PLATFORM DEVELOPER · CHEAT SHEET · CERTIFYDECK

Salesforce Certified Platform Developer

Dominios, pesos oficiales y habilidades clave del temario de Platform Developer, para repasar antes del examen.

Nivel: Intermediate (Associate) Duración: 105 min Fuente: guía de estudio oficial de Microsoft Learn

1. Developer Fundamentals

27–27%

Developer Fundamentals

- Apex se guarda, compila y ejecuta en una plataforma multiinquilino basada en metadatos, por lo que el motor aplica límites del regulador compartidos por todo el código de un contexto de ejecución. Visualforce sigue el patrón MVC: la página es la vista, el controlador estándar o de Apex aporta la lógica y la base de datos es el modelo. Para interfaces nuevas se prefiere Lightning Web Components, que renderiza en el cliente y convive con Aura.
- Antes de programar conviene usar herramientas declarativas: campos de fórmula para calcular valores de un registro o de su padre, campos de resumen acumulativo para contar, sumar o hallar el mínimo o el máximo de los registros detalle de una relación principal-detalle, reglas de validación para rechazar datos no válidos y Flow Builder para automatizar. Los triggers de Apex quedan para lógica compleja o intensiva en CPU y deben procesar en bloque.
- Usa una relación de búsqueda cuando los objetos solo a veces están relacionados y una principal-detalle cuando el detalle depende del principal (borrado en cascada y acceso controlado; el campo va en el detalle); un objeto de unión con dos principal-detalle crea una relación de muchos a muchos. En Apex y SOQL, los nombres personalizados acaban en __c y las relaciones en __r; upsert busca por Id o por un campo External ID.
- Agentforce Vibes extension (antes Agentforce for Developers) trabaja en proyectos de Salesforce DX con la telemetría habilitada: su chat genera y explica Apex, LWC y pruebas, y con el servidor Salesforce DX MCP usa el contexto real de la organización; además autocompleta código en archivos de Apex y LWC. Las reglas siempre se aplican y las skills solo cuando encajan, y la salida no es determinista: revísala siempre.

2. Process Automation and Logic

28–28%

Process Automation and Logic

- Flow Builder es la herramienta para automatizar sin código: los flujos de pantalla piden datos al usuario, los lanzados automáticamente se llaman desde otro flujo o Apex, y los activados por registro, programación o evento de plataforma van en segundo plano. En los activados por registro, las actualizaciones rápidas cambian el registro antes de guardarlo; acciones y registros relacionados, las rutas asíncronas y las programadas actúan después.
- En Apex toda variable empieza en null y el lenguaje no distingue mayúsculas de minúsculas. Los métodos static se llaman con el nombre de la clase; los private solo desde su clase, y global abre el acceso entre espacios de nombres. Una interfaz declara firmas sin cuerpo que cada clase implementa (Schedulable exige execute()); las variables static solo duran un contexto de ejecución, y Database.Stateful conserva las de instancia entre lotes.
- En Apex, if, else if ejecuta solo la primera rama verdadera y cada else se une al if más cercano; switch on ejecuta el bloque when que coincide. while comprueba la condición antes de cada vuelta, do-while ejecuta el bloque al menos una vez y for repite un número fijo de veces o recorre una colección. Un bucle for de SOQL con lista itera por lotes de 200, y las DML o SOQL dentro de un bucle agotan los límites del regulador.
- SOQL lee registros de un solo objeto con coincidencia exacta, admite subconsultas por nombre de relación (Contacts, __r) y notación de punto hacia el principal; SOSL busca palabras en varios objetos y devuelve una lista de listas en el orden de RETURNING. Las instrucciones DML son de todo o nada; Database.insert(lista, false) permite éxito parcial con SaveResult, y upsert usa el Id, un campo idLookup o un ID externo.
- Usa un único trigger por objeto que delegue en métodos de una clase de controlador y diséñalo siempre para colecciones: recorre Trigger.new y haz SOQL y DML fuera de los bucles. Los triggers before cambian campos sin DML; los after leen valores del sistema como el Id y tratan sus registros como de solo lectura. Elige bien las variables de contexto (oldMap en update y delete) y usa addError() para bloquear el guardado.

3. User Interface

25–25%

User Interface

- En Apex, with sharing aplica las reglas de colaboración del usuario y el modo de usuario (WITH USER_MODE, as user, AccessLevel.USER_MODE) respeta sus permisos de objeto y de campo. En el cliente, deja que LWC y Lightning Web Security gestionen el DOM, evita lwc:dom="manual" y destinos como parseFromString, prefiere listas de permitidos y codificación de salida, y carga scripts de terceros solo desde recursos estáticos.
- Un componente web Lightning aparece en Lightning App Builder con isExposed a true y al menos un destino; targetConfigs limita objetos y @api recordId recibe el Id en páginas de registro. Las páginas de Visualforce se reutilizan en Lightning Experience (App Builder, acciones rápidas, sustitución de botones) y se comunican con LWC mediante Lightning Message Service; los flujos de pantalla cubren formularios con lógica sin código.
- Lightning Web Components se basa en estándares web que el navegador ejecuta de forma nativa: un componente necesita como mínimo un HTML y un JavaScript con el mismo nombre, y su CSS queda encapsulado por Shadow DOM. La información baja del padre al hijo con propiedades y métodos @api y sube con CustomEvent y dispatchEvent; entre hermanos se usa el patrón eventos arriba, propiedades abajo.
- Un método Apex llamado desde un componente Lightning debe ser static, public o global y llevar @AuraEnabled; con cacheable=true puede conectarse con @wire y refrescarse con refreshApex, pero no admite DML, que se hace con llamadas imperativas. Para Flow y Agentforce se usa @InvocableMethod con clases de entrada y salida marcadas con @InvocableVariable, cuyas descripciones guían al agente.

4. Testing, Debugging, and Deployment

20–20%

Testing, Debugging, and Deployment

- Las clases de prueba llevan @IsTest, no ven los datos de la organización salvo objetos como User y Profile, y sus registros se revierten al terminar. Los datos se crean a mano, con fábricas, métodos @TestSetup o CSV cargados con Test.loadData(); Test.startTest() y Test.stopTest() dan límites nuevos y ejecutan el código asíncrono. Los flujos se prueban con Debug y con View Tests.
- Salesforce DX lleva la fuente de verdad al control de versiones: el Dev Hub crea scratch orgs a partir de config/project-scratch-def.json y Salesforce CLI (sf) autoriza organizaciones, implementa y recupera código, mueve datos de prueba y asigna permisos. Las extensiones de VS Code ejecutan esos comandos desde la paleta, y Developer Console sirve para Apex anónimo, consultas y conjuntos de pruebas en el navegador.
- Los registros de depuración son la fuente principal: sus niveles (de NONE a FINEST) son acumulativos, y los checkpoints de Developer Console capturan datos sin detener la ejecución. El Apex asíncrono se vigila en Apex Jobs, la flex queue y con consultas a AsyncApexJob o CronTrigger; los flujos se depuran con Debug y sus acciones programadas se ven en Time-Based Automations.
- El Apex se escribe en un sandbox u organización de desarrollo y se despliega a producción con al menos un 75 % de cobertura, todas las pruebas superadas y algo de cobertura en cada trigger. Cada entorno tiene su papel: Developer para desarrollar, Partial Copy y Full para pruebas y preproducción, y scratch orgs desechables. Un package.xml lista los componentes; se valida, se hace quick deploy y se aplica la lista de ejecución.