Reactor
COURSES

BROCHURE DEL PROGRAMA



ARQUITECTURA E INGENIERÍA · CONFIABILIDAD Y RENDIMIENTO

Arquitectura de rendimiento y escalabilidad

Medir primero, optimizar después

Encuentra y resuelve problemas de rendimiento con datos: modela la carga, mide sin engañarte, localiza el cuello de botella con método y corrígelo donde está, expresando cada mejora en latencia de cola, capacidad y costo por transacción.

Lo que vas a lograr Aprendes a encontrar por qué un sistema es lento, midiendo en lugar de adivinar, y a prepararlo para un pico de tráfico.

- Percentiles y latencia de cola
- Pruebas de carga válidas
- Perfilado y gráficos de llama
- Base de datos y caché
- Ley de escalabilidad universal
- Costo por transacción

horas en vivo	sesiones	semanas	la capacidad máxima en la prueba de sobrecarga
60	20	10	3×

INICIO
Por confirmar

PRECIO
US$360 pago único · USD

MODALIDAD
Clases en vivo por Zoom · Avanzado

reactorcourses.com/programas/performance-scalability-architecture

EL PROGRAMA

Optimizar sin medir es adivinar. Medir mal es adivinar con gráficos.

Encuentra y resuelve problemas de rendimiento con datos: modela la carga, mide sin engañarte, localiza el cuello de botella con método y corrígelo donde está, expresando cada mejora en latencia de cola, capacidad y costo por transacción. Un programa de **60 horas** para equipos con un sistema lento y opiniones encontradas sobre por qué, y para quienes preparan un evento de alta carga y tienen que llegar con números.

Duración	60 horas · 20 sesiones	Calendario	10 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	Miércoles y viernes, 19:00–22:00 (GMT-5)
Precio	US$360 · pago único		

Casi todos los problemas de rendimiento empiezan con una opinión —es la base, es el lenguaje, hay que escalar— y terminan con un cambio que no movió nada. Este programa enseña el método: modelar la carga, medir sin engañarse, localizar el cuello de botella con evidencia, corregirlo donde está y expresar el resultado en latencia de cola, capacidad y costo por transacción.

01

Medir y localizar

Percentiles, modelo de carga, pruebas válidas y método

Percentiles en lugar de promedios, un modelo de carga derivado del uso real, pruebas que no sufren omisión coordinada y se pueden repetir, y los métodos USE y RED con perfilado de CPU, memoria y espera para encontrar el cuello de botella antes de tocar una línea.

- Histogramas, percentiles y latencia de cola
- Modelo de carga y pruebas repetibles sin omisión coordinada
- Métodos USE y RED
- Perfilado de CPU, memoria y tiempo fuera de CPU

02

Corregir y dimensionar

Base de datos, caché, concurrencia, modelos, sobrecarga y costo

Planes de ejecución, N+1 y pool de conexiones; caché con protección contra estampidas; contención por filas calientes; modelos de Little y de escalabilidad universal para predecir; colas y salas de espera para sobrevivir al pico; y el costo por transacción antes y después de cada mejora.

- Consultas, índices, N+1 y pool de conexiones
- Caché, contención y tamaño de las respuestas
- Ley de escalabilidad universal y escalado medido
- Sobrecarga, borde, costo por transacción y capacidad

> Un promedio de 200 milisegundos puede esconder a un usuario de cada cien esperando diez segundos. Ese usuario es el que se va.

PARA QUIÉN ES

¿Es el programa indicado para ti?

Está pensado para desarrolladores que quieren medir y mejorar el rendimiento de una aplicación real bajo carga.

Es para ti si

- ✓ Desarrolladores y líderes técnicos con un sistema lento y opiniones encontradas sobre por qué.
- ✓ Equipos que preparan un evento de alta carga: campañas, cierres, preventas o temporadas.
- ✓ Arquitectos que deben dimensionar un sistema y justificar el costo.
- ✓ Ingenieros que hicieron pruebas de carga y no confían en sus resultados.
- ✓ Responsables técnicos que tienen que explicar al negocio cuánto cuesta cada transacción.

Te conviene otro programa

- – Quien no escriba código: los laboratorios perfilan y modifican una aplicación.
- – Quien busque teoría de sistemas distribuidos o particionado global de datos.
- – Quien espere trucos de un lenguaje concreto: el método es transversal.
- – Quien busque optimizar el frontend o el navegador: aquí se trabaja del servidor hacia atrás.
- – Quien ya cursó un programa de confiabilidad o de sistemas distribuidos: cerca de un tercio de este programa — pruebas de carga, caché con estampidas, escalado, colas y contrapresión, CDN y capacidad— toca temas que esos programas tratan con otro objetivo. Lee en las preguntas frecuentes la que trata ese caso.
- – Quien no pueda dedicar entre 9 y 11 horas por semana durante 10 semanas.

EL RESULTADO

Terminas con un sistema diagnosticado con método y un informe que cualquiera puede repetir.

Cada laboratorio deja una mejora medida sobre la aplicación de venta de entradas del caso. El proyecto final es una versión con problemas que no conoces, diagnosticada y corregida con el mismo método.

01 **Medición honesta** El promedio aceptable que escondía un p99 inaceptable, demostrado con histogramas.

02 **Modelo de carga** Mezcla, llegada, pico y sesiones derivados de registros reales y versionados como escenario.

03 **Prueba repetible** Omisión coordinada demostrada y corregida, y una prueba repetible con variación del p99 bajo el diez por ciento.

04 **Cuello de botella localizado** USE y RED aplicados con cada descarte documentado, y la primera causa encontrada con evidencia.

05 **CPU y memoria corregidas** Función costosa encontrada en el gráfico de llama y pausas de recolección eliminadas del p99.

06 **Base de datos optimizada** Tres consultas corregidas por plan, N+1 eliminados con una prueba que lo vigila y pool dimensionado con la ley de Little.

07 **Caché sin estampidas** Caché con coalescencia de peticiones, con acierto, carga de la base y frescura medidos.

08 **Contención resuelta** Fila caliente del stock resuelta sin vender una entrada de más.

09 **Capacidad predicha** Ley de escalabilidad universal ajustada a mediciones y su predicción verificada con una corrida real.

10 **Pico sobrevivido** Sala de espera virtual y colas acotadas que sostienen tres veces la capacidad, con recuperación medida.

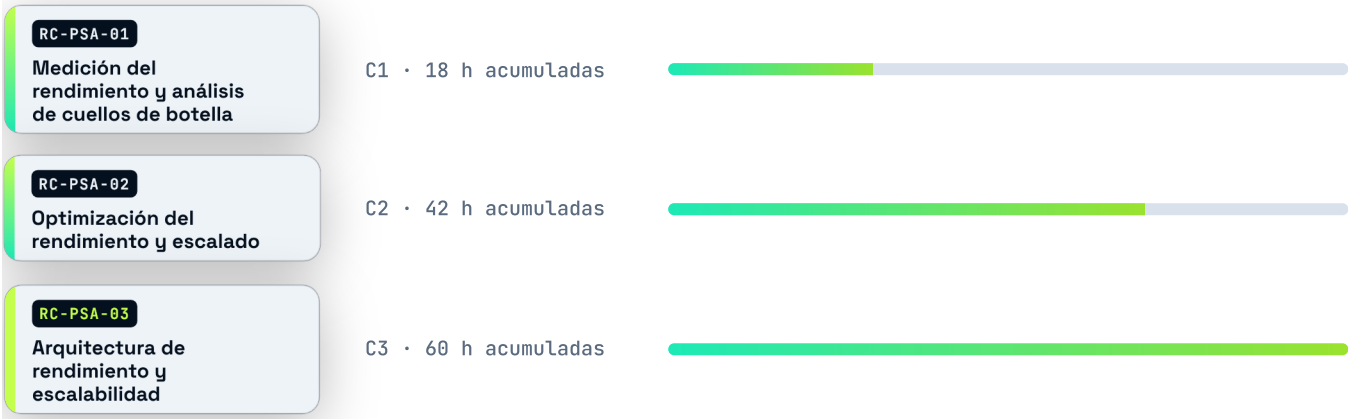
11 **Costo por transacción** Costo antes y después de las mejoras, con la mejora que más lo bajó y la que lo subió.

12 **Plan para el día del evento** Plan de capacidad validado con un ensayo general a la escala esperada más el margen.

RUTA DE CERTIFICACIÓN

3 credenciales progresivas, no una al final.

Cada etapa cierra un bloque completo: primero medir y localizar, después optimizar y escalar, y al final sobrecarga, costo y capacidad. Son credenciales de Reactor Courses, no certificaciones de terceros.



CERTIFICACIÓN INTEGRAL

Performance & Scalability Architecture — Medir primero, optimizar después (60 horas)

Requiere entregar el diagnóstico de la aplicación con problemas sembrados, con una prueba de carga por mejora y el informe antes y después. 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 20 sesiones, abiertas y sin letra chica.

60 horas repartidas en 8 módulos. Así sabes qué verás, qué practicarás y qué entregarás en cada una de las 10 semanas.

MÓDULO 1 **MEDIR BIEN** — 9 h · 3 sesiones

01 **Latencia, percentiles y por qué el promedio miente**

el caso: la venta de entradas que se prepara para la preventa de un concierto · distribución de la latencia y su cola · percentiles: p50, p95, p99 y p99,9 · por qué no se promedian percentiles · histogramas de alta resolución · la latencia que sufre un usuario que hace muchas peticiones

LABORATORIO: Levantar la aplicación del caso, medir la misma operación con promedio y con histograma, y demostrar con los datos que un promedio aceptable esconde un p99 que dejaría sin comprar a uno de cada cien usuarios

02 **Modelar la carga a partir del uso**

mezcla de operaciones y su peso · proceso de llegada de las peticiones · picos, ráfagas y la apertura de una preventa · tiempo de pensamiento y sesiones de usuario · derivar el modelo desde los registros de acceso · datos de prueba que no favorecen a la caché

LABORATORIO: Derivar el modelo de carga de la preventa a partir de registros de acceso entregados —mezcla, llegada, pico y sesiones— y expresarlo como un escenario de carga versionado

03 Pruebas de carga que no engañan

modelo abierto frente a modelo cerrado · omisión coordinada según la documentación de wrk2 · calentamiento y estado estable · el generador de carga como cuello de botella · repetibilidad: varias corridas y su variación · informe de resultados con percentiles y no con promedios

LABORATORIO: Ejecutar la misma carga con un generador que sufre omisión coordinada y con uno que la corrige, mostrar la diferencia en el p99 y dejar una prueba con k6 repetible cinco veces con variación del p99 menor al diez por ciento

MÓDULO 2 LOCALIZAR EL CUELLO DE BOTELLA

9 h · 3 sesiones

04 Métodos USE y RED

método USE: uso, saturación y errores por recurso · método RED: tasa, errores y duración por servicio · lista de recursos del sistema del caso · de la hipótesis a la medición y al descarte · herramientas de Linux para observar recursos · el cuello de botella que se mueve al resolver otro

LABORATORIO: Aplicar USE a cada recurso y RED a cada servicio del caso bajo la carga de la preventa, y localizar el primer cuello de botella con la evidencia de cada descarte

05 Perfilado de CPU y gráficos de llama

perfilado por muestreo frente a instrumentación · gráficos de llama: cómo leerlos sin engañarse · async-profiler para la JVM y py-spy para Python · perfilar en un sistema bajo carga real · trabajo inútil y trabajo repetido · medir el efecto de un cambio con el mismo perfil

LABORATORIO: Perfilar la aplicación bajo carga, encontrar en el gráfico de llama la función que más CPU consume sin aportar valor, corregirla y demostrar con la prueba repetible la mejora en el p99 y en el uso de CPU

06 Memoria, recolección de basura y espera

asignaciones y su costo · pausas de recolección de basura en la cola de latencia · perfilado de asignaciones · tiempo fuera de CPU: esperas de E/S y de bloqueos · fugas de memoria que aparecen tras horas · configuración del recolector como último recurso

LABORATORIO: Encontrar con perfilado de asignaciones y de tiempo fuera de CPU la causa de los picos de latencia del caso, corregirla y demostrar que las pausas dejan de aparecer en el p99 de una prueba de una hora

MÓDULO 3 BASE DE DATOS

9 h · 3 sesiones

07 Planes de ejecución e índices

leer un plan de ejecución en PostgreSQL · estimaciones equivocadas y estadísticas · índices compuestos, parciales y de cobertura · el costo de escribir con muchos índices · consultas lentas registradas como fuente · medir la consulta dentro de la carga, no aislada

LABORATORIO: Encontrar las tres consultas que más tiempo suman bajo la carga de la preventa, corregirlas con índices y reescritura guiadas por el plan de ejecución, y medir el efecto en el p99 y en la escritura

08 Patrones de acceso: N+1 y el ORM

el problema N+1 y cómo se ve en una traza · carga ansiosa frente a carga perezosa · consultas por lotes · paginación eficiente · lo que el ORM genera y nadie lee · pruebas que cuentan consultas

LABORATORIO: Detectar con trazas los N+1 del listado de eventos, corregirlos con carga por lotes y dejar una prueba automática que falla si el número de consultas por petición vuelve a crecer

09 Pool de conexiones y réplicas de lectura

conexiones como recurso escaso · tamaño del pool y la ley de Little · PgBouncer y sus modos · réplicas de lectura y su atraso · qué consultas pueden ir a una réplica · agotamiento del pool como falla en cascada

LABORATORIO: Dimensionar el pool de conexiones del caso con la ley de Little y PgBouncer, demostrar que el tamaño calculado sostiene la preventa, y desviar las lecturas tolerantes al atraso a una réplica midiendo la carga del primario

MÓDULO 4 CACHÉ, CONCURRENCIA Y DATOS EN TRÁNSITO

9 h · 3 sesiones

10 Caché con invalidación y protección contra estampidas

caché al lado, de lectura y de escritura · Valkey como caché del laboratorio · invalidación por evento y por expiración · estampida al expirar una clave popular · coalescencia de peticiones y expiración temprana probabilística · medir acierto, ahorro y frescura

LABORATORIO: Introducir caché en el detalle del evento con Valkey, provocar una estampida al expirar la clave durante la preventa y mitigarla con coalescencia de peticiones, midiendo acierto, carga de la base y frescura de los precios

11 Concurrencia, contención y E/S asíncrona

contención por bloqueos en el código y en la base · filas calientes: la última entrada disponible · E/S asíncrona y modelos de hilos · límites de concurrencia por recurso · operaciones atómicas frente a bloqueos · medir la contención con perfiles de espera

LABORATORIO: Reproducir la contención por la fila caliente del stock de entradas en la apertura de la preventa, resolverla con una operación atómica y un límite de concurrencia, y demostrar que no se vende una entrada de más

12 Serialización, compresión y tamaño de respuesta

el costo de serializar y deserializar · respuestas más grandes que lo que se usa · compresión y su costo de CPU · formatos binarios frente a JSON · paginación y campos a pedido · medir bytes por transacción

LABORATORIO: Medir bytes y tiempo de serialización por transacción en las tres operaciones más frecuentes, reducirlos con campos a pedido y compresión, y reportar el efecto en latencia y en CPU de ambos lados

MÓDULO 5 MODELOS Y ESCALADO

6 h · 2 sesiones

13 Little, Amdahl y la ley de escalabilidad universal

ley de Little aplicada a un servicio · ley de Amdahl y la parte que no se paraleliza · ley de escalabilidad universal: contención y coherencia · ajustar el modelo a mediciones reales · predecir el rendimiento máximo antes de comprar hardware · cuándo el modelo deja de valer

LABORATORIO: Medir el rendimiento del caso con 1, 2, 4, 8 y 16 trabajadores, ajustar la ley de escalabilidad universal a esas mediciones y predecir el punto de rendimiento máximo, verificándolo después con una corrida real

14 Escalado vertical, horizontal y sus retrasos

escalar hacia arriba frente a escalar hacia afuera · estado que impide escalar horizontalmente · autoescalado y su tiempo de reacción · calentamiento de instancias nuevas · escalar antes del pico conocido · límites que aparecen al escalar

LABORATORIO: Medir el tiempo que tarda el autoescalado del caso en absorber la apertura de la preventa, compararlo con un escalado programado antes del pico y documentar qué usuarios sufren en cada variante

MÓDULO 6 SOBRECARGA Y BORDE

6 h · 2 sesiones

15 Colas y contrapresión bajo sobrecarga

qué pasa más allá de la capacidad máxima · colas acotadas y rechazo temprano · sala de espera virtual para una preventa · contrapresión entre servicios · respuesta útil al usuario rechazado · medir la recuperación después del pico

LABORATORIO: Llevar el caso a tres veces su capacidad máxima, demostrar que sin protección colapsa, y sostener la preventa con una sala de espera virtual y colas acotadas, midiendo el tiempo de recuperación tras el pico

16 CDN y borde

qué se puede servir desde el borde · cabeceras de caché y su efecto en la CDN · invalidación en el borde · contenido personalizado y caché · medir desde la ubicación del usuario · costo de la transferencia y de la CDN

LABORATORIO: Rediseñar las cabeceras de caché del caso para que la CDN sirva el catálogo de eventos y las imágenes, y medir con la carga de la preventa la reducción de peticiones y de bytes que llegan al origen

MÓDULO 7 **COSTO Y CAPACIDAD** 6 h · 2 sesiones

17 **Costo por transacción**

costo por transacción como métrica de diseño · repartir el costo de infraestructura entre operaciones · la optimización que baja la latencia y sube el costo · costo de la capacidad ociosa · precios de referencia de la nube con fecha · informe de costo antes y después

LABORATORIO: Calcular el costo por transacción del caso con precios de lista de una nube de referencia antes y después de las mejoras del programa, e identificar la mejora que más bajó el costo y la que lo subió

18 **Planificación de capacidad para el evento**

capacidad máxima sostenible medida · margen para el pico y para la falla · plan de escalado del día del evento · ensayo general de carga · criterios para cancelar o postergar la apertura · comunicación con el negocio

LABORATORIO: Producir el plan de capacidad de la preventa con capacidad sostenible medida, margen y plan de escalado, y validarlo con un ensayo general de carga a la escala esperada más el margen

MÓDULO 8 **PROYECTO INTEGRADOR** 6 h · 2 sesiones

19 **Diagnóstico de la aplicación con problemas sembrados**

problemas sembrados que no se conocen de antemano · método antes que intuición · mejoras priorizadas por impacto medido · una mejora a la vez con su prueba · registro de hipótesis y descartes · informe antes y después

LABORATORIO: Diagnosticar con método una versión de la aplicación con problemas sembrados que no conoces, corregirlos en orden de impacto con una prueba de carga por mejora, y entregar el informe con p50, p95, p99, capacidad máxima y costo por transacción antes y después

20 **Defensa del diagnóstico**

método aplicado y descartes · mejoras con su evidencia · capacidad y costo · lo que no se optimizó y por qué · riesgos para el día del evento · preguntas cruzadas del grupo

LABORATORIO: Defender ante el grupo el diagnóstico y las mejoras del proyecto con las pruebas de carga repetidas en vivo, y responder con datos del propio laboratorio por qué cada mejora era la siguiente

LO QUE VAS A USAR

Herramientas libres que sirven al método, no al revés.

Carga

k6 · wrk2

Perfilado

async-profiler · py-spy · Gráficos de llama

Datos

PostgreSQL · PgBouncer · Valkey

Observación

Herramientas de Linux para recursos · Trazas distribuidas

Modelos

Ley de Little · Ley de escalabilidad universal

Borde

Cabeceras de caché para CDN

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, una aplicación bajo carga y ninguna mejora aceptada sin su prueba.

01 **Clases en vivo por Zoom** 20 sesiones de 3 horas, 2 veces por semana. Se mide y se corrige en vivo, con el grupo discutiendo cada hipótesis.

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

03 **30% concepto / 70% laboratorio** Cada técnica se aplica el mismo día sobre la aplicación del caso y se mide antes y después.

04 **Sin promedios** Ningún laboratorio reporta promedios: se reporta p50, p95 y p99 con la cantidad de muestras.

05 **Una mejora, una prueba** Cada mejora se acepta solo con una prueba de carga repetible que la demuestra. Las cifras son las del laboratorio, no de terceros.

06 **Problemas que no conoces** El proyecto final es una versión de la aplicación con problemas sembrados que el alumno no conoce de antemano.

Horarios · Primer grupo

Miércoles	19:00 - 22:00
Viernes	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 mide la aplicación del caso bajo carga en la primera hora.

Programación

- ✓ Un lenguaje backend: Java, C#, Go, Python o Node.js
- ✓ Leer y modificar un servicio ajeno
- ✓ Git

Datos

- ✓ SQL e índices a nivel de uso
- ✓ Transacciones

Sistemas

- ✓ HTTP
- ✓ Linux en terminal
- ✓ Docker a nivel de uso

Números

- ✓ Percentiles y distribuciones a nivel básico
- ✓ Leer gráficos de series de tiempo

Equipo y tiempo

- ✓ Computadora con 16 GB de memoria y 4 núcleos
- ✓ Seis horas de clase por semana
- ✓ Entre tres y cinco horas de laboratorio
- ✓ 10 semanas de compromiso

Costos de herramientas que asumes tú `US$0 a 15 aprox.`

Estos costos NO están incluidos en el precio del programa. Todo el software del laboratorio es gratuito.

- El laboratorio corre en tu máquina con contenedores: la aplicación de referencia, su base de datos, la caché y los generadores de carga, todo libre
- Si tu equipo no alcanza (se recomiendan 16 GB de memoria y 4 núcleos), una máquina virtual alquilada solo durante las prácticas cuesta del orden de US$15 en todo el programa
- Los costos de nube del laboratorio de costo por transacción se calculan con precios de lista, sin desplegar nada

PRECIO

60 horas para dejar de optimizar por opinión.

US$360

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

- ✓ 20 sesiones en vivo por Zoom (60 horas lectivas)
- ✓ Grabación de todas las clases con acceso por 12 meses
- ✓ Aplicación de referencia del caso con sus datos y escenarios de carga
- ✓ Versión con problemas sembrados para el proyecto final
- ✓ Plantilla de informe de rendimiento con percentiles, capacidad y costo
- ✓ Los 3 certificados progresivos
- ✓ Canal privado del grupo para dudas entre sesiones
- ✓ Actualizaciones del material por 12 meses

No incluye

- — Una máquina virtual alquilada si tu equipo no alcanza (rango de costos propios: US$0 a 15)
- — Exámenes ni certificaciones de terceros: este programa no prepara para ninguno
- — Herramientas comerciales de monitoreo de rendimiento: todo el laboratorio usa software libre

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: 20 sesiones de 3 horas, 2 veces por semana. Todas quedan grabadas y disponibles por 12 meses.

¿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 Miércoles y viernes 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.

¿Qué nivel necesito?

Un lenguaje backend con soltura, SQL e índices a nivel de uso, HTTP, Linux y Docker. No hace falta experiencia previa en pruebas de carga ni en perfilado.

¿En qué lenguaje está la aplicación del caso?

Los laboratorios de perfilado cubren la JVM y Python, y el método se aplica a cualquier lenguaje backend. Cada alumno puede repetir las técnicas en su propio stack con las herramientas equivalentes.

Ya cursé un programa de confiabilidad o de sistemas distribuidos. ¿Me sirve este?

Sí, y conviene que sepas qué se repite. Cerca de un tercio de las sesiones toca temas que esos programas tratan con otro objetivo: pruebas de carga bien modeladas, caché con estampidas, escalado, colas y contrapresión, CDN y capacidad. Aquí el centro es encontrar la causa del rendimiento con método y corregirla donde está, con otros laboratorios. Lo nuevo serían percentiles, USE y RED, perfilado, base de datos, contención, modelos de escalabilidad y costo por transacción. No hay descuento ni convalidación entre programas.

¿Necesito una cuenta de nube?

No. Todo corre en tu máquina con contenedores. Se recomienda un equipo con 16 GB de memoria y 4 núcleos, porque el generador de carga no debe competir con la aplicación; si no alcanza, el rango de costos propios es de US$0 a 15, por alquilar una máquina virtual solo durante las prácticas.

¿Por qué Valkey y no Redis?

Porque Valkey tiene licencia libre BSD y nació como bifurcación del código abierto de Redis justo antes de que Redis cambiara de licencia. Lo que se aprende sobre patrones de caché y estampidas aplica igual a cualquiera de los dos.

¿Se ve optimización de frontend?

No. El programa trabaja del servidor hacia atrás: aplicación, base de datos, caché, red y borde. La CDN se trata desde las cabeceras que controla el servidor.

¿Qué pasa si falto a una clase?

La grabación queda disponible y cada laboratorio tiene su punto de partida en el repositorio. Las mejoras se acumulan sobre la misma aplicación: faltar a varias sesiones seguidas sin recuperarlas te deja atrás en el proyecto final.

¿Los certificados tienen validez oficial?

No. Son certificados de Reactor Courses que acreditan las horas y competencias del programa. No son títulos universitarios ni certificaciones de terceros.

¿Cuánto tiempo debo dedicarle por semana?

Seis horas de clase más entre tres y cinco horas de laboratorio. Considera entre 9 y 11 horas semanales durante 10 semanas.

¿Qué métodos de pago aceptan?

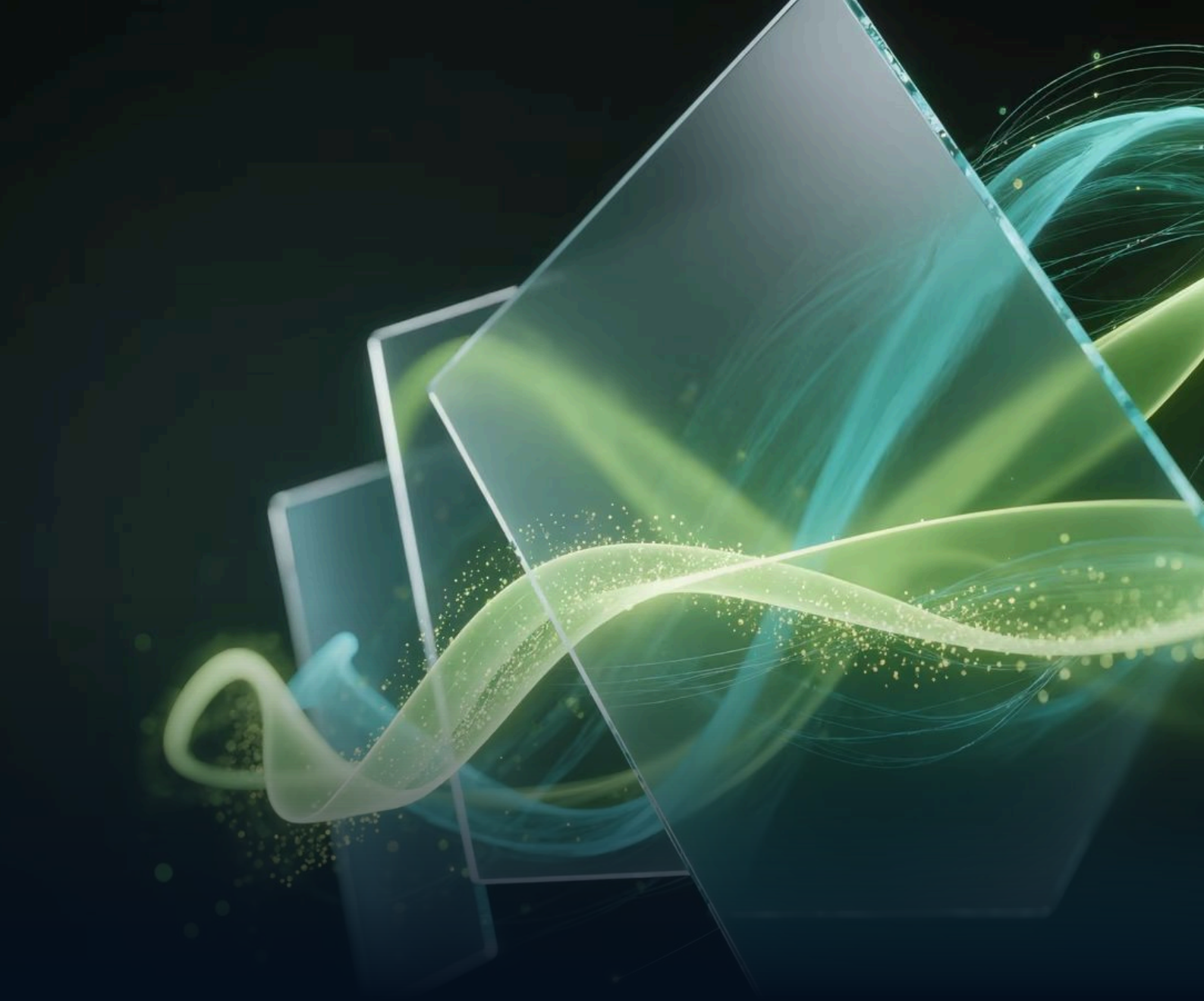
El pago lo procesa Paddle y acepta Tarjeta de crédito, Tarjeta de débito internacional, PayPal, Apple Pay, Google Pay. El cobro es en dólares. Importante: no procesamos Yape, Plin, OXXO, Mercado Pago ni transferencias locales. Si tu banco rechaza compras internacionales, habilítalas antes de pagar o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

¿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.

¿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.



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/performance-scalability-architecture.

`reactorcourses.com/programas/performance-scalability-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.