisiona entre agentes paralelos.

`ADR-{NNNN}-{slug}.md · manual vía README Index (sin script)`

5 · Identificadores de env

Nombres de env solo implícitos por URL.

`local · qa · staging · production`

6 · Carpetas de módulo de test

Subcarpetas e2e/integration sin spec.

`{domain-plural}/ kebab-case`

HUECOS · REPORTING, CORRIDAS Y MODELOS DE DATOS

Ratificadas · 7–12

7 · Labels de suite Allure

Agrupación del reporter indefinida.

`derivar del tag Playwright — única fuente`

8 · Archivos de test-execution

Un archivo md por ejecución vinculada (convención de sync).

`test-executions/{TESTEXEC|RETESTEXEC}-{KEY}-{slug}.md`

9 · Archivos de defect anidados

Un archivo md por defect vinculado (convención de sync).

`defects/DEFECT-{KEY}-{slug}.md`

10 · Data factory / types

Payloads Faker y modelos varían.

`DataFactory.ts · types.ts · constants.ts`

11 · Variables Gherkin

Estilo de placeholder implícito.

`{snake_case} → {user_id}, {order_amount}`

12 · Marcador de test bloqueado

Tests bloqueados por bug se taggean inconsistente.

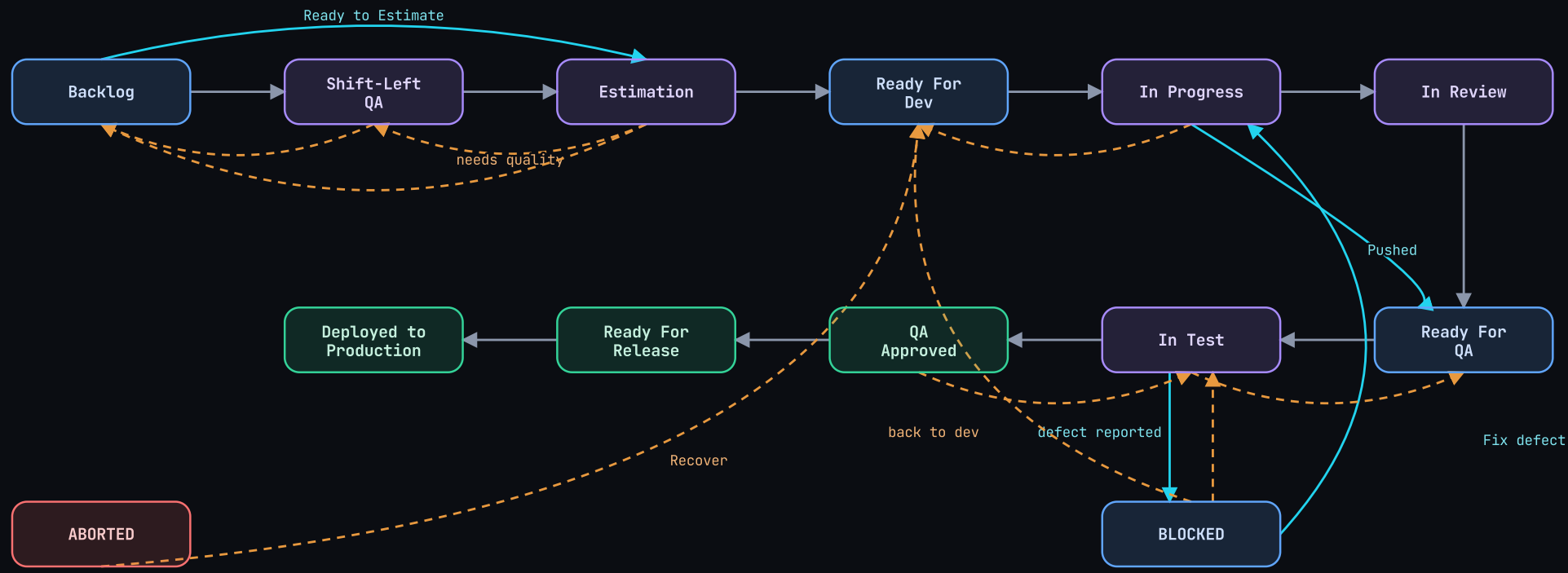
`@blocked:{BUG-KEY} + test.fail()`

Las 12 están ratificadas — ahora canon en los references de los skills. Siguiente: enforcement por lint (opcional).

Workflow de Story

Nuevo · En progreso · Hecho · Abortado / rechazado · Avance · Retorno (back)

⚡ Global (desde cualquier estado): Create → Backlog · Ready For Dev · Ready For QA · ABORTED



El camino feliz va Backlog → Shift-Left QA → Estimation → Ready For Dev → In Progress → In Review → Ready For QA → In Test → QA Approved → Ready For Release → Deployed to Production.

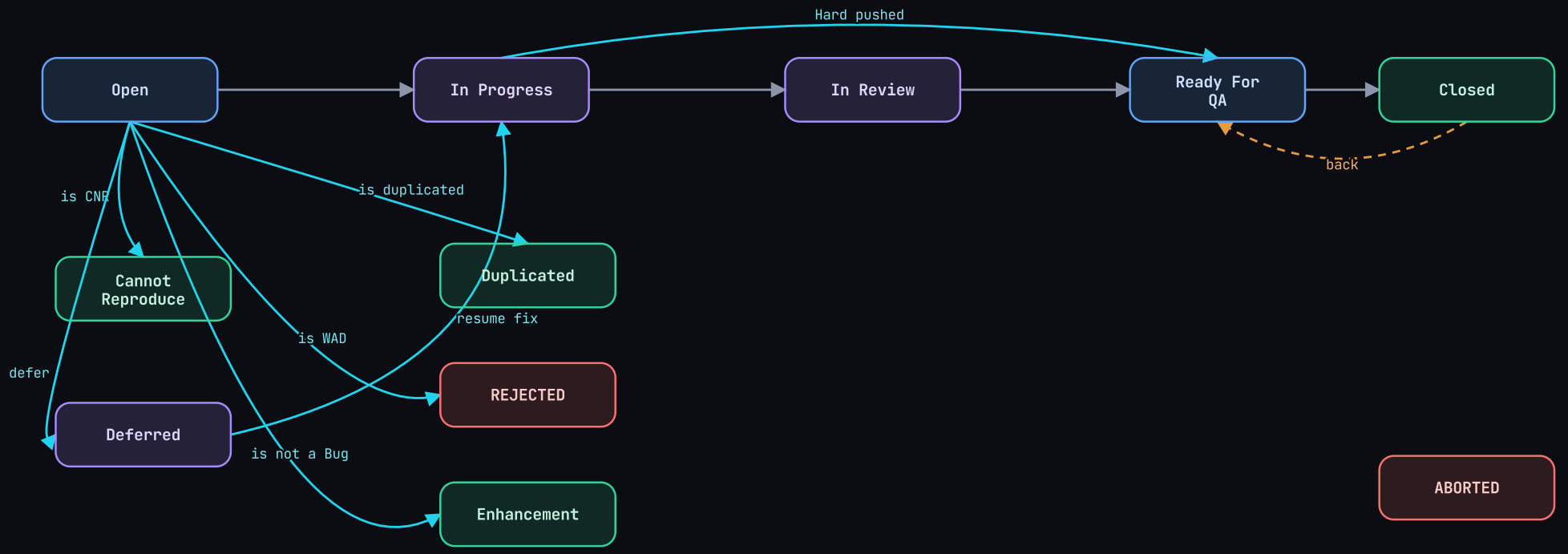
> **Notas.** El camino feliz es la espina de izquierda a derecha en las dos filas. Bucles principales: "back" retorna en casi cada etapa, "needs quality" devuelve Estimation a Shift-Left QA, y la rama BLOCKED (defect reported → Fix defect / back to dev) atiende defectos hallados en pruebas. Los globales Create / Ready For Dev / Ready For QA / ABORTED aplican desde cualquier estado; Recover devuelve una historia ABORTED a Ready For Dev.

CAPA · WORKFLOW JIRA — BUG / DEFECT / IMPROVEMENT (UN CICLO DE VIDA COMPARTIDO)

Bug / Defect / Improvement

Nuevo · En progreso · Hecho · Abortado / rechazado · Avance · Retorno (back)

⚡ Global (desde cualquier estado): Create → Open · Re-Open → Open · ABORTED



El camino feliz va Open → In Progress → In Review → Ready For QA → Closed; desde Open, el triage se bifurca a CNR / Deferred / Duplicated / REJECTED / Enhancement.

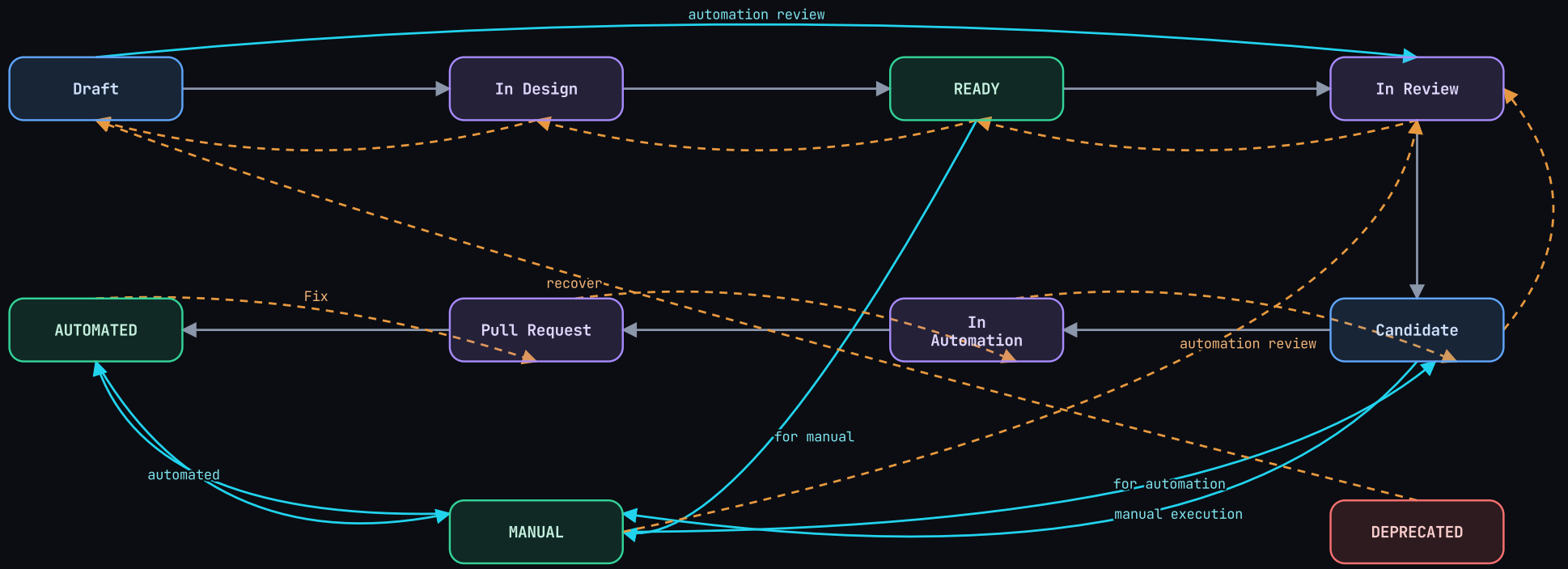
> **Notas.** Bug, Defect e Improvement comparten UN workflow byte-idéntico. El camino feliz es la espina superior. El triage se abre desde Open (is CNR / defer / is duplicated / is WAD / is not a Bug); Deferred puede reanudar la corrección. Bucles: Hard pushed salta la revisión, "back" reabre un ítem Closed para re-test, y Re-Open (global) reabre cualquier ítem resuelto.

#naming #conventions #single-source-of-truth

Workflow de Test

Nuevo · En progreso · Hecho · Abortado / rechazado · Avance · Retorno (back)

⚡ Global (desde cualquier estado): Create → Draft · Deprecated → DEPRECATED



El camino de automatización va Draft → In Design → READY → In Review → Candidate → In Automation → Pull Request → AUTOMATED; la rama MANUAL puede reingresar a automatización.

> **Notas.** El camino feliz de automatización es la espina de izquierda a derecha en las dos filas. MANUAL es el hub: un caso puede enviarse a manual (desde READY / Candidate / AUTOMATED) y reingresar a automatización (automated / for automation / automation review). Bucles: "back" baja la cadena en cada etapa, Fix devuelve AUTOMATED a Pull Request. Deprecated (global) retira cualquier caso; recover devuelve DEPRECATED a Draft.

LAS LEYES DE VERDAD — TODO NOMBRE DEL CÓDICE OBEDECE ESTAS CINCO

Los Cinco Axiomas

No son reglas sueltas: son el patrón que emana de toda la nomenclatura. Si un nombre nuevo no cabe en estos cinco, está mal formado.

A · CLAVE	Todo nombre lleva su clave Jira — así Story ↔ test ↔ código ↔ rama quedan atados	`PROJ-101: …`
B · CAPA	El primer token codifica la capa/altitud — se lee antes que la frase	`ATP: · TS: · @atc`
C · RESULTADO	Los casos se nombran por su resultado, no por su acción — verbo primero	`should <outcome>`
D · PLAN vs RESULTS	Plan y results se distinguen en el primer token (P = Plan, R = Results)	`ATP → ATR`
E · UNA FORMA	Una sola forma por artefacto; la membresía es un link, nunca se mete en el nombre	`link ≠ prefijo`

El siguiente slide aplica estos axiomas a los **5 errores** que más se cometen al ignorarlos.

#naming #conventions #single-source-of-truth

Errores & Reglas de Oro

1 · `should` = caso · `Validate` = grupo

El caso afirma un comportamiento (`should`); el grupo solo nombra la caja (`Validate`). Nunca al revés.

2 · ATC = Acceptance Test Case

No "Automated". El ATC es el método KATA mapeado 1:1 a un TC; la clave `@atc` lo ata a Jira.

3 · R = Results, no Run

ATR / FTR / STR son Test **Results**. Un "run" es solo una iteración de la ejecución; Results es el paraguas.

4 · El prefijo = la clave de la Story

El título de un TC lleva la clave de su User Story. La pertenencia a un Test Set es un **link**, nunca va en el nombre.

5 · `@atc('PROJ-101')` literal

El decorador lleva una clave Jira literal — sin template literals, sin IDs sintéticos.

6 · `@integration`, no `@api`

El tag de API tests es `@integration` (contraparte de `@e2e`). `@api/` es el alias de import — otra cosa.

7 · El casing sigue a la capa

`should` minúscula (continúa el prefijo) · `Validate` mayúscula (inicia el summary del grupo).

8 · Una forma · primer token = capa

Una sola forma por artefacto; el primer token codifica su capa/altitud. Lee la intención antes de la frase.

UNA PANTALLA · TODO EL CÓDICE

Hoja de referencia maestra

CAPA	ARTEFACTO	PATRÓN	EJEMPLO
CASE	TC title	{US_ID}: TC#: should <outcome> [conn cond] [given pre]	PROJ-101: TC1: should grant access when valid
CASE	test() block	'{KEY}: should {behavior} when {cond}'	'UPEX-411: should apply discount when valid'
GROUP	describe()	'{KEY}: Validate {feature}'	'UPEX-411: Validate discount codes'
GROUP	Test Set	TS: {EPIC|module}: Validate {feature}	TS: GX-101: Validate credit card payment
CONTAINER	Plan / Run	{ACRONYM}: {scope}: {desc}	ATP: PROJ-123: … · STR: Sprint#30: Regression Testing
CONTAINER	ReTest · PRC · TS	ReTest: {BUG}: … · PRC: {COMPONENT}: {state} · TS: {EPIC}: Validate {feature}	ReTest: GX-202: … · PRC: Payment: … · TS: GX-101: Validate …
CODE	@atc · ATC · file	@atc('KEY') · {verb}{Res}{Scen} · {verb}{Feat}.test.ts	@atc('PROJ-101') · applyDiscount.test.ts
CODE	components	{Resource}Api · {Page}Page · {Domain}Steps · {Type}Fixture	UsersApi · LoginPage · AuthSteps
CODE	tags	@critical @smoke @regression @e2e @integration @flaky	@integration (no @api)
JIRA	quality title	<EPIC>: <COMPONENT>: <summary>	CheckoutFlow: Payment: Error not shown
GIT	branch · commit · PR	{prefix}/{KEY}-{slug} · {type}({scope}): …	test/UPEX-123-… · feat(UPEX-123): …
FILESYSTEM	árbol PBI	EPIC-<KEY>-<slug> · STORY-<KEY>-<slug> · {PREFIX}-T{NN}	EPIC-GX-101-user-auth

#naming #conventions #single-source-of-truth

MANTENLO CANÓNICO

Una forma, una verdad

Dónde vive

Este deck vive en `agentic-qa-core` — el skill global citado por cada workflow. La fuente en prosa son los `references/` de los skills.

Cuándo cambia

Edita los `references/*.md` canónicos, regenera `REGISTRY.md`, y luego refresca este deck (EN + ES) para que el códice nunca derive.

Qué sigue

Todos los huecos ratificados. Siguiente: reglas de lint opcionales para hacerlos cumplir.

`should = caso · Validate = grupo · prefijo = Story · @atc = clave Jira · @integration ≠ @api`

#naming #conventions #single-source-of-truth