**Glosario completo – términos que un desarrollador va a encontrar a lo largo de la vida útil real de cualquier proyecto software, desde la primera línea de código hasta el retiro en producción.**

(Orden alfabético + breve def. + etiqueta «F», «O», «D» → fase: *Feature*, *Operaciones*, *Decommission*)

A

* A/B experiment – dos versiones activas contra usuario real.
* Acceptance criteria – checklist negociado antes de empezar.
* Accessibility (a11y) – normativa WCAG.
* ACID – atomicidad, consistencia… en BD.
* ADR – Architecture Decision Record.
* Affinity Estimation – planning-poker relativo.
* Agile – iteraciones <= 4 sem.
* Alert fatigue – demasiadas páginas → nadie reacciona.
* Alpha – feature incompleto, solo equipo.
* Anti-corruption layer – patrón DDD en legacy-front.
* API contract – OpenAPI / protobuf firmado.
* Apdex – índice user-satisfaction de latencias.
* Approver – quien firma el MR/PR.
* Architectural runway – infra suficiente para próximas features.
* Artifact – compilado/docker/tar.gz publicado.
* Attack surface – entry-points auditados.
* Atomic deploy – todo o nada.
* Audit log – trazabilidad legal.
* Autoscaling – pods/ec2 bajo métrica.

B

* BDD – Behaviour Driven Dev.
* Benchmark – suite de referencia y SLA.
* Beta – usu externos voluntarios.
* Big-bang release – todo de golpe (*riesgo↑*).
* Blue/Green – dos producciones, swap rápido.
* Boilerplate – código repetitivo.
* Bottleneck – recurso que limita el throughput.
* Branching model – GitFlow / trunk-based.
* Breaking change – compatibilidad rota.
* Brownfield – desarrollo sobre sistema existente.
* Build matrix – varios OS/ver en CI.
* Burn-down – gráfico horas vs. días.
* Bus factor – cantidad de dev desaparecidos que matan el proyecto.
* Business logic – reglas del dominio.

C

* Canary – % de tráfico real sobre nueva versión.
* Capacity planning – CPUs, licencias, $ futuro.
* CAP theorem – choose 2/3.
* CDN – edge estático.
* Chaos Engineering – experimentos controlados (ChaosMonkey).
* Changelog – lista humana de cambios.
* Check-in – commit diario.
* Cherry-pick – portar commit entre ramas.
* CI – integración continua *build+test*.
* Circuit breaker – patrón de resilencia.
* Clean architecture – dependencias hacia dentro.
* Click-through prototype – sin código.
* Cloud-native – contenedores + orquestación.
* Cluster – grupo de nodos.
* CMS – gestor de contenido.
* Cohesion – cosas relacionadas juntas.
* Cold start – latencia al arrancar función/serverless.
* Commit message convention – feat/fix/docs…
* Compliance – GDPR, HIPAA…
* Continuous Delivery – siempre *listo* para PROD.
* Continuous Deployment – automático hasta PROD.
* Contract testing – tests de API por consumidor.
* Corner case – input extremo.
* Coverage – % líneas ejecutadas en tests.
* CRC card – Class-Responsibility-Collaborator.
* Critical path – tareas que deciden la fecha.
* CRUD – Create-Read-Update-Delete.
* CVE – identificador de vulnerabilidad.
* Cyclomatic Complexity – caminos independientes por método.

D

* Dark launch – código en PROD sin users.
* Data migration – movers/transformar entre versiones.
* Dead code – unreachable.
* Dead letter queue – mensajes que fallaron reiteradas veces.
* Debt Snowball – pagar técnico de menor a mayor.
* Defect leakage – bugs encontrados post-release.
* Dependency hell – conflictos de versiones.
* Dependency injection – patrón.
* Deployable unit – artefacto + script de despliegue.
* Design pattern – GoF, Domain patterns…
* Design system – UI kit reutilizable (tokens).
* DevOps – dev y ops misma cadena.
* Disaster Recovery – plan RTO/RPO.
* Documentation debt – docs obsoletas.
* Dogfooding – usamos nuestro producto.
* Domain model – entidades del negocio.
* Downstream – componentes que dependen de mí.

E

* Early-adopter – clientes voluntarios de betas.
* Edge case – situación improbable pero posible.
* E2E test – flujo completo front-back.
* Elevator pitch – resumen 30 seg.
* Embedded docs – swagger/rst dentro del repo.
* Encapsulation – ocultar internos.
* End-of-life – fecha en que deja de existir soporte.
* Epic – gran historia que se parte.
* Error budget – % de fallos permitido por SLO.
* Event sourcing – guardar solo eventos.
* Evolutionary architecture – cambios incrementales.
* Exploratory testing – sesión sin script.
* Extendability – hooks para plugins.
* Extreme Programming – XP (pair, TDD, no overtime).

F

* Fail-fast – detectar y abortar pronto.
* Fallback – plan B cuando falla primario.
* Feature flag – toggle en runtime.
* Feature parity – mismo valor que versión anterior.
* FIFO/LIFO – colas.
* Fire-and-forget – envío asíncrono sin confirmar.
* Fingerprinting – hash de versión para cache-busting.
* First-time-right – calidad upstream.
* Five-whys – técnica root-cause.
* Fixture – datos estáticos de test.
* Force multiplier – herramienta que acelera equipo.
* Fork – copia independiente del repo.
* Forward compatible – viejo entiende nuevo.
* Fuzz testing – inputs aleatorios.

G

* Gamma – pre-release casi lista.
* Garbage collection – autoliberación memoria.
* Gate – aprobación manual previa a PROD.
* Git hook – script pre-commit/push.
* Glamorous champ – celebración micro-éxito (cultura).
* Governance – quién decide qué.
* Graceful degradation – menos funciones pero vivo.
* Green build – pipeline sin rojos.
* Greenfield – proyecto nuevo, sin legacy.
* Grooming – pulir backlog.

H

* Hardening – sprint final antes de lanzar.
* Hard-coded – valor fijo en código.
* Heat map – visualización de hotspots.
* High-cohesion / Low-coupling – mantra de calidad.
* Hot-fix – parche urgente sobre PROD.  
  -Hot-swap – cambio sin downtime.
* HTTP verb – GET/POST/PUT/PATCH/DELETE.
* HTTPS/TLS – certificados obligatorios.
* Hybrid cloud – mixto on-prem + cloud.

I

* Ice-box – tickets congelados.
* Idempotency – repetir sin efecto extra.
* Immutable infra – no cambios in-place.
* Impact map – visual de objetivos vs. features.
* Incident commander – rol durante outage.
* Inception deck – diapositivas iniciales de proyecto.
* Increment – software funcional al terminar sprint.
* Infrastructure as Code – IaC (Terraform, Pulumi).
* Injection (SQL/) – fallo seguridad.
* InnerSource – open-source dentro de la empresa.
* Integration test – varios módulos juntos.
* Interface segregation – principio SOLID.
* Internal milestone – fecha interna antes de la oficial.
* Invariant – regla siempre verdadera.
* ISO-25k/27k – norma seguridad info.
* Iteration – ciclo de 1-4 semanas.

J

* Jailer test – impide salir a PROD si falla.
* Job queue – workers desacoplados.
* JSON-schema – contrato de tipos.
* Jitter – variabilidad de latencia.
* Journey map – pasos del usuario.
* JWT – token stateless.

K

* Kanban – flujo continuo con WIP-limit.
* Kaizen – mejora continua pequeña.
* Key rotation – renovar credenciales.
* Kill switch – apaga feature flag de golpe.
* KLOC – miles de líneas (métrica).
* KPI – indicador clave de negocio.

L

* Latency – tiempo respuesta red/proc.
* Lazy loading – traer al vuelo.
* LTS – soporte extendido.
* Lead time – desde idea hasta PROD.
* Lean – eliminar desperdicio.
* Library vs. framework – tú llamas / te llaman.
* License compliance – GPL, MIT, audit.
* Linting – análisis estático estilo.
* Load balancer – distribuye tráfico.
* Load test – medir comportamiento alto volumen.
* Lock-in – dependencia difícil de abandonar.
* Logging level – TRACE…FATAL.
* Long-lived branch – duran varios sprint (prohibido en trunk-based).
* Look-forward – reunión post-mortem “qué haríamos distinto”.

M

* MAINTAINERS – archivo repo con responsables.
* Maintenance mode – solo bugs críticos.
* Major.Minor.Patch – semver.
* Mallory – atacante clásico crypto.
* Master/trunk – rama principal.
* Mean time to recovery – MTTR.
* Memory leak – reserva sin liberar.
* Microservice – servicio pequeño, desplegable solo.
* Milestone – fecha + alcance.
* Mock – objeto simulado.
* Monorepo – todo en un repo.
* Moratorium – congelamiento de cambios previo a release.
* Multi-tenant – varios clientes mismia instancia.
* MVP – producto mínimo vendible.

N

* Negative testing – probar con datos inválidos.
* No-code / Low-code – config visual.
* No-ops – automatización total (ideal serverless).
* Non-functional req. – -ilities: escalabilidad, seguridad…
* Normalized DB – sin redundancia.
* Null-pointer – error típico.

O

* Observability – métricas, trazas, logs.
* On-call – rotación 24×7.
* One-pager – doc ultra-resumida.
* Open-source license – copy-left vs. permissive.
* Operational expenditure – OPEX recurrente.
* ORM – mapeo objeto-relacional.
* Ownership – “you build it, you run it”.

P

* Package manager – npm, maven, cargo…
* Pair programming – dos dev, una estación.
* Patching – aplicar hot-fix.
* Path-to-prod – diagrama desde commit hasta PROD.
* Pay-down – reducir deuda técnica planificado.
* Pen-test – ataque simulado externo.
* Percentile 95/99 – latencia no solo promedio.
* Performance budget – regla: +100 ms = bloqueo.
* Pet vs. cattle – servidores únicos vs. desechables.
* Poka-yoke – diseño antic-error.
* Post-mortem – documento incidente + lecciones.
* PR (Pull Request) – solicitud de merge.
* Pre-mortem – imaginar fallos antes de lanzar.
* Principle – guía permanente (DRY, SOLID).
* Prioritization matrix – valor vs. esfuerzo.
* Product backlog – lista dinámica user-stories.
* Production-ready – check-list pasado (monitoring, rollback…).
* Proof-of-concept – demo tira-y-afloja.
* Protobuf – serialización binaria.
* Provisioning – crear infra recursos.

Q

* Quality gate – mínimo coverage + análisis estático.
* QA – Quality Assurance equipo.
* Quick win – cambio barato con retorno visible.
* Queue depth – cantidad de mensajes por procesar.
* Quiesce – pausar componente sin pérdida.

R

* RACI – responsables por tarea.
* RAD – rapid app development.
* Ransomware – amenaza relevante backup.
* RC – Release Candidate.
* RCOD – Red, Clean, Orange deploy states (algunos pipelines).
* Reactive system – resiliente, message-driven.
* Refactoring – cambiar estructura, no comportamiento.
* Regression test – suite antigua que debe seguir verde.
* Release note – changelog + upgrade guide user-facing.
* Reliability – SLO % uptime.
* Replay test – volver a ejecutar tráfico real.
* Repo archaeology – investigar historia de código.
* Request-for-comments – RFC para decisiones grandes.
* Resilience – absorber fallo y seguir.
* Response time – idem latencia.
* Retrospective – reunión mejora equipo.
* Return on investment – ROI de feature.
* Reverse proxy – nginx, envoy.
* Rollback – volver versión anterior automática.
* Roll-forward – arreglar adelante en lugar de volver.
* Root cause – origen real del defecto.
* Runbook – pasos manuales durante incidente.

S

* SAAS – software como servicio.
* SAM – Serverless Application Model (AWS).
* Sandboxing – aislar código peligroso.
* Scale-up vs. scale-out – más grande vs. más números.
* Scenario test – caso real usuario.
* Schema evolution – compatibilidad hacia atrás/adelante.
* SCRUM – framework ágil con roles time-box.
* Security by design – pensar threat-model desde UX.
* Semaphore – variable control acceso concurrente.
* SemVer – 2.1.4 → major.minor.patch.
* Serverless – función sin servidor (FaaS).
* Service mesh – comunicación infra entre serv.
* Service Level Agreement – compromiso externo.
* Service Level Objective – interno (<= SLA).
* Service Level Indicator – métrica que alimenta SLO.
* Shadow mode – nuevo procesa pero no afecta resultado.
* SLA breach – incumplimiento penalizable.
* Smoke test – 5 min después del deploy.
* Snake-case – naming con \_.
* Soft delete – flag “borrado” sin delete-SQL.
* SOLID – 5 principios orientación objetos.
* Source of truth – único lugar oficial de dato.
* Spin-up – tiempo de arranque contenedor.
* Spike – investigación time-box.
* Split brain – mitad clúster no ve a otra mitad.
* Sprint – iteración SCRUM.
* SQL injection – fallo clásico.
* Staging – ambiente pre-prod con datos parecidos.
* Stakeholder – interesado (cliente, legal…).
* Stateless – no guarda sesión local.
* Story point – unidad esfuerzo relativo.
* Strangle fig. – migrar legacy envolviendo.
* Stress test – cargar hasta romper.
* Stub – sustituto mínimo.
* Supply-chain security – firmar builds, SBOM.
* Swagger/OpenAPI – contrato REST.
* Syslog – estándar log Unix.

T

* T-shirt sizing – S/M/L estimation.
* TDD – test antes que código.
* Technical debt – atajos que generan interés.
* Tech radar – visualización tecnologías adoptar/trial.
* Telemetry – datos runtime al monitor.
* Threat modeling – STRIDE, PASTA.
* Throughput – trabajos por segundo.
* Time-box – duración fija imposible de ampliar.
* TLS – transport layer security.
* TOGAF – framework arquitectura empresa.
* Tracer bullet – feature mínima end-to-end.
* Trace-ID – correlación logs distribuidos.
* Trade-off – ventaja vs. pérdida documentada.
* Traffic shadow – copiar 100 % peticiones a nuevo clúster sin usar respuesta.
* Trunk-based – pocos commits directos a main.
* TTL – tiempo vida caché.
* Two-pizza team – tamaño ideal (<= 9 personas).

U

* Ubiquitous language – mismo vocabulario dev ↔ negocio.
* UI regression – cambio visual no deseado.
* UML – diagramas (secuencia, clases…).
* Unit test – valida unidad mínima.
* Unix philosophy – haz una cosa y hazla bien.
* Upgrade path – secuencia versions admitidas.
* Usability – facilidad de uso.
* User-journey – pasos usuario mapeados.
* UX writer – redactor micro-textos interfaz.

V

* V-model – contraparte en cascada a TDD.
* Valley of death – fase entre funding rounds (start-ups).
* Version pinning – congelar libs a una exacta.
* Vertical slice – historia cruza todas capas.
* Vex – Vulnerability Exploitability eXchange.
* VHD / AMI – imágenes máquinas.
* VirusTotal – escaneo binario.
* Vulnerability – defecto seguridad.

W

* WIP-limit – máximo trabajos simultáneos Kanban.
* Waterfall – fases secuenciales sin vuelta atrás.
* White-box test – con conocimiento interno.
* Wheel reinvention – volver a escribir lo que ya existe.
* Wildcard cert – \*.company.com.
* Wireframe – boceto UI sin diseño.
* Workaround – bypass temporal que se olvida.
* Write-ahead log – técnica ACID.

X

* XSS – cross-site scripting.
* XA transactions – 2-phase commit distribuido.
* X-matrix – OKR visual año.
* X-Sight – analytics query DSL (algunos APM).

Y

* YAML – formato config declarativo.
* Yellow-flag – riesgo no crítico aún.
* YAGNI – You Aren’t Gonna Need It.

Z

* Zero-day – exploit público sin parche.
* Zero-downtime – blue/green o rolling perfecto.
* Zero-trust – autenticar cada salto.
* Zip Bomb – archivo pequeño que descomprime gigante (vector DoS).