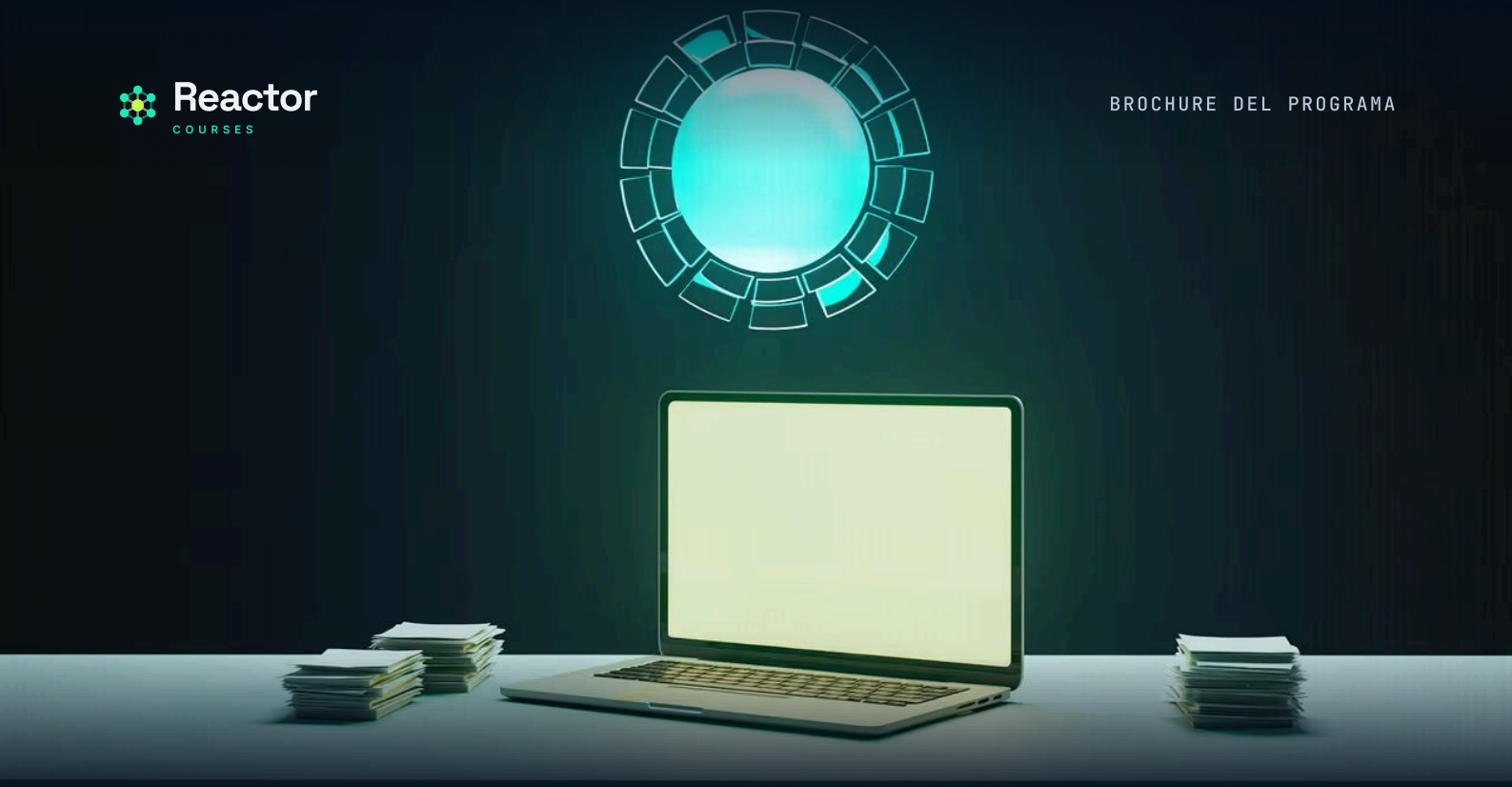
Reactor
COURSES

BROCHURE DEL PROGRAMA



ARQUITECTURA E INGENIERÍA · DISEÑO Y ESTRATEGIA DE ARQUITECTURA

Modernización de sistemas legados y patrones strangler

Cambiar el motor con el avión en vuelo

Moderniza un sistema que no se puede apagar: entiéndelo sin documentación, protégelo con pruebas de caracterización, extrae capacidades de a una y migra sus datos sin perder un registro, con tráfico real circulando todo el tiempo.

Lo que vas a lograr Aprendes a reemplazar un sistema antiguo por partes, sin apagarlo y sin perder datos, mientras sigue atendiendo a sus usuarios.

- Pruebas de caracterización
- Strangler fig
- Branch by abstraction
- Captura de cambios
- Reconciliación de datos
- Métricas DORA

horas en vivo	sesiones	semanas	errores permitidos durante cada corte
60	20	10	0

INICIO
Por confirmar

PRECIO
US$360 pago único · USD

MODALIDAD
Clases en vivo por Zoom · Avanzado

reactorcourses.com/programas/legacy-modernization-strangler-patterns

EL PROGRAMA

Reescribir todo es tentador. Casi siempre es la forma más cara de no terminar.

Moderniza un sistema que no se puede apagar: entiéndelo sin documentación, protégelo con pruebas de caracterización, extrae capacidades de a una y migra sus datos sin perder un registro, con tráfico real circulando todo el tiempo. Un programa de **60 horas** para quienes heredaron un sistema crítico en banca, seguros, retail o gobierno y tienen que modernizarlo sin que el negocio se entere.

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	Martes y jueves, 19:00–22:00 (GMT-5)
Precio	US$360 · pago único		

Los sistemas que sostienen el negocio no se pueden apagar, no tienen documentación confiable y guardan reglas que nadie recuerda haber escrito. Este programa enseña a modernizarlos desde adentro: entenderlos, protegerlos con pruebas antes de tocarlos, extraer capacidades de a una con tráfico real circulando, migrar sus datos sin perder un registro y demostrar el avance con números.

01

Entender y proteger

Arqueología, IA con verificación, pruebas de caracterización y estrategia

Mapa de dependencias y puntos calientes desde el código y su historial, asistentes de IA usados para leer código ajeno con su tasa de error medida, pruebas que fijan lo que el sistema hace hoy, costuras para poder cambiarlo y una estrategia por componente que se defiende ante quien paga.

- ✓ Mapa de dependencias y puntos calientes del historial
- ✓ IA para comprender código, con sus errores medidos
- ✓ Pruebas de caracterización y costuras
- ✓ Estrategia por componente y caso de negocio

02

Extraer y migrar

Strangler fig, anticorrupción, datos, ejecución paralela y corte

Monolito modular como paso intermedio, extracción con fachada de interceptación, capa anticorrupción, reemplazo de dependencias sin rama larga y banderas para encender y apagar. Los datos se migran con captura de cambios y reconciliación, la implementación nueva se compara en paralelo con la vieja y el corte final se hace sin un solo error.

- ✓ Strangler fig y capa anticorrupción con tráfico continuo
- ✓ Branch by abstraction y lanzamiento oscuro
- ✓ Captura de cambios y reconciliación registro a registro
- ✓ Ejecución paralela, integración con mainframe y métricas DORA

> Si no puedes demostrar que el sistema nuevo hace lo mismo que el viejo, no estás modernizando: estás apostando.

PARA QUIÉN ES

¿Es el programa indicado para ti?

Está pensado para quienes mantienen un sistema en producción y tienen que modernizarlo sin detener el negocio.

Es para ti si

- ✓ Desarrolladores y arquitectos de banca, seguros, retail o gobierno que heredaron un monolito crítico.
- ✓ Líderes técnicos que deben presentar un plan de modernización con riesgo acotado.
- ✓ Equipos que ya intentaron una reescritura completa y quieren otra estrategia.
- ✓ Ingenieros que tienen que migrar datos de un sistema a otro sin perder un registro.
- ✓ Responsables de un sistema que se integra con un mainframe que nadie va a tocar.

Te conviene otro programa

- – Quien no haya mantenido un sistema en producción: el programa parte de esa experiencia.
- – Quien busque diseñar sistemas nuevos desde cero.
- – Quien espere aprender COBOL o un mainframe: se integran, no se programan.
- – Quien quiera migrar a la nube tal cual, sin cambiar el sistema.
- – Quien ya cursó un programa de arquitectura de soluciones o de integración: cerca de un tercio de este programa —diagnóstico del legado, estrategias de migración, capa anticorrupción, captura de cambios, reconciliación e integración con sistemas centrales— toca temas que esos programas tratan en el diseño. 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 dos capacidades extraídas de un monolito real y sus datos migrados sin perder un registro.

El caso es Apache OFBiz, un ERP de código abierto con base de datos real y lógica de negocio que no es de juguete. Cada laboratorio deja una pieza de la modernización, siempre con tráfico simulado circulando.

01 **Línea base que no se puede empeorar** Errores, latencia y volumen del monolito con tráfico continuo, medidos antes de tocar nada.

02 **Mapa del sistema** Dependencias entre módulos y tablas cruzadas con los puntos calientes del historial, y dos capacidades elegidas con argumento.

03 **IA medida** Veinte reglas de negocio explicadas por un asistente, verificadas contra el código, con la tasa de error reportada.

04 **Red de seguridad** Pruebas de aprobación del cálculo de facturas y dos costuras introducidas sin cambiar ninguna aprobación.

05 **Uso real del sistema** Funciones que se usan de verdad identificadas, y dos que se decidió no migrar.

06 **Estrategia defendida** Estrategia por componente y caso de negocio por etapas, defendidos ante un comité que pregunta por qué no reescribir.

07 **Primera extracción sin errores** Catálogo extraído detrás de una fachada, con tráfico desviado hasta el cien por ciento y cero errores.

08 **Dependencia reemplazada** Motor de impuestos reemplazado con branch by abstraction y activado con banderas, primero en modo oscuro.

09 **Datos sincronizados** Facturas sincronizadas con captura de cambios, sin pérdidas ni duplicados ante la caída del sincronizador.

10 **Reconciliación y paralelo** Corrupciones de datos detectadas antes del corte, y una semana de ejecución paralela con cero diferencias sin explicar.

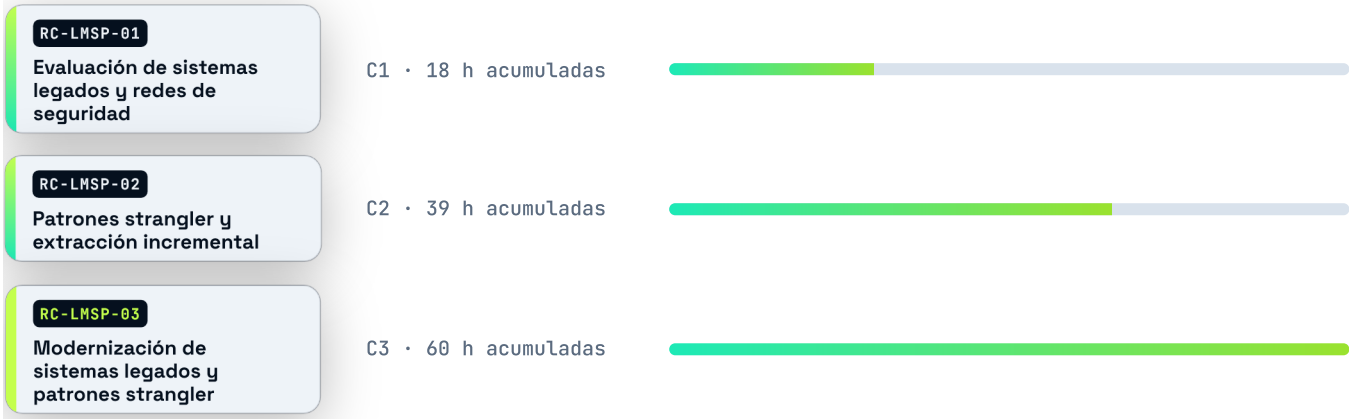
11 **Integración con el sistema central** Intercambio de archivos de longitud fija con reproceso sin duplicar movimientos.

12 **Corte final y métricas** Facturación cortada sin errores, código viejo retirado y métricas DORA antes y después.

RUTA DE CERTIFICACIÓN

3 credenciales progresivas, no una al final.

Cada etapa cierra un bloque completo: primero entender y proteger el sistema, después extraer capacidades, y al final migrar los datos y cortar. Son credenciales de Reactor Courses, no certificaciones de terceros.



CERTIFICACIÓN INTEGRAL

Legacy Modernization & Strangler Patterns — Cambiar el motor con el avión en vuelo (60 horas)

Requiere entregar las dos extracciones con sus pruebas, la migración de datos reconciliada y el corte final sin errores. 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 **ENTENDER LO QUE HAY** — 9 h · 3 sesiones

01 **El sistema que no se puede apagar**

por qué las reescrituras completas suelen fallar · el monolito del caso: Apache OFBiz con su base de datos y su lógica de negocio · datos sintéticos realistas y tráfico simulado continuo · qué significa cero tiempo de inactividad para el negocio · riesgo de cambiar frente a riesgo de no cambiar · el entorno local con el monolito, la base y el generador de tráfico

LABORATORIO: Levantar el monolito con datos sintéticos y tráfico continuo, y registrar la línea base de errores, latencia y volumen que ninguna extracción posterior podrá empeorar

02 **Arqueología de código y mapa de dependencias**

mapa de módulos y dependencias desde el código · puntos calientes por frecuencia de cambio en el historial de Git · tablas compartidas por varios módulos · código muerto y código que nadie se anima a tocar · entrevista al equipo que lo mantiene · candidatos a extracción por valor y por riesgo

LABORATORIO: Construir el mapa de dependencias de OFBiz entre módulos y tablas, cruzarlo con los puntos calientes del historial de Git, y elegir con argumento las dos capacidades a extraer en el programa

03 **Asistentes de IA para comprender código heredado, con verificación**

qué hace bien un asistente al leer código ajeno y qué inventa · explicaciones de reglas de negocio verificadas contra el código · pruebas como verificación de lo que dice el asistente · tasa de error medida en un conjunto de preguntas sobre el código · datos sensibles del código y su exposición a un proveedor · documentación generada que se mantiene viva

LABORATORIO: Preguntar a un asistente de IA veinte reglas de negocio del módulo de facturación de OFBiz, verificar cada respuesta contra el código y con una prueba, y reportar la tasa de respuestas incorrectas o inventadas

MÓDULO 2 LA RED DE SEGURIDAD — 9 h · 3 sesiones

04 **Pruebas de caracterización**

probar lo que el sistema hace, no lo que debería hacer · pruebas de aprobación con ApprovalTests · capturar entradas y salidas reales como casos · normalizar fechas, identificadores y otros datos variables · cobertura del comportamiento que se va a extraer · cuando la prueba encuentra un error histórico que el negocio usa

LABORATORIO: Cubrir el cálculo de facturas de OFBiz con pruebas de aprobación a partir de casos reales capturados, y demostrar que un cambio de una línea en el redondeo hace fallar exactamente las pruebas afectadas

05 **Costuras y refactor seguro**

costura: el lugar donde se puede cambiar comportamiento sin editar · romper dependencias para poder probar · extraer un método o una interfaz sin cambiar comportamiento · refactor guiado por las pruebas de caracterización · cambios pequeños y frecuentes en un código frágil · cuándo no refactorizar

LABORATORIO: Introducir dos costuras en el módulo de facturación para aislarlo de la base de datos en las pruebas, sin que cambie ninguna aprobación, en una secuencia de cambios pequeños con la integración continua en verde

06 **El comportamiento real en producción**

registros como especificación de facto · rutas que se usan y rutas que nadie usa · consultas reales a la base y su frecuencia · instrumentar sin modificar el negocio · identificar a los consumidores del monolito · datos para decidir qué no migrar

LABORATORIO: Instrumentar el monolito sin tocar su lógica, identificar con una semana de tráfico simulado qué funciones del catálogo se usan de verdad y a quién sirven, y declarar dos funciones que no se migrarán

MÓDULO 3 DECIDIR Y JUSTIFICAR — 3 h · 1 sesión

07 **Estrategia por componente y caso de negocio**

las 7 R de AWS y las 8 estrategias de Microsoft: equivalencias y diferencias · estrategia por componente, no por sistema · valor, riesgo y costo por etapa · criterios para detener una modernización a mitad de camino · monolito modular como destino legítimo · presentar el caso a quien paga

LABORATORIO: Clasificar cada componente del caso con una estrategia de migración, armar el caso de negocio por etapas con valor, riesgo y costo, y presentarlo en diez minutos a un comité que pregunta por qué no reescribir todo

MÓDULO 4 EXTRAER DE A UNA CAPACIDAD

18 h · 6 sesiones

08 Monolito modular como paso intermedio

límites de módulo dentro del mismo proceso · reglas de dependencia verificadas automáticamente · datos propios de cada módulo dentro de la misma base · extraer un módulo antes de extraer un servicio · costo y beneficio de quedarse en el monolito modular · señales de que hay que seguir

LABORATORIO: Convertir el catálogo de OFBiz en un módulo con límites verificados por una prueba de arquitectura, y demostrar que ninguna otra parte del monolito accede a sus tablas directamente

09 Strangler fig: interceptar y desviar

fachada de interceptación delante del monolito · desviar tráfico por ruta, por cliente o por porcentaje · convivencia de la implementación vieja y la nueva · volver atrás en segundos · el estrangulamiento que nunca termina · medir el porcentaje de tráfico ya estrangulado

LABORATORIO: Extraer la consulta de catálogo y precios a un servicio nuevo detrás de una fachada de interceptación, desviar el tráfico por porcentaje hasta el cien y demostrar con el tráfico continuo cero errores durante todo el corte

10 Capa anticorrupción

el modelo del monolito no debe filtrarse al servicio nuevo · traducción de modelos en la frontera · fachada, adaptador y traductor · errores y códigos del sistema viejo · la capa anticorrupción como pieza temporal o permanente · pruebas de la traducción

LABORATORIO: Construir la capa anticorrupción entre el servicio de catálogo y el monolito con pruebas de traducción, y demostrar que un cambio de nombre de campo en OFBiz no requiere tocar el servicio nuevo

11 Branch by abstraction

reemplazar una dependencia interna sin rama larga · abstracción, implementación vieja y nueva conviviendo · migrar a los llamadores de a uno · eliminar la implementación vieja y la abstracción · integración continua durante todo el reemplazo · diferencia con el estrangulamiento

LABORATORIO: Reemplazar el motor de cálculo de impuestos de la facturación por uno nuevo con branch by abstraction, integrando a la rama principal en cada paso y con las pruebas de caracterización en verde de principio a fin

12 Banderas y lanzamiento oscuro

banderas para elegir implementación en tiempo de ejecución · lanzamiento oscuro: ejecutar sin mostrar · segmentos de usuarios y activación gradual · deuda de banderas y su eliminación · OpenFeature con flagd · apagar la implementación nueva ante un error

LABORATORIO: Activar con banderas de OpenFeature el nuevo cálculo de impuestos primero en modo oscuro y después para un segmento, y demostrar que apagar la bandera restaura el comportamiento anterior en menos de un minuto

13 Segunda extracción con datos propios

extraer una capacidad que escribe, no solo lee · la base de datos propia del servicio nuevo · quién es dueño del dato durante la convivencia · transacciones que antes eran locales · consistencia aceptable para el negocio · orden de los pasos de la extracción

LABORATORIO: Extraer la facturación a un servicio con base de datos propia, definiendo quién es dueño de cada dato durante la convivencia, y dejarla lista para la migración de datos de las sesiones siguientes con el tráfico continuo sin errores

MÓDULO 5 MIGRAR LOS DATOS SIN PERDER UN REGISTRO

9 h · 3 sesiones

14 **Por qué no escrituras duales: captura de cambios**

escrituras duales y su falla silenciosa · captura de cambios del registro de la base con Debezium · carga inicial y continuidad de los cambios · esquema viejo traducido al esquema nuevo · atraso de la réplica y su efecto en el corte · qué hacer con los borrados

LABORATORIO: Sincronizar las facturas del monolito hacia la base del servicio nuevo con Debezium, con carga inicial y cambios continuos, y demostrar que una falla del sincronizador a mitad de camino no pierde ni duplica facturas

15 **Reconciliación**

conteos, sumas de control y comparación registro a registro · diferencias esperadas por traducción y diferencias reales · reconciliación continua durante la convivencia · informe de diferencias para el negocio · corrección de diferencias sin tocar a mano la base · criterio de cero diferencias para cortar

LABORATORIO: Implementar la reconciliación continua entre la base vieja y la nueva con sumas de control y comparación por registro, inyectar tres corrupciones de datos y demostrar que las detecta y explica antes del corte

16 **Ejecución paralela y comparación de resultados**

ejecutar las dos implementaciones con la misma entrada · comparar resultados sin afectar al usuario · experimentos en producción al estilo Scientist · diferencias aceptables y diferencias que bloquean · costo de ejecutar dos veces · decidir el corte con datos

LABORATORIO: Ejecutar en paralelo la facturación vieja y la nueva sobre una semana de tráfico simulado, reportar las diferencias por tipo y decidir el corte solo cuando la tasa de diferencias no explicadas sea cero

MÓDULO 6 INTEGRAR LO QUE NO SE TOCA

3 h · 1 sesión

17 **Integración con mainframe por archivos, colas o APIs**

el sistema que se integra pero no se modifica · intercambio de archivos de longitud fija y codificaciones viejas · colas de mensajes como frontera · APIs expuestas por el propio mainframe · ventanas de proceso por lotes y su efecto en el diseño · pruebas contra un simulador del sistema central

LABORATORIO: Integrar el servicio de facturación con un simulador de sistema central que solo acepta archivos de longitud fija en una codificación antigua, y demostrar que un archivo rechazado se reprocesa sin duplicar movimientos

MÓDULO 7 MEDIR EL AVANCE

3 h · 1 sesión

18 **Métricas de entrega y de riesgo**

las cinco métricas de DORA: tiempo de cambio, frecuencia, recuperación, tasa de fallas y retrabajo · línea base antes de modernizar · porcentaje del sistema ya modernizado · riesgo residual y deuda de convivencia · informe de avance para el negocio · cuándo declarar terminada una etapa

LABORATORIO: Medir las métricas de DORA del caso antes y después de las extracciones con los datos del propio repositorio y la tubería, y presentar el informe de avance con riesgo residual y deuda de convivencia

MÓDULO 8 PROYECTO INTEGRADOR

6 h · 2 sesiones

19 **Corte final sin tiempo de inactividad**

lista de verificación del corte · corte de la facturación con tráfico continuo · reconciliación final · retiro del código viejo · criterio de aborto y vuelta atrás ensayados · registro para la defensa

LABORATORIO: Ejecutar el corte final de la facturación con el tráfico continuo, reconciliación en cero, retiro del código viejo del monolito y la vuelta atrás ensayada, sin un solo error registrado durante el corte

20 **Defensa de la modernización**

evaluación, caso de negocio y estrategia · red de seguridad y extracciones · migración de datos y reconciliación · métricas antes y después · lo que no se modernizó y por qué · preguntas cruzadas del grupo

LABORATORIO: Defender ante el grupo la modernización del caso con las dos extracciones, la migración de datos reconciliada, las métricas antes y después y lo que se decidió no modernizar

LO QUE VAS A USAR

Software libre y un monolito real, elegidos por el paso de la modernización que resuelven.

El sistema heredado

- Apache OFBiz
- Su base de datos relacional
- Simulador de sistema central

Red de seguridad

- ApprovalTests
- Pruebas de arquitectura

Extracción

- Fachada de interceptación
- OpenFeature con flagd

Datos

- Debezium
- Reconciliación con sumas de control

Medición

- Métricas DORA
- Generador de tráfico continuo

Comprensión

- Historial de Git como fuente de puntos calientes
- Un asistente de IA de código

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, un monolito real y tráfico circulando mientras lo cambias.

01 **Clases en vivo por Zoom** 20 sesiones de 3 horas, 2 veces por semana. Se trabaja en vivo sobre el monolito y se revisa cada paso con el grupo.

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 patrón se aplica el mismo día sobre el sistema del caso, con su criterio de aceptación.

04 **Tráfico siempre encendido** Cada extracción se valida con tráfico simulado continuo: el criterio es cero errores durante el corte.

05 **La IA con su tasa de error** Los asistentes de IA se usan para leer código ajeno, y cada respuesta se verifica contra el código y una prueba.

06 **Un sistema de verdad** Un ERP de código abierto con base de datos real y reglas de negocio que no son de juguete, con licencia Apache.

Horarios · Primer grupo

Martes	19:00 - 22:00
Jueves	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 levanta el monolito con tráfico y mide lo que no se podrá empeorar.

Experiencia

- ✓ Haber mantenido un sistema en producción
- ✓ Pruebas automatizadas básicas

Programación

- ✓ Un lenguaje backend como Java, C# o similar
- ✓ Leer código que no escribiste
- ✓ Git

Datos

- ✓ SQL
- ✓ Transacciones en una base de datos

Herramientas

- ✓ Docker a nivel de uso
- ✓ Terminal

Equipo y tiempo

- ✓ Computadora con 16 GB de memoria
- ✓ 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: el monolito de código abierto, su base de datos, el generador de tráfico y los servicios nuevos
- Si tu equipo no alcanza (se recomiendan 16 GB de memoria), una máquina virtual alquilada solo durante las prácticas cuesta del orden de US$15 en todo el programa
- Para el laboratorio con un asistente de IA puedes usar el que ya tengas o un modelo abierto local; si usas uno de pago, su costo no está incluido

PRECIO

60 horas para cambiar el motor sin bajar a nadie del avión.

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
- ✓ Entorno del monolito con datos sintéticos realistas y generador de tráfico continuo
- ✓ Simulador de sistema central para la integración por archivos
- ✓ Plantillas de evaluación, caso de negocio y lista de verificación del corte
- ✓ 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
- — Migración de infraestructura a la nube: aquí se moderniza el sistema, no dónde corre

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 Martes y jueves 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?

Haber mantenido un sistema en producción, manejar un lenguaje backend como Java o C#, SQL y pruebas automatizadas básicas. El monolito del caso está en Java, pero el criterio aplica a cualquier tecnología.

¿Se aprende COBOL o a programar un mainframe?

No. El mainframe se trata como un sistema que se integra y no se modifica: archivos, colas y APIs que expone, simulados en el laboratorio.

Ya cursé un programa de arquitectura de soluciones o de integración. ¿Me sirve este?

Sí, y conviene que sepas qué se repite. Cerca de un tercio de las sesiones toca temas que esos programas tratan en el diseño: diagnóstico del legado, estrategias de migración, capa anticorrupción, captura de cambios, reconciliación e integración con sistemas centrales. Aquí todo se ejecuta en código y en datos, con tráfico circulando, y los laboratorios son otros. Lo nuevo serían las pruebas de caracterización, las costuras, branch by abstraction, la ejecución paralela, la IA con verificación y el corte sin errores. No hay descuento ni convalidación entre programas.

¿Se usa IA para modernizar?

Se usa para entender código ajeno, y se mide cuánto se equivoca. Los laboratorios no aceptan una explicación del asistente sin verificarla contra el código y una prueba. La IA no moderniza sola.

¿Necesito una cuenta de nube?

No. Todo corre en tu máquina con contenedores. Se recomienda un equipo con 16 GB de memoria; 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 Apache OFBiz como caso?

Porque es un ERP de código abierto, con licencia Apache, mantenido activamente, con una base de datos real y reglas de negocio —pedidos, facturación, inventario— que no son de juguete. Modernizar algo de ese tamaño se parece a lo que vas a encontrar en tu trabajo.

¿Qué pasa si falto a una clase?

La grabación queda disponible y cada laboratorio tiene su punto de partida en el repositorio. Las extracciones se construyen una sobre otra: faltar a varias sesiones seguidas sin recuperarlas te deja sin piezas para el corte 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?

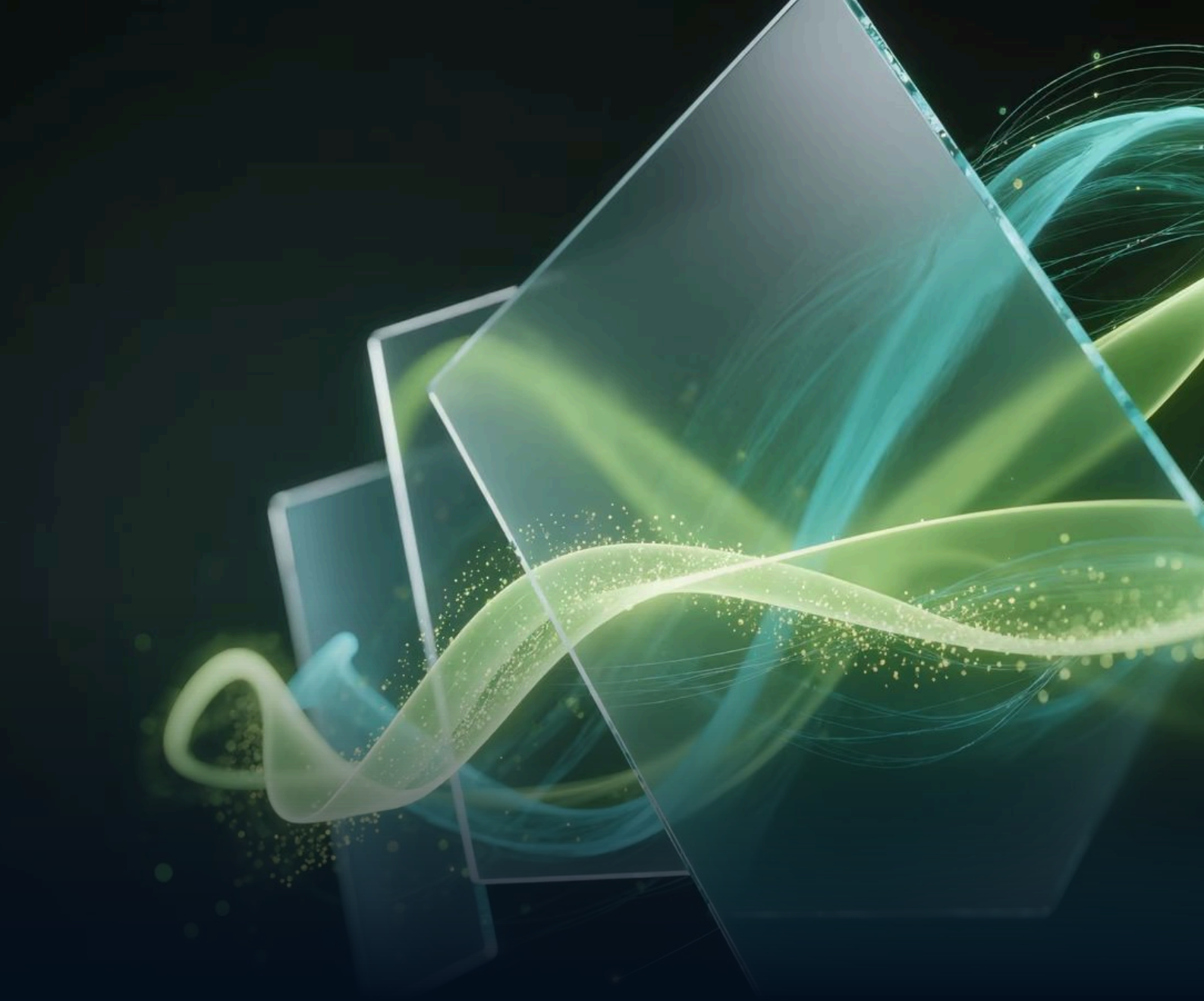
El pago lo procesa Paddle y acepta Tarjeta de crédito, Tarjeta de débito internacional, PayPal, Apple Pay, Google Pay. El cobro es en dólares. Importante: no procesamos Yape, Plin, OXXO, Mercado Pago ni transferencias locales. Si tu banco rechaza compras internacionales, habilítalas antes de pagar o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

`reactorcourses.com/programas/legacy-modernization-strangler-patterns`

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.