Reactor
COURSES

BROCHURE DEL PROGRAMA



ARQUITECTURA E INGENIERÍA · INTEGRACIÓN Y DATOS

Arquitectura de API e integración

Contratos que sobreviven a quien los consume

Diseña, asegura, versiona y gobierna los contratos entre sistemas que no controlas —APIs, eventos, webhooks y conexiones con sistemas heredados y SaaS— y retíralos sin romper a quien todavía los usa.

Lo que vas a lograr Aprendes a diseñar, asegurar y cambiar las APIs que conectan sistemas entre sí y con terceros, sin romperle nada a quien ya las usa.

- OpenAPI y AsyncAPI
- OAuth 2 y OWASP API
- Gateway y cuotas
- Versionado y deprecación
- Webhooks y legado
- Pruebas de contrato

horas en vivo	sesiones	semanas	sistemas que vas a integrar
60	20	10	3

INICIO
Por confirmar

PRECIO
US$360 pago único · USD

MODALIDAD
Clases en vivo por Zoom · Intermedio

reactorcourses.com/programas/api-integration-architecture

EL PROGRAMA

Publicar una API es fácil. Cambiarla sin romper a quien ya la usa es el trabajo de verdad.

Diseña, asegura, versiona y gobierna los contratos entre sistemas que no controlas —APIs, eventos, webhooks y conexiones con sistemas heredados y SaaS— y retíralos sin romper a quien todavía los usa. Un programa de **60 horas** para quienes publican APIs que usan otros equipos u otras empresas, y para quienes tienen que conectar un ERP viejo, un SaaS y una app sin que nadie se entere del cambio.

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	Intermedio
Inicio	Por confirmar	Horario	Miércoles y viernes, 19:00–22:00 (GMT-5)
Precio	US$360 · pago único		

Una API interna se vuelve un contrato el día que otro equipo depende de ella, y una API externa lo es desde el primer socio. A partir de ahí cada cambio tiene consumidores que no ves, horarios que no controlas y costos que no pagas tú. Este programa trata el contrato como producto: cómo se diseña, se asegura, se observa, se versiona, se retira y se integra con sistemas que no hablan tu idioma.

01

Diseñar y proteger el contrato

Estilo, OpenAPI, AsyncAPI, seguridad y gateway

Elegir el estilo de interfaz con criterio, escribir el contrato antes que el código, normalizar errores, paginar, evitar duplicados y ediciones pisadas, y proteger la API con OAuth, confianza entre organizaciones y un gateway con planes por consumidor. La seguridad se prueba atacando la propia API.

- OpenAPI 3.2 y AsyncAPI 3 con contrato primero
- Problem Details, paginación por cursor, idempotencia y concurrencia optimista
- OAuth 2, mTLS y el OWASP API Security Top 10 atacado y bloqueado
- Cuotas, límites y caché por consumidor en el gateway

02

Evolucionar e integrar

Versionado, deprecación, legado, webhooks y pruebas de contrato

Detectar cambios incompatibles en la tubería, deprecar y retirar de punta a punta sin dejar a nadie colgado, integrar un ERP que solo habla archivos y SOAP, recibir webhooks que pueden llegar falsos, repetidos o nunca, y verificar contratos con pruebas dirigidas por el consumidor.

- Compatibilidad verificada automáticamente y estrategia de versionado
- Deprecation y Sunset con el consumidor antiguo funcionando hasta el final
- Patrones de integración con Apache Camel y capa anticorrupción
- Webhooks firmados e idempotentes, y pruebas de contrato con Pact

> Todo comportamiento observable de una API termina siendo parte del contrato, lo hayas documentado o no.

PARA QUIÉN ES

¿Es el programa indicado para ti?

Está pensado para desarrolladores que ya construyen y consumen APIs y quieren diseñarlas para que otros equipos dependan de ellas sin sorpresas.

Es para ti si

- ✓ Desarrolladores backend que publican APIs consumidas por otros equipos o por clientes.
- ✓ Arquitectos de integración que conectan ERP, CRM y servicios SaaS.
- ✓ Equipos que mantienen una API con consumidores que no pueden romper.
- ✓ Responsables de una plataforma de APIs que quieren reglas de diseño que se cumplan solas.
- ✓ Ingenieros que heredaron integraciones por archivos, SOAP o base de datos compartida y tienen que ordenarlas.

Te conviene otro programa

- – Quien no haya construido ni consumido una API HTTP: métodos, códigos y JSON se dan por sabidos.
- – Quien busque un curso de una plataforma de integración comercial: aquí se usan herramientas libres y se enseña el criterio.
- – Quien espere teoría de sistemas distribuidos: la consistencia se trata desde el contrato, no desde el algoritmo.
- – Quien quiera aprender a dividir un sistema en microservicios: eso es otro problema.
- – Quien ya cursó un programa de arquitectura de soluciones o de plataforma: cerca de un tercio de este programa —contratos OpenAPI y AsyncAPI, OAuth, cuotas, compatibilidad y deprecación— toca temas que esos programas tratan dentro de una solución o una plataforma. 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 tres sistemas integrados por contrato y una deprecación ejecutada sin romper a nadie.

Cada laboratorio deja una pieza de la integración del caso con su criterio de aceptación cumplido. Juntas forman el proyecto que defiendes en la última sesión.

01 **Inventario de consumidores** Consumidores reales de la API sacados de los registros del gateway, y diez cambios clasificados con quién romperían.

02 **Contrato primero** API de pedidos en OpenAPI 3.2 sin errores de estilo, consumida por la app contra un simulador antes de implementarse.

03 **Operaciones seguras de repetir** Errores normalizados, paginación por cursor, idempotencia y concurrencia optimista demostradas con reenvíos y ediciones simultáneas.

04 **Eventos con contrato** Eventos de socios descritos en AsyncAPI 3 y validados al publicarse, con mensajes inválidos rechazados.

05 **Autorización que separa socios** Tokens válidos que igual no dan acceso a datos de otro socio ni a operaciones fuera de sus alcances.

06 **Ataques bloqueados** Seis riesgos del OWASP API Security Top 10 explotados, corregidos y bloqueados con la misma prueba automatizada.

07 **Gateway con planes** Tres planes de consumo con cuota, límite y caché, y un socio excedido que no degrada a los demás.

08 **Compatibilidad en la tubería** Cambios incompatibles bloqueados automáticamente y estrategia de versionado elegida con su costo.

09 **Deprecación completa** Operación deprecada y retirada con Deprecation y Sunset, con el consumidor antiguo funcionando hasta la fecha.

10 **Legado integrado** ERP por archivos y SOAP detrás de una capa anticorrupción, con conciliación que no pierde ni duplica pedidos.

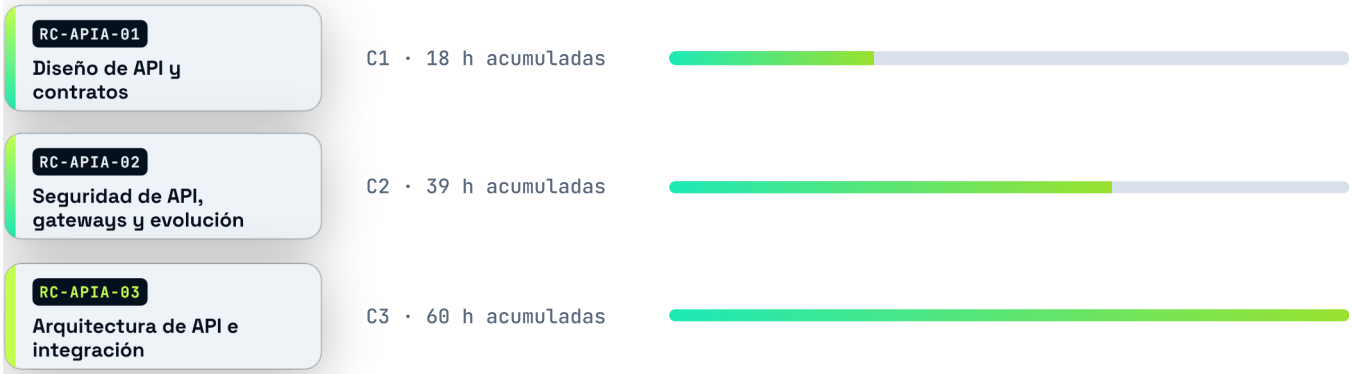
11 **Webhooks confiables** Webhooks falsificados, repetidos y perdidos que terminan en el estado correcto.

12 **Pruebas de contrato** Contratos de Pact verificados en la tubería que bloquean el cambio que rompe a un consumidor.

RUTA DE CERTIFICACIÓN

3 credenciales progresivas, no una al final.

Cada etapa cierra un bloque completo: primero diseñar contratos, después asegurarlos, operarlos y hacerlos evolucionar, y al final integrar sistemas reales. Son credenciales de Reactor Courses, no certificaciones de terceros.



CERTIFICACIÓN INTEGRAL

API & Integration Architecture — Contratos que sobreviven a quien los consume (60 horas)

Requiere entregar la integración de los tres sistemas, la deprecación ejecutada y las pruebas de contrato y de seguridad en la tubería. 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 **EL CONTRATO COMO PRODUCTO** — 6 h · 2 sesiones

01 **Contratos que sobreviven a quien los consume**

el caso: una cadena de farmacias con un ERP heredado, un SaaS de fidelización y una app móvil · consumidores que no controlas y la ley de Hyrum · la API como producto: dueño, ciclo de vida y usuarios · cambio compatible, incompatible y peligroso · registros del gateway como inventario real de consumidores · costo de romper a un socio frente a costo de no cambiar · el entorno local: sistemas simulados, gateway e identidad

LABORATORIO: Levantar los tres sistemas del caso, inventariar los consumidores de la API existente desde los registros del gateway y clasificar diez cambios propuestos como compatibles o incompatibles, con el consumidor concreto que rompería cada uno

02 **Elegir el estilo de interfaz**

REST, gRPC, GraphQL y mensajería asíncrona como opciones · criterios: tipo de consumidor, latencia, herramientas y evolución · consulta flexible frente a contrato estable · integración síncrona y su acoplamiento temporal · archivos y lotes como integración legítima · registro de decisión con alternativas descartadas

LABORATORIO: Evaluar los cuatro estilos para tres integraciones del caso con una matriz por consumidor y restricción, y prototipar la alternativa descartada más cercana para medir la diferencia en latencia y en cambios del cliente

MÓDULO 2 DISEÑO DEL CONTRATO — 12 h · 4 sesiones

03 **Recursos y semántica HTTP**

modelar recursos a partir de las operaciones del negocio · métodos seguros e idempotentes según HTTP · códigos de estado que el consumidor puede automatizar · cabeceras de caché, ETag y respuestas condicionales · operaciones largas: 202 y recurso de estado · acciones que no encajan en CRUD

LABORATORIO: Rediseñar la API de pedidos del caso con recursos, métodos y códigos correctos, y demostrar con pruebas que la caché condicional ahorra transferencia y que una operación larga se consulta sin bloquear al cliente

04 **OpenAPI 3.2 y contrato primero**

OpenAPI 3.2: novedades frente a 3.1 y compatibilidad · esquemas, componentes reutilizables y ejemplos · contrato antes que implementación · reglas de estilo como código con Spectral · servidor simulado con Microcks para el consumidor · generación de código: qué conviene generar y qué no

LABORATORIO: Escribir el contrato de la API de pedidos en OpenAPI 3.2 sin una línea de implementación, pasarlo por reglas propias de Spectral sin errores, y hacer que el equipo de la app móvil desarrolle contra el simulador de Microcks con una prueba que pase

05 **Errores, paginación, idempotencia y concurrencia**

errores normalizados con Problem Details (RFC 9457) · paginación por cursor frente a paginación por desplazamiento · filtros y ordenamiento sin inventar un lenguaje de consulta · clave de idempotencia: borrador del IETF y diseño propio · concurrencia optimista con ETag e If-Match · actualización parcial con PATCH · límites de tamaño y de página como parte del contrato

LABORATORIO: Implementar en la API de pedidos errores RFC 9457, paginación por cursor, clave de idempotencia y concurrencia optimista, y demostrar que un pedido reenviado no se duplica y que dos ediciones simultáneas no se pisan

06 **AsyncAPI para eventos de integración**

evento de integración frente a evento de dominio interno · AsyncAPI 3: canales, operaciones y mensajes · esquema del mensaje y sus versiones · validación del mensaje en tiempo de ejecución · documentación generada para el consumidor · simulación de eventos con Microcks

LABORATORIO: Describir en AsyncAPI 3 los eventos que la cadena publica a sus socios, validar cada mensaje contra el esquema al publicarlo y demostrar que un mensaje inválido se rechaza antes de llegar al consumidor

MÓDULO 3 SEGURIDAD DE LA API — 9 h · 3 sesiones

07 **OAuth 2 y OpenID Connect entre organizaciones**

flujos vigentes: código de autorización con PKCE y credenciales de cliente · OAuth 2.1: qué consolida y su estado de borrador · alcances diseñados por operación de negocio · validación del token en el gateway y en el servicio · audiencia, expiración y rotación de claves de firma · Keycloak como servidor de autorización del laboratorio · token de un socio usado para datos de otro socio

LABORATORIO: Proteger la API con Keycloak y validación de tokens en el gateway, y demostrar que el token del socio A, aun válido, no permite leer pedidos del socio B ni operaciones fuera de sus alcances

08 **OWASP API Security Top 10 atacado y bloqueado**

autorización a nivel de objeto y de propiedad (API1 y API3) · autorización a nivel de función (API5) · consumo de recursos sin límite (API4) · flujos de negocio sensibles (API6) · SSRF desde una API que acepta URL (API7) · inventario de API olvidadas (API9) y consumo inseguro de terceros (API10) · pruebas automáticas basadas en propiedades con Schemathesis

LABORATORIO: Atacar la propia API del caso con seis riesgos del OWASP API Security Top 10 2023, documentar cada explotación exitosa, corregirla y demostrar con la misma prueba automatizada en Schemathesis que queda bloqueada

09 **Confianza entre organizaciones**

mTLS con socios: certificados, rotación y revocación · un cliente, un secreto: credenciales por socio · listas de permitidos y su mantenimiento · firma de mensajes cuando TLS no alcanza · registro de auditoría por socio · incorporación y baja de un socio

LABORATORIO: Incorporar un socio nuevo con mTLS y credenciales propias, rotar su certificado sin interrupción y darlo de baja, verificando en el registro de auditoría que después de la baja ninguna llamada suya es aceptada

MÓDULO 4 GATEWAY Y OPERACIÓN — 6 h · 2 sesiones

10 **Gateway: cuotas, límites y caché por consumidor**

qué va en el gateway y qué no · Apache APISIX: rutas, consumidores y complementos · planes por consumidor: cuota mensual y límite por segundo · 429, Retry-After y el borrador de cabeceras RateLimit · caché en el borde y su invalidación · transformación de peticiones sin esconder lógica de negocio · por qué no Kong OSS en este programa

LABORATORIO: Configurar en APISIX tres planes de consumo con cuota, límite y caché, y demostrar que el socio que excede su plan recibe 429 con Retry-After mientras los otros dos mantienen su latencia

11 **Observar la API por consumidor**

métricas por consumidor, por operación y por versión · trazas que cruzan el gateway · detectar al consumidor que todavía usa lo deprecado · errores del cliente frente a errores del servicio · informes de uso para el dueño de la API · datos personales en los registros del gateway

LABORATORIO: Construir el tablero de uso por consumidor y por versión desde la telemetría del gateway, identificar al consumidor que más errores del cliente genera y entregarle un informe con las tres correcciones que le pedirías

MÓDULO 5 EVOLUCIÓN SIN ROMPER — 6 h · 2 sesiones

12 **Compatibilidad y versionado**

reglas de compatibilidad para lectores tolerantes · versionado por ruta, por cabecera o sin versiones · detección automática de cambios incompatibles con oasdiff · campos opcionales, valores por defecto y enumeraciones que crecen · convivencia de versiones y su costo · cuándo una versión nueva es en realidad otra API

LABORATORIO: Integrar oasdiff a la tubería del contrato, demostrar que bloquea tres cambios incompatibles y deja pasar dos compatibles, y elegir la estrategia de versionado del caso con su costo de convivencia estimado

13 **Deprecación y retiro de punta a punta**

cabecera Deprecation (RFC 9745) y cabecera Sunset (RFC 8594) · calendario de deprecación y período de gracia · comunicación a consumidores identificados por el uso · migración asistida y guía de cambios · respuesta después del retiro: 410 y enlace a la alternativa · el consumidor antiguo que nadie recuerda

LABORATORIO: Deprecar una operación de la API con Deprecation y Sunset, migrar a los consumidores activos según el uso medido, y demostrar que el consumidor antiguo sigue funcionando hasta la fecha de retiro y recibe 410 con la alternativa después

MÓDULO 6 INTEGRACIÓN CON LEGADO, SAAS Y SOCIOS

12 h · 4 sesiones

14 Patrones de integración empresarial con Apache Camel

patrones como vocabulario: canal, enrutador, traductor y agregador · divisor, enriquecedor y filtro de mensajes · canal de mensajes fallidos y reintento · rutas declarativas en YAML con Camel · transformación de formatos sin perder trazabilidad · cuándo un integrador central se vuelve el cuello de botella

LABORATORIO: Construir con Camel una ruta que reciba pedidos de la app, los divida por farmacia, los enriquezca con stock y los enrute al ERP, enviando a un canal de fallidos los mensajes inválidos sin detener el resto

15 Integración con el sistema heredado

el ERP que solo habla archivos y SOAP · intercambio de archivos: llegada, validación, acuse y reproceso · SOAP y WSDL desde una integración moderna · capa anticorrupción que traduce el modelo viejo · base de datos compartida y por qué evitarla · conciliación entre lo enviado y lo registrado

LABORATORIO: Integrar el ERP del caso por archivos y por SOAP detrás de una capa anticorrupción, y demostrar con una conciliación diaria que ningún pedido se pierde ni se duplica cuando un archivo llega corrupto o repetido

16 Webhooks confiables

webhook entrante y saliente: responsabilidades de cada lado · firma del mensaje y especificación Standard Webhooks · protección contra repetición con marca de tiempo · reintentos con retroceso y desactivación del destino · recepción idempotente y procesamiento asincrónico · reconciliación cuando un webhook nunca llegó

LABORATORIO: Recibir los webhooks del SaaS de fidelización con verificación de firma según Standard Webhooks, protección contra repetición y procesamiento idempotente, y demostrar que un webhook falsificado, uno repetido y uno perdido terminan en el estado correcto

17 Consumir APIs de terceros sin quedar a su merced

autenticación ante el SaaS y rotación de credenciales · límites de tasa del tercero y espera respetuosa · cambios no anunciados del proveedor y cómo detectarlos · aislar al proveedor detrás de un adaptador propio · entorno de pruebas del proveedor y sus diferencias con producción · consumo inseguro de APIs (API10) en la práctica

LABORATORIO: Encapsular el SaaS del caso detrás de un adaptador con límites y reintentos, y demostrar con WireMock que un cambio no anunciado en su respuesta se detecta con una alerta y no rompe a los consumidores internos

MÓDULO 7 VERIFICACIÓN Y GOBIERNO

6 h · 2 sesiones

18 Pruebas de contrato dirigidas por el consumidor

por qué las pruebas de extremo a extremo no escalan entre organizaciones · Pact: expectativas del consumidor y verificación del proveedor · broker de contratos y matriz de compatibilidad · puede desplegar: la pregunta que responde la matriz · pruebas de contrato frente a pruebas de esquema · contratos de eventos asíncronos

LABORATORIO: Escribir los contratos de Pact de la app y del socio, verificarlos en la tubería del proveedor, y demostrar que un cambio que rompe a un consumidor queda bloqueado antes de desplegarse

19 Gobierno del catálogo y experiencia del desarrollador

guía de estilo de la organización convertida en reglas · revisión de diseño liviana y quién la hace · catálogo de APIs con dueño, estado y documentación · portal para desarrolladores: lo mínimo que sirve · medir la experiencia: tiempo hasta la primera llamada exitosa · gobierno sin comité que frene todo

LABORATORIO: Convertir la guía de estilo del caso en un conjunto de reglas de Spectral aplicado a todas las APIs del catálogo, publicar el portal con su documentación y medir el tiempo hasta la primera llamada exitosa de un desarrollador nuevo

MÓDULO 8 **PROYECTO INTEGRADOR** 3 h · 1 sesión

20 **Defensa de la integración**

los tres sistemas integrados por contrato · seguridad probada con ataques · deprecación ejecutada en vivo · pruebas de contrato en la tubería · decisiones de estilo y de versionado con sus alternativas · preguntas cruzadas del grupo

LABORATORIO: Defender ante el grupo la integración del ERP, el SaaS y la app con una deprecación ejecutada en vivo sin romper al consumidor antiguo, y con las pruebas de contrato y de seguridad corriendo en la tubería

LO QUE VAS A USAR

Especificaciones abiertas y herramientas libres, elegidas por la decisión que resuelven.

Contratos

OpenAPI 3.2 · AsyncAPI 3 · Problem Details (RFC 9457) · Deprecation (RFC 9745) y Sunset (RFC 8594)

Diseño y gobierno

Spectral · oasdiff · Microcks

Seguridad

Keycloak · OAuth 2 y OpenID Connect · mTLS · Schemathesis

Gateway

Apache APISIX

Integración

Apache Camel · Standard Webhooks · WireMock

Pruebas de contrato

Pact

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 caso con tres sistemas y cada decisión probada contra un consumidor real.

01 **Clases en vivo por Zoom** 20 sesiones de 3 horas, 2 veces por semana. Se diseña en vivo y se revisan los contratos 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 **35% concepto / 65% laboratorio** Se explica la decisión y sus alternativas, y enseguida se aplica a la integración del caso con su criterio de aceptación.

04 **Siempre hay un consumidor antiguo** Los laboratorios de evolución incluyen un cliente viejo que tiene que seguir funcionando. Si se rompe, el laboratorio no está aprobado.

05 **La seguridad se ataca** Cada control se prueba ejecutando contra la propia API el ataque que debe impedir.

06 **Estándares con su estado** Se enseña cada especificación en su versión vigente, y se marca como borrador lo que todavía lo es en el IETF.

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 levanta los tres sistemas del caso y empieza por los consumidores que ya existen.

HTTP y APIs

- ✓ Métodos, códigos y cabeceras
- ✓ JSON
- ✓ Haber consumido una API de terceros

Programación

- ✓ Un lenguaje backend con soltura
- ✓ SQL básico
- ✓ Git y pull requests

Seguridad

- ✓ Nociones de OAuth a nivel de usuario
- ✓ Qué es un certificado TLS

Herramientas

- ✓ Docker en tu máquina
- ✓ Terminal
- ✓ Leer YAML

Equipo y tiempo

- ✓ Computadora con 8 GB de memoria libres para contenedores
- ✓ 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 20 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 y software de código abierto: gateway, servidor de identidad, simuladores e integrador
- Si tu equipo no tiene 8 GB de memoria libres para los contenedores, una máquina virtual alquilada solo durante las prácticas cuesta del orden de US$15 en todo el programa
- No se usa ningún gateway ni servicio de pago

PRECIO

60 horas para que tus contratos sobrevivan a quien los consume.

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 caso con el ERP, el SaaS y la app simulados, listo para levantar
- ✓ Reglas de estilo de Spectral y plantillas de contrato reutilizables
- ✓ Banco de ataques del OWASP API Security Top 10 contra la API del caso
- ✓ 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 20)
- — Exámenes ni certificaciones de terceros: este programa no prepara para ninguno
- — Plataformas de integración comerciales: 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?

Haber construido y consumido APIs HTTP, manejar un lenguaje backend y tener nociones de OAuth como usuario. Si nunca publicaste una API, conviene empezar por ahí antes de este programa.

¿En qué lenguaje se programa?

Los ejemplos del caso vienen en un lenguaje backend común y los laboratorios se pueden hacer en el tuyo: el contrato, el gateway, el integrador y las pruebas de contrato son independientes del lenguaje.

¿Necesito una cuenta de nube?

No. Todo el laboratorio corre en tu máquina con contenedores y software libre: gateway, servidor de identidad, integrador y simuladores. No se usa ningún servicio de pago.

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

Sí, y conviene que sepas qué se repite. Cerca de un tercio de las sesiones toca temas que esos programas tratan dentro de una solución o de una plataforma: escribir contratos OpenAPI y AsyncAPI, OAuth, cuotas por consumidor, compatibilidad y deprecación. Aquí el contrato es el producto, con consumidores externos que no controlas, y los laboratorios son otros. Lo nuevo serían seguridad atacada, gateway, legado por archivos y SOAP, webhooks, patrones de integración y pruebas de contrato. No hay descuento ni convalidación entre programas.

¿Se enseñan GraphQL y gRPC?

Como alternativas en una decisión, no como temario propio. Se evalúan contra REST y mensajería asíncrona para casos concretos, y se prototipa la alternativa descartada para medir la diferencia.

¿Por qué Apache APISIX y no otro gateway?

Porque es de código abierto con licencia Apache, se mantiene activamente y cubre lo que el programa necesita: consumidores, planes, límites, caché y autenticación. Lo que se aprende se traslada a cualquier gateway, incluidos los gestionados de las nubes.

¿Qué pasa con las especificaciones que todavía son borradores?

Se enseñan como tales. Las cabeceras RateLimit, la clave de idempotencia y OAuth 2.1 siguen siendo borradores del IETF: se explica qué proponen, qué se puede usar hoy y qué podría cambiar.

¿Qué pasa si falto a una clase?

La grabación queda disponible y cada laboratorio tiene su punto de partida en el repositorio. La integración se construye por partes: faltar a varias sesiones seguidas sin recuperarlas te deja sin piezas del 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?

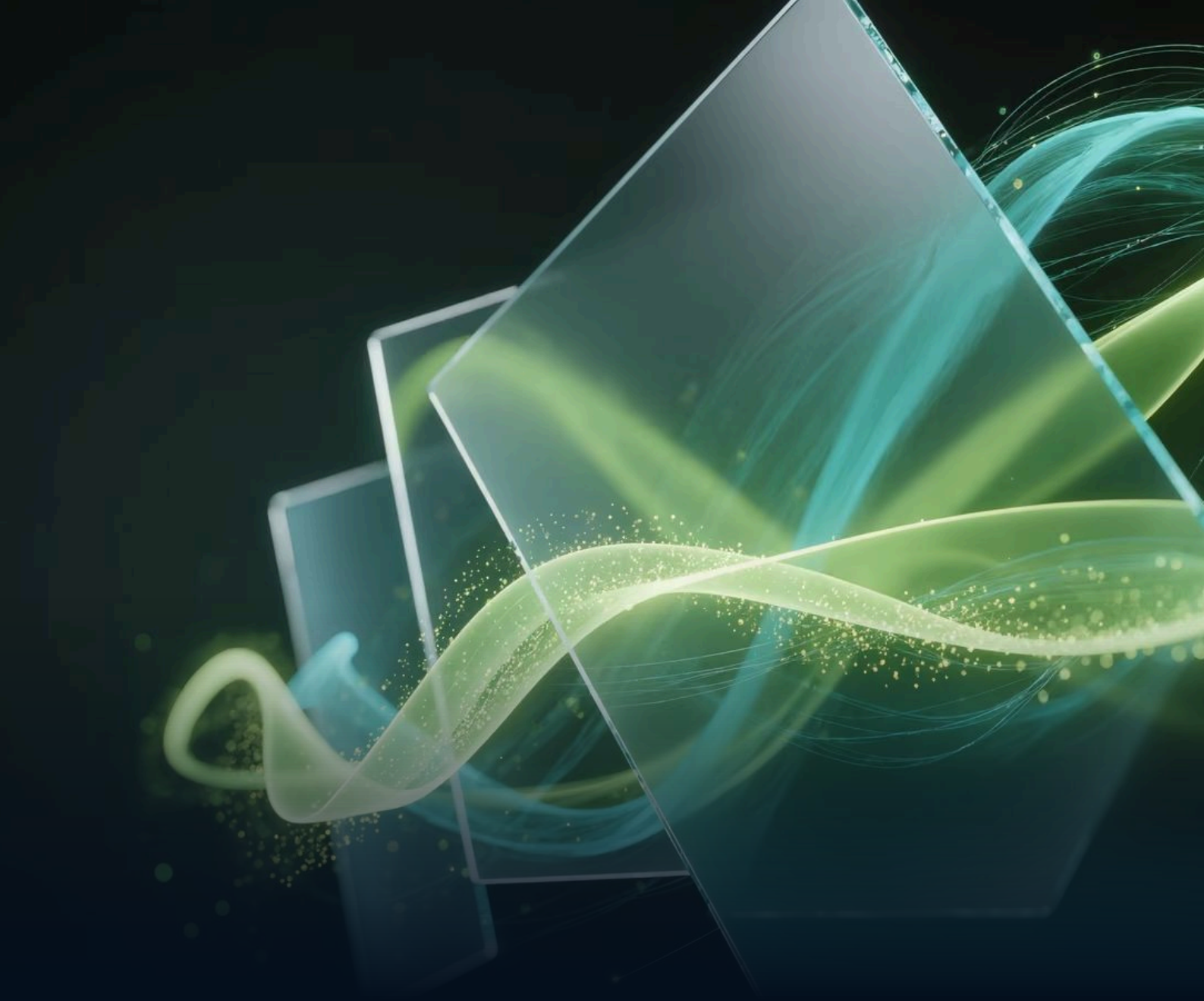
El pago lo procesa Paddle y acepta Tarjeta de crédito, Tarjeta de débito internacional, PayPal, Apple Pay, Google Pay. El cobro es en dólares. Importante: no procesamos Yape, Plin, OXXO, Mercado Pago ni transferencias locales. Si tu banco rechaza compras internacionales, habilítalas antes de pagar o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

`reactorcourses.com/programas/api-integration-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.