Reactor
COURSES

BROCHURE DEL PROGRAMA



ARQUITECTURA E INGENIERÍA · CONFIABILIDAD Y RENDIMIENTO

# Distributed Systems Architecture

## Datos correctos aunque fallen partes del sistema

Diseña y verifica sistemas que siguen dando la respuesta correcta cuando la red se parte, el reloj de un nodo marca otra hora y la mitad de los mensajes llegan dos veces.

**Lo que vas a lograr** Aprendes a diseñar sistemas repartidos en muchos servidores que siguen dando datos correctos aunque falle la red o una máquina.

- Consistencia y consenso
- Replicación y particionado
- Idempotencia y sagas
- Backpressure
- Inyección de fallos
- Pruebas deterministas

| horas en vivo | sesiones | semanas | fallos que vas a inyectar |
|---|---|---|---|
| 90 | 30 | 15 | 9 |

**INICIO**
Por confirmar

**PRECIO**
**US$420** pago único · USD

**MODALIDAD**
Clases en vivo por Zoom · Avanzado

reactorcourses.com/programas/distributed-systems-architecture

EL PROGRAMA

# Todo funciona hasta que la red se parte. Este programa empieza justo ahí.

Diseña y verifica sistemas que siguen dando la respuesta correcta cuando la red se parte, el reloj de un nodo marca otra hora y la mitad de los mensajes llegan dos veces. Un programa de **90 horas** para ingenieros que ya escriben servicios y descubrieron que el bug no está en su código, sino en los supuestos que hicieron sobre la red.

| | | | |
|---|---|---|---|
| Duración | 90 horas · 30 sesiones | Calendario | 15 semanas · 2 sesiones por semana de 3 h |
| Modalidad | Clases en vivo por Zoom | Grabaciones | Disponibles por 12 meses |
| Credenciales | 3 certificados con insignia de Reactor Courses | Nivel | Avanzado |
| Inicio | Por confirmar | Horario | Lunes y miércoles, 19:00–22:00 (GMT-5) |
| Precio | US$420 · pago único | | |

En un solo proceso una llamada falla o funciona. En un sistema distribuido puede haber funcionado y no enterarte nunca. Casi todos los incidentes graves de producción nacen de tratar lo segundo como si fuera lo primero.

01

### Entender el modelo

Fallos parciales, tiempo, orden y consistencia

La teoría que evita meses de depuración: qué garantiza y qué no una red asíncrona, por qué los relojes mienten, qué significa realmente cada nivel de consistencia y qué se puede y qué no se puede lograr al mismo tiempo. Todo demostrado con código que falla en vivo.

- ✓ Fallos parciales, particiones y el problema de los dos generales
- ✓ Relojes físicos y lógicos, causalidad y orden de eventos
- ✓ Modelos de consistencia, del lineal al eventual, y su costo
- ✓ Replicación, quórum y consenso: Raft explicado implementándolo

02

### Construir y verificar

Idempotencia, sagas, flujo y prueba de fallos

Las técnicas con las que un sistema real sobrevive: operaciones repetibles sin daño, transacciones de negocio que abarcan varios servicios, control de flujo para no caerse por exceso de trabajo, y una batería de pruebas que provoca los fallos en vez de esperarlos.

- ✓ Idempotencia, deduplicación y publicación transaccional de eventos
- ✓ Sagas, compensaciones y flujos de negocio de larga duración
- ✓ Presión de carga, aislamiento de fallos y degradación deliberada
- ✓ Inyección de fallos, pruebas de propiedades y verificación de historias

> En un sistema distribuido no existe «no respondió». Existe «todavía no respondió», y decidir cuánto esperar antes de actuar es una decisión de arquitectura, no un parámetro de configuración.

PARA QUIÉN ES

## ¿Es el programa indicado para ti?

Está pensado para desarrolladores que programan con soltura y quieren entender por dentro los sistemas distribuidos implementando sus protocolos.

### Es para ti si

- ✓ Ingenieros de backend con experiencia que ya operan varios servicios en producción.
- ✓ Quienes depuraron un incidente de datos inconsistentes y quedaron con la sensación de no haber entendido la causa.
- ✓ Arquitectos que deciden entre consistencia y disponibilidad y quieren dejar de hacerlo por intuición.
- ✓ Ingenieros de datos y de plataforma que trabajan con streaming, colas y replicación.
- ✓ SRE que quieren entender por qué su sistema falla en cascada y cómo se detiene esa cascada.

### Te conviene otro programa

- – Quien no programe con soltura. Aquí se implementan protocolos: no hay forma de seguir el curso sin escribir código.
- – Quien busque un curso de una tecnología concreta. Kafka o PostgreSQL aparecen como ejemplos, no como temario.
- – Quien quiera preparación para un examen de certificación. Este programa no prepara ni acredita para ninguno.
- – Quien espere recetas: casi todas las respuestas aquí empiezan con «depende de qué garantía necesites».
- – Quien no pueda dedicar entre 11 y 14 horas por semana durante 15 semanas.
- – Quien no tenga un equipo con al menos 16 GB de memoria: el laboratorio levanta varios nodos en tu máquina.

EL RESULTADO

## Terminas con un sistema que sobrevive a fallos que tú mismo le provocas, y con la evidencia de que sobrevive.

El proyecto final no se juzga porque funcione en la demostración. Se juzga sometiéndolo a una batería de fallos y comprobando que las garantías que prometiste se cumplen.

01 **Registro replicado propio** Implementación de elección de líder y replicación con quórum, con sus pruebas.

02 **Capa de comunicación** Tiempos de espera, reintentos con retroceso, límite de intentos y presupuesto de latencia.

03 **Operaciones idempotentes** Claves de idempotencia, deduplicación y resultados repetibles sin efectos duplicados.

04 **Publicación transaccional** Patrón de bandeja de salida que garantiza que el evento y el dato no se separen.

05 **Saga de negocio** Flujo de varios servicios con compensaciones, reintentos y estado consultable.

06 **Particionado y rebalanceo** Distribución de datos, manejo de claves calientes y movimiento de particiones sin pérdida.

07 **Control de flujo** Presión de carga, aislamiento por compartimentos, corte de circuito y degradación con criterio.

08 **Banco de fallos** Latencia, pérdida, duplicación, reordenamiento, partición, pausa de proceso y desfase de reloj.

09 **Pruebas de propiedades** Invariantes del sistema expresadas como propiedades y ejecutadas con entradas generadas.

10 **Verificación de historias** Comprobación de que las ejecuciones registradas cumplen el modelo de consistencia prometido.

11 **Observabilidad de fallos** Trazas correlacionadas que permiten reconstruir qué pasó durante la partición.

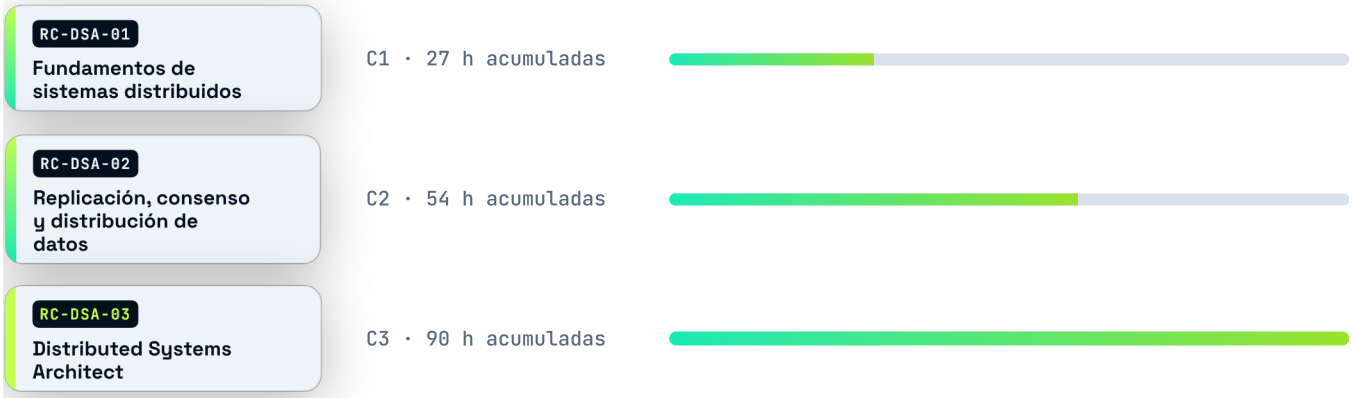
12 **Defensa técnica** Declaras tus garantías y el comité intenta romperlas en vivo con fallos elegidos por ellos.

---

RUTA DE CERTIFICACIÓN

## 3 credenciales progresivas, no una al final.

Cada etapa cierra un bloque completo de competencias. Si algo te obliga a pausar en la semana 9, ya tienes dos credenciales defendibles en tu CV.




CERTIFICACIÓN INTEGRAL

**Distributed Systems Architecture — Datos correctos aunque fallen partes del sistema (90 horas)**

Requiere entregar el sistema, su banco de fallos y la evidencia de que las garantías declaradas se sostienen. Los certificados son emitidos por Reactor Courses y acreditan las horas y competencias del programa.

Cada certificado de Reactor Courses viene con su insignia, aparte de cualquier certificación oficial.

Cualquiera puede comprobar un certificado de Reactor Courses con su código en reactorcourses.com/verificar.

---

TEMARIO

## Las 30 sesiones, abiertas y sin letra chica.

90 horas repartidas en 10 módulos. Así sabes qué verás, qué practicarás y qué entregarás en cada una de las 15 semanas.

## MÓDULO 1 POR QUÉ LO DISTRIBUIDO ES OTRA COSA

9 h · 3 sesiones

### 01 Fallos parciales y el fin de las certezas

las falacias de la computación distribuida · fallo total frente a fallo parcial · el problema de los dos generales · una petición sin respuesta: tres explicaciones posibles · modelos de fallo: caída, omisión, temporal, arbitrario · modelo síncrono, asíncrono y parcialmente síncrono · detección de fallos y su imposibilidad perfecta · por qué el reintento ingenuo empeora el problema

LABORATORIO: Levantar un clúster local de tres nodos y provocar la primera respuesta ambigua

### 02 Tiempos de espera, reintentos y sus consecuencias

elegir un tiempo de espera con datos, no con intuición · presupuesto de latencia repartido entre saltos · reintento con retroceso exponencial y dispersión · límite de intentos y agotamiento del presupuesto · tormenta de reintentos y su efecto multiplicador · cancelación propagada y trabajo huérfano · latencia de cola y su impacto en la disponibilidad percibida · medir antes de configurar

LABORATORIO: Provocar una tormenta de reintentos, medirla y contenerla con retroceso y presupuesto

### 03 Garantías: lo que un sistema promete de verdad

seguridad y vivacidad como categorías de garantía · entrega como máximo una vez, al menos una vez y su combinación · por qué la entrega exactamente una vez no existe en la red · efecto exactamente una vez frente a entrega exactamente una vez · invariante del sistema y cómo se escribe · declarar garantías por escrito antes de implementar · el costo de cada garantía adicional · garantías que el equipo cree tener y no tiene

LABORATORIO: Declarar por escrito las garantías del sistema del programa y sus invariantes verificables

## MÓDULO 2 TIEMPO, ORDEN Y CAUSALIDAD

9 h · 3 sesiones

### 04 Relojes físicos y por qué mienten

deriva de reloj y sincronización por red · salto de reloj hacia atrás y sus efectos · reloj monótono frente a reloj de pared · marcas de tiempo como identificador: por qué falla · el último en escribir gana y la pérdida silenciosa de datos · intervalos de incertidumbre y relojes con cota · ordenar por tiempo en registros de auditoría · detectar problemas de reloj en producción

LABORATORIO: Desfasar el reloj de un nodo y demostrar la pérdida de una escritura por resolución temporal

### 05 Relojes lógicos y orden causal

relación de ocurrencia previa · relojes de Lamport y su límite · relojes vectoriales y detección de concurrencia · orden total, orden parcial y orden causal · difusión con orden causal · identificar conflictos en lugar de ocultarlos · costo de metadatos de causalidad · poda de historia y compactación

LABORATORIO: Implementar relojes vectoriales y detectar escrituras concurrentes en un almacén replicado

### 06 Resolución de conflictos

estrategias de resolución: último gana, unión, específica del dominio · estructuras de datos que convergen sin coordinación · contadores, conjuntos y registros convergentes · resolución en el cliente y sus implicaciones · hermanos y su presentación al usuario · idempotencia como forma de evitar el conflicto · cuándo conviene rechazar en vez de fusionar · consecuencias de negocio de cada estrategia

LABORATORIO: Implementar dos estructuras convergentes y comparar su comportamiento tras una reunificación — compuerta Model Ready

## MÓDULO 3 REPLICACIÓN Y CONSISTENCIA — 9 h · 3 sesiones

### 07 Modelos de replicación

replicación con líder único y flujo de cambios · replicación síncrona, asíncrona y semisíncrona · retraso de réplica y lecturas obsoletas · lectura de tus propias escrituras · replicación con varios líderes y su costo · replicación sin líder y escritura por quórum · promoción de réplica y pérdida de datos en el traspaso · elegir el modelo según la garantía requerida

LABORATORIO: Construir replicación líder-seguidor y medir la ventana de pérdida en una promoción forzada

### 08 Modelos de consistencia

linealizabilidad y su costo real · consistencia secuencial y causal · consistencia eventual y sus anomalías · lectura monótona y consistencia de prefijo · el teorema de las tres propiedades y su lectura correcta · compensación entre latencia y consistencia en operación normal · elegir consistencia por operación, no por sistema · explicar la anomalía al equipo de negocio

LABORATORIO: Mostrar tres anomalías distintas en el mismo sistema variando el nivel de consistencia

### 09 Quórum y sus límites

lectura y escritura por quórum · condición de solapamiento y sus supuestos · lectura reparadora y antientropía · árboles de resumen para comparar réplicas · quórum descuidado y su riesgo · nodos lentos frente a nodos caídos · afinar el quórum según el perfil de carga · casos en que el quórum no alcanza

LABORATORIO: Almacén con quórum configurable y demostración de una lectura obsoleta con quórum insuficiente

## MÓDULO 4 CONSENSO Y COORDINACIÓN — 9 h · 3 sesiones

### 10 El problema del consenso

acuerdo, integridad, validez y terminación · por qué el consenso es imposible en un modelo asíncrono puro · cómo se sortea esa imposibilidad en la práctica · difusión de orden total y su equivalencia con el consenso · el líder como optimización, no como requisito · número de nodos y tolerancia a fallos · el costo de una decisión consensuada · qué merece consenso y qué no

LABORATORIO: Diseñar el registro replicado del proyecto y justificar el tamaño del clúster

### 11 Raft, implementado en clase

términos, votos y elección de líder · registro replicado y confirmación · seguridad de la elección y restricciones de voto · latidos y detección de líder caído · líder aislado que aún se cree líder · compactación del registro y capturas de estado · cambios de membresía sin perder la garantía · errores clásicos al implementarlo

LABORATORIO: Implementar elección de líder y replicación de registro sobre tres nodos

### 12 Coordinación práctica

arrendamientos con vencimiento en lugar de bloqueos eternos · bloqueo distribuido y por qué casi siempre está mal usado · número de cerca para evitar al cliente zombi · membresía y descubrimiento de nodos · difusión de rumores para estado no crítico · coordinación externa frente a coordinación propia · el costo operativo de un servicio de coordinación · diseñar para necesitar la menor coordinación posible

LABORATORIO: Demostrar el fallo de un bloqueo distribuido ingenuo y corregirlo con arrendamiento y número de cerca — compuerta Replication Ready

## MÓDULO 5 PARTICIONADO Y ESCALA DE DATOS — 9 h · 3 sesiones

### 13 Estrategias de particionado

partición por rango y por función de dispersión · elección de la clave de partición · dispersión consistente y movimiento mínimo · particiones virtuales para repartir mejor · sesgo de carga y claves calientes · mitigar una clave caliente sin rediseñar todo · partición y localidad de las consultas · el error de particionar por lo que es fácil

LABORATORIO: Implementar dispersión consistente y medir el desbalance con una distribución realista

14 **Rebalanceo y crecimiento**

agregar y quitar nodos sin detener el servicio · movimiento de particiones y doble escritura durante la transición · enrutamiento de peticiones durante el rebalanceo · rebalanceo automático y su riesgo en cascada · límite de velocidad del movimiento de datos · verificación de integridad tras el movimiento · reversión de un rebalanceo a medias · planificar la capacidad antes de necesitarla

LABORATORIO: Rebalancear un clúster en caliente bajo carga y verificar que no se perdió ni duplicó ningún registro

15 **Consultas sobre datos particionados**

índices secundarios locales y globales · consulta dispersa y agregación de resultados · el costo del nodo más lento en una consulta dispersa · paginación estable sobre datos distribuidos · desnormalización deliberada por patrón de acceso · vistas materializadas y su actualización · consulta analítica sobre un almacén operacional · cuándo la consulta no debe existir

LABORATORIO: Índice secundario global con actualización asíncrona y medición de su ventana de obsolescencia

## MÓDULO 6 TRANSACCIONES DISTRIBUIDAS Y FLUJOS DE NEGOCIO — 9 h · 3 sesiones

16 **Idempotencia y deduplicación**

operación idempotente por diseño frente a por accidente · clave de idempotencia y su ciclo de vida · almacenar el resultado, no solo la marca de procesado · ventana de deduplicación y su vencimiento · idempotencia con efectos externos irreversibles · reintento seguro en cada capa del sistema · prueba de idempotencia como requisito de aceptación · el cobro duplicado y cómo se evita

LABORATORIO: Convertir una operación con efecto monetario en idempotente y demostrarlo con reintentos agresivos

17 **Consistencia entre el dato y el evento**

la doble escritura y su fallo silencioso · patrón de bandeja de salida transaccional · captura de cambios del registro de la base de datos · orden de publicación y particionado del evento · consumidores idempotentes · reproceso desde el principio del registro · esquema del evento y su evolución compatible · borrado y datos personales en un registro inmutable

LABORATORIO: Implementar bandeja de salida transaccional y demostrar que el dato y el evento nunca se separan

18 **Sagas y flujos de larga duración**

confirmación en dos fases y por qué se evita · saga por coreografía y por orquestación · acción compensatoria y su diseño · compensación que no puede deshacer del todo · estado del flujo consultable y su persistencia · vencimientos, reintentos y abandono controlado · visibilidad del flujo para soporte y negocio · pruebas de una saga con fallos en cada paso

LABORATORIO: Saga de tres servicios con compensaciones, sometida a fallo en cada uno de sus pasos

## MÓDULO 7 FLUJO, PRESIÓN Y ESTABILIDAD — 9 h · 3 sesiones

19 **Colas, presión de carga y estabilidad**

teoría de colas aplicada: utilización y latencia · cola sin límite como bomba de tiempo · presión de carga extremo a extremo · rechazo temprano frente a encolamiento infinito · desprendimiento de carga y prioridades · control de admisión y equidad entre clientes · cola de mensajes fallidos y su tratamiento · medir la profundidad de cola como señal temprana

LABORATORIO: Saturar el sistema, observar el colapso y estabilizarlo con presión de carga y desprendimiento

20 **Aislamiento de fallos**

compartimentos estancos por cliente o por dependencia · corte de circuito: estados y umbrales · corte de circuito mal configurado y su daño · límite de concurrencia adaptativo · aislamiento de recursos: conexiones, hilos y memoria · dependencia crítica frente a dependencia opcional · degradación deliberada y respuesta parcial · el fallo en cascada y dónde se corta

LABORATORIO: Provocar una cascada por una dependencia lenta y detenerla con compartimentos y límite de concurrencia

21 **Caché y coherencia**

caché como decisión de consistencia, no de rendimiento · invalidación y sus modos de fallo · estampida de caché y su mitigación · caché de valor vacío y contaminación · caché local frente a caché compartida · coherencia entre réplicas de caché · caché durante un incidente: aliado o mentiroso · medir el acierto y el costo del error

LABORATORIO: Reproducir una estampida de caché y mitigarla, midiendo el efecto sobre la fuente de datos

## MÓDULO 8 VERIFICACIÓN: PROBAR LO QUE FALLA — 9 h · 3 sesiones

22 **Inyección de fallos**

banco de fallos: latencia, pérdida, duplicación, reordenamiento · partición de red y reunificación · pausa de proceso y su parecido con una caída · nodo lento, el fallo más difícil de detectar · corrupción de mensajes y validación defensiva · automatizar el fallo en la suite de pruebas · escenarios de fallo derivados de las garantías declaradas · experimentos con hipótesis y radio de daño acotado

LABORATORIO: Banco de fallos automatizado que ejecuta nueve escenarios contra el sistema del proyecto

23 **Pruebas de propiedades y simulación determinista**

invariante expresada como propiedad ejecutable · generación de entradas y reducción del contraejemplo · simulación determinista con semilla reproducible · control del tiempo y del planificador en la prueba · explorar entrelazados en vez de esperar suerte · reproducir un fallo intermitente a voluntad · cobertura de estados frente a cobertura de líneas · el error que solo aparece una vez cada mil ejecuciones

LABORATORIO: Encontrar una violación de invariante con pruebas de propiedades y reproducirla con semilla fija

24 **Verificación de historias y observabilidad**

registrar la historia de operaciones con inicio y fin · comprobar linealizabilidad sobre una historia registrada · interpretar un contraejemplo de verificación · trazas distribuidas durante una partición · correlación de identificadores a través de saltos · reconstruir qué pasó sin acceso a los nodos caídos · métricas que delatan un problema de consistencia · evidencia que se presenta en una revisión posterior

LABORATORIO: Verificar historias del almacén replicado y presentar la evidencia de la garantía declarada — compuerta Failure Ready

## MÓDULO 9 DISEÑO DE SISTEMAS DISTRIBUIDOS REALES — 9 h · 3 sesiones

25 **Descomposición y límites de servicio**

cuándo dividir y cuándo no dividir un sistema · el monolito distribuido y sus síntomas · límites por propiedad del dato y por transacción · acoplamiento temporal entre servicios · comunicación síncrona y su efecto en la disponibilidad · disponibilidad compuesta de una cadena de servicios · contratos y evolución sin coordinación · el costo distribuido de una decisión de organización

LABORATORIO: Rediseñar un caso mal descompuesto y demostrar la mejora en disponibilidad compuesta

26 **Estimación y capacidad**

modelo de carga: pico, promedio y ráfaga · cálculo de orden de magnitud de tráfico y almacenamiento · presupuesto de latencia por salto · límite práctico de un nodo antes de repartir · sobrecarga de coordinación al agregar nodos · capacidad de reserva para tolerar la caída de una zona · pruebas de carga que reproducen el perfil real · el punto donde escalar deja de ser rentable

LABORATORIO: Modelo de capacidad del sistema del proyecto validado contra una prueba de carga real

27 **Operación y evolución**

despliegue de un cambio de protocolo sin detener el servicio · compatibilidad entre versiones que conviven · migración de esquema en un sistema replicado · banderas de característica para cambios de garantía · reversión cuando el estado ya cambió · ejercicios de recuperación programados · documentar las garantías para quien opera de guardia · resultados incorrectos aceptados por ahora y su fecha de vencimiento

LABORATORIO: Actualizar el protocolo del clúster con versiones conviviendo y sin interrumpir el servicio

## MÓDULO 10 SISTEMA FINAL Y DEFENSA BAJO FALLOS

9 h · 3 sesiones

28 **Integración del sistema final**

ensamblar replicación, particionado, sagas y control de flujo · coherencia entre las garantías declaradas y las implementadas · documento de garantías como contrato del sistema · instrumentación completa y trazas correlacionadas · reproducibilidad del entorno de prueba · lista de verificación de resultados correctos · empaquetado del repositorio para mostrarlo · qué se entrega y en qué formato

LABORATORIO: Sistema integrado corriendo con su banco de fallos completo en verde

29 **Revisión cruzada y ataque entre equipos**

leer las garantías declaradas por otro equipo · diseñar el fallo que las rompe · buscar el reintento no idempotente y la escritura doble · atacar la ventana de promoción de líder · hallazgos priorizados por severidad · corrección sin debilitar la garantía · verificación de la corrección con la prueba que falló · preparación de la defensa

LABORATORIO: Romper el sistema de otro equipo, documentar el ataque y corregir los ataques recibidos

30 **Defensa: el comité elige los fallos**

¿qué garantiza tu sistema y qué no? · ¿qué pasa si el líder se aísla pero sigue vivo? · ¿qué ocurre si este mensaje llega dos veces y en desorden? · ¿dónde puede perderse una escritura confirmada? · ¿cómo detectas que estás sirviendo datos obsoletos? · ¿qué invariante protege el dinero del usuario? · ¿cuánto cuesta la garantía que elegiste? · ¿qué simplificarías si pudieras relajar una garantía?

LABORATORIO: Defensa individual: el comité elige los fallos a inyectar y se ejecutan en vivo

HERRAMIENTAS

# Todo lo que vas a tocar con las manos.

### Lenguaje y ejecución

- Python
- asyncio
- Contenedores
- Docker Compose
- Multiproceso y multinodo local

### Comunicación

- HTTP
- gRPC
- Serialización binaria
- Tiempos de espera y reintentos
- Presupuesto de latencia

### Mensajería y streaming

- Registro de eventos particionado
- Grupos de consumidores
- Reproceso y compensación de desfase
- Entrega y sus garantías

### Almacenamiento

- PostgreSQL
- Niveles de aislamiento
- Bandeja de salida transaccional
- Almacén clave-valor
- Índices y claves calientes

### Coordinación

- Raft implementado en clase
- Elección de líder
- Concesiones y arrendamientos
- Bloqueos distribuidos y sus trampas

### Inyección de fallos

- Proxy de fallos de red
- Partición y reunificación
- Pausa de procesos
- Desfase de relojes
- Pérdida y duplicación

### Verificación

- pytest
- Pruebas basadas en propiedades
- Simulación determinista
- Verificación de historias
- Semilla reproducible

### Observabilidad

- Trazas distribuidas
- OpenTelemetry
- Relojes lógicos en la traza
- Métricas de cola y de latencia

Los logos son marcas de sus respectivos dueños. Se muestran solo para identificar las herramientas que se usan en el programa y no implican aval, patrocinio ni afiliación.

CÓMO FUNCIONA

## Clases en vivo, protocolos implementados a mano y fallos provocados en casi todas las sesiones.

01 **Clases en vivo por Zoom** 30 sesiones de 3 horas, 2 veces por semana. Cámara y micro abiertos: aquí se discute cada supuesto.

02 **Todo queda grabado** Si te pierdes una sesión o vives en otro huso horario, la grabación queda disponible. Acceso por 12 meses.

03 **20% teoría / 80% implementación** La teoría se explica y se implementa el mismo día. Un protocolo que no escribiste no lo entendiste todavía.

04 **Todo corre en tu máquina** Varios nodos en contenedores locales. Sin cuenta de nube, sin costos y con la red bajo tu control para poder romperla.

05 **Cada entrega se ataca** Los laboratorios de implementación no se aprueban porque funcionen: se aprueban cuando sobreviven al banco de fallos correspondiente.

06 **Tres compuertas de calidad** Model Ready, Replication Ready y Failure Ready. No se avanza con garantías declaradas y no demostradas.

### Horarios · Primer grupo

| | |
|---|---|
| Lunes | 19:00 - 22:00 |
| Miércoles | 19:00 - 22:00 |

Horario referido a GMT-5 (Perú, Colombia, Ecuador y Panamá; en Ciudad de México es una hora menos).

Los días y el horario son la propuesta para este grupo y se confirman junto con la fecha de inicio, antes de abrir la matrícula. El calendario definitivo se entrega al confirmar la matrícula.

REQUISITOS

## Lo que necesitas traer a la sesión 01.

No hay nivelación previa. La sesión 01 asume que puedes leer y escribir un servicio con concurrencia sin que sea una aventura.

### Programación

- ✓ Python con soltura, o capacidad de traducir desde otro lenguaje
- ✓ Concurrencia y asincronía
- ✓ Estructuras de datos
- ✓ Depuración de código que no es tuyo

### Servicios

- ✓ Construir y consumir APIs
- ✓ Serialización y contratos
- ✓ Manejo de errores y reintentos
- ✓ Procesos, hilos y señales

### Datos

- ✓ SQL y transacciones
- ✓ Niveles de aislamiento a nivel conceptual
- ✓ Índices
- ✓ Nociones de colas y mensajería

### Redes

- ✓ TCP y sus tiempos de espera
- ✓ Latencia frente a ancho de banda
- ✓ DNS y descubrimiento de servicios
- ✓ Diagnóstico básico de conectividad

### Pruebas

- ✓ Pruebas unitarias y de integración
- ✓ Dobles de prueba
- ✓ Ejecutar una suite en CI
- ✓ Leer una traza de error hasta el final

### Entorno

- ✓ Docker y Docker Compose
- ✓ Equipo con 16 GB de RAM o más
- ✓ Terminal y Git
- ✓ Editor con depurador

PRECIO

# 90 horas para dejar de descubrir en producción lo que la teoría ya sabía.

**US$420**

Pago único · USD · Incluye impuestos

Todavía no hay fecha de inicio. Escríbenos y te avisamos cuando la confirmemos: no cobramos nada antes de abrir la matrícula.

### Incluye

- ✓ 30 sesiones en vivo por Zoom (90 horas lectivas)
- ✓ Grabación de todas las clases con acceso por 12 meses
- ✓ Repositorio base, banco de fallos y arneses de prueba listos para usar
- ✓ Revisión de entregables en las 3 compuertas de calidad
- ✓ Revisión de código entre pares en cada módulo
- ✓ Sistema final con acompañamiento y defensa técnica bajo fallos
- ✓ Los 3 certificados progresivos + la certificación integral
- ✓ Canal privado del grupo para dudas entre sesiones
- ✓ Actualizaciones del material por 12 meses

### No incluye

- — El equipo de trabajo: necesitas una máquina con 16 GB de memoria como mínimo
- — Exámenes ni certificaciones de terceros: este programa no prepara para ninguno
- — Licencias comerciales: todo el laboratorio corre con software de código abierto

### Métodos de pago

Tarjeta de crédito · Tarjeta de débito internacional · PayPal · Apple Pay · Google Pay

**Garantía de 14 días.** Si dentro de los primeros 14 días del programa decides que no es para ti, te devolvemos el 100 % de lo pagado. No pedimos justificación.

Paddle actúa como comerciante registrado (merchant of record): emite el comprobante y gestiona los impuestos de cada país.

---

PREGUNTAS FRECUENTES

# Preguntas frecuentes sobre el programa.

### ¿Las clases son en vivo o grabadas?

En vivo por Zoom: 30 sesiones de 3 horas, 2 veces por semana. Todas quedan grabadas y disponibles por 12 meses, así que puedes recuperar una clase o volver a ver una implementación con calma.

### ¿Cuándo empiezan las clases?

La fecha de inicio y el calendario definitivo todavía no están confirmados: se publican en esta misma ficha antes de que se abra la matrícula. Los días propuestos son Lunes y miércoles en horario de GMT-5 (Perú, Colombia, Ecuador y Panamá; en Ciudad de México es una hora menos). Si quieres que te avisemos cuando se confirmen, escribe a soporte@reactorcourses.com.

### ¿Es un curso teórico?

Es teórico y práctico en la misma sesión. La teoría de sistemas distribuidos es la parte útil, pero no se queda en la pizarra: cada concepto se implementa el mismo día y se somete a fallos. Vas a escribir elección de líder, replicación por quórum y sagas con tus manos.

### ¿En qué lenguaje se programa?

En Python, porque deja ver el protocolo sin que el lenguaje estorbe. Si trabajas en Go, Java, C# o TypeScript, todo lo que aprendas se traduce sin fricción: lo que se enseña son garantías y protocolos, no una biblioteca.

### ¿Necesito una cuenta de nube?

No. Todo el laboratorio corre en tu máquina con contenedores, y eso es deliberado: para romper la red a voluntad necesitas controlarla. Lo único que hace falta es un equipo con 16 GB de memoria o más.

### ¿Se enseña Kafka, Kubernetes o alguna tecnología concreta?

Aparecen como ejemplos cuando ilustran un concepto, pero no son el temario. Este programa enseña las garantías que hay detrás: cuando entiendes qué significa entrega al menos una vez, configurar cualquier herramienta que la implemente deja de ser adivinanza.

### ¿Este programa prepara para alguna certificación?

No. No prepara ni acredita para ninguna certificación de AWS, Microsoft, Google, la CNCF ni ninguna otra organización, y Reactor Courses no tiene relación ni aval de ellas.

### ¿Cuánto tiempo debo dedicarle por semana?

Seis horas de clase más entre cinco y ocho horas de implementación. Considera entre 11 y 14 horas semanales durante 15 semanas. Es el programa más exigente del catálogo en carga de código.

### ¿Sirve si mi sistema es un monolito?

Sirve, y probablemente más de lo que crees. En cuanto tu monolito habla con una base de datos, una cola o un servicio externo, ya tienes fallos parciales. Además, entender bien este material es la mejor defensa contra dividir un sistema que no hacía falta dividir.

### ¿Qué métodos de pago aceptan?

El pago lo procesa Paddle y acepta Tarjeta de crédito, Tarjeta de débito internacional, PayPal, Apple Pay, Google Pay. El cobro es en dólares. Importante: no procesamos Yape, Plin, OXXO, Mercado Pago ni transferencias locales. Si tu banco rechaza compras internacionales, habilítalas antes de pagar o usa PayPal.

### ¿Puedo pagar en cuotas?

Por ahora el programa se cobra en un pago único de US$420. Algunos bancos permiten diferir la compra en cuotas desde su propia app o banca en línea; eso depende de tu emisor, no de nosotros.

### ¿Emiten factura o comprobante?

Sí. Paddle actúa como comerciante registrado (merchant of record): emite el comprobante y gestiona los impuestos de cada país. Lo recibirás por correo automáticamente al completar el pago. Si necesitas datos fiscales de empresa, cárgalos en el campo correspondiente de la ventana de pago.

### ¿Cuál es la política de reembolso?

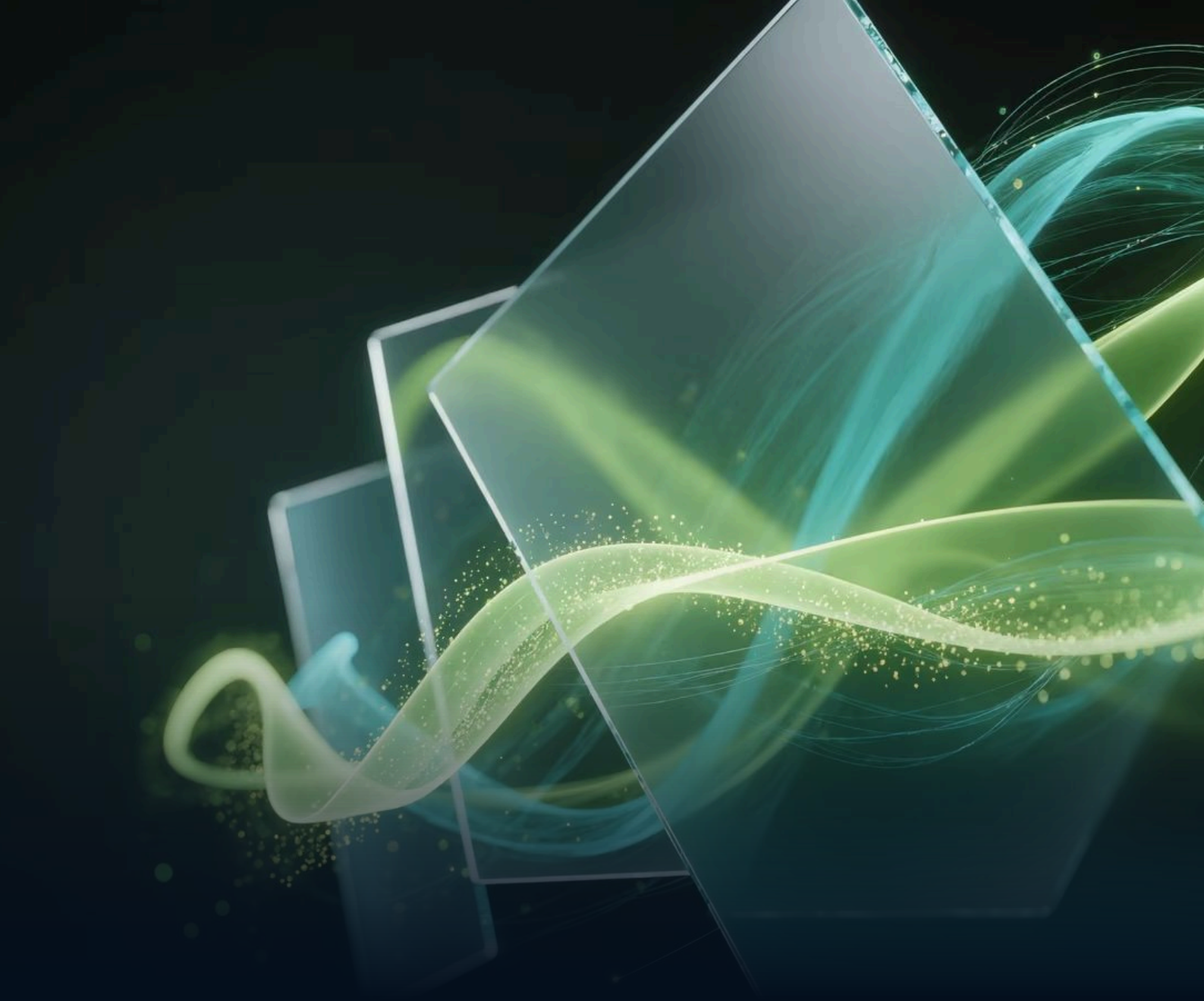
Tienes 14 días desde el inicio del programa para pedir el reembolso completo escribiendo a soporte@reactorcourses.com. Publicamos el temario completo y los requisitos justamente para que decidas informado y nadie llegue a ese punto.

### ¿Los certificados tienen validez universitaria?

No. Son certificados profesionales emitidos por Reactor Courses que acreditan horas y competencias. No son un grado académico ni una certificación oficial de ninguna organización. Lo que de verdad te va a servir en una entrevista es el repositorio y poder explicar por qué tu sistema no pierde datos durante una partición.

### ¿Este programa se cruza con otros del catálogo?

Cada programa de Reactor Courses es un producto cerrado: trae sus propios fundamentos, su propio proyecto final y no da por visto nada de otro curso. Otros programas de la escuela mencionan colas, reintentos o resiliencia como una decisión de diseño más; aquí son el objeto de estudio y se implementan y se verifican. Si cursas dos programas vas a volver a ver algunos temas, y no hay convalidaciones ni descuentos entre programas: preferimos decirlo antes y no que lo descubras pagando.



Reactor
COURSES

Ingeniería de software en la era de los agentes

## ¿Te quedó alguna duda?

Escríbenos a info@reactorcourses.com. La ficha al día, con el temario interactivo y la inscripción, está en reactorcourses.com/programas/distributed-systems-architecture.

`reactorcourses.com/programas/distributed-systems-architecture`

---

Información vigente al 23 de noviembre de 2026. El precio, las fechas y el temario pueden cambiar: la versión al día es la de la web.

Reactor Courses no está afiliado a Microsoft, Amazon Web Services, OpenAI, Anthropic ni Google. Los nombres de productos se mencionan únicamente con fines descriptivos del contenido académico.