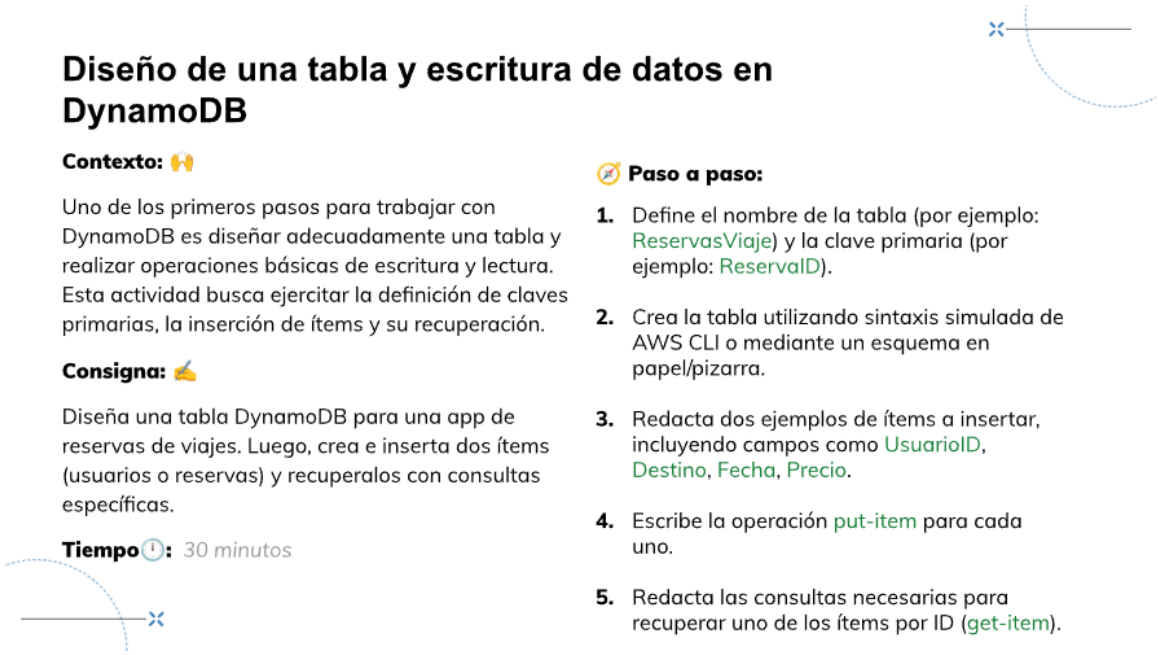

Ejercicio – Diseño de una tabla y escritura de datos en DynamoDB

Objetivo

Diseñar una tabla DynamoDB para una app de reservas de viajes. Insertar ítems representativos y recuperarlos usando consultas por clave primaria.

1. Definición de la tabla

- Nombre de la tabla: ReservasViaje
- Clave primaria: Partition Key (PK): ReservaID

2. Estructura de ítems

Ejemplo de ítem 1 (JSON):

```
{
 "ReservaID": "R001",
 "UsuarioID": "U100",
 "Destino": "Cusco",
 "Fecha": "2025-09-15",
 "Precio": 320
}
```

Ejemplo de ítem 2 (JSON):

```
{
  "ReservaID": "R002",
  "UsuarioID": "U101",
  "Destino": "Isla de Pascua",
  "Fecha": "2025-11-02",
  "Precio": 560
}
```

3. Comando para insertar ítems (put-item)

Comando para insertar ítem 1:

```
aws dynamodb put-item \
  --table-name ReservasViaje \
  --item '{
    "ReservaID": {"S": "R001"},
    "UsuarioID": {"S": "U100"},
    "Destino": {"S": "Cusco"},
    "Fecha": {"S": "2025-09-15"},
    "Precio": {"N": "320"}
  }'
```

Comando para insertar ítem 2:

```
aws dynamodb put-item \
  --table-name ReservasViaje \
  --item '{
    "ReservaID": {"S": "R002"},
    "UsuarioID": {"S": "U101"},
    "Destino": {"S": "Isla de Pascua"},
    "Fecha": {"S": "2025-11-02"},
    "Precio": {"N": "560"}
  }'
```

4. Consulta para obtener ítem por ID (get-item)

```
aws dynamodb get-item \
  --table-name ReservasViaje \
  --key '{"ReservaID": {"S": "R001"}}'
```

Preguntas Clase:

1. ¿En qué casos conviene usar DynamoDB por sobre una base relacional?

DynamoDB es recomendable cuando:

- Se requiere **alta escalabilidad horizontal** y rendimiento constante con grandes volúmenes de datos.
- Se busca **baja latencia en milisegundos**, especialmente en aplicaciones en tiempo real.
- No se necesita realizar **joins complejos** ni transacciones múltiples entre tablas.
- El modelo de datos es **flexible o semiestructurado**, como en aplicaciones móviles, IoT, e-commerce, chat o juegos online.
- Se desean aprovechar características serverless y **gestión automática de infraestructura**.

2. ¿Cómo afecta el diseño de la clave primaria al rendimiento?

- El diseño de la clave primaria (Partition Key y Sort Key) es **clave para el rendimiento** en DynamoDB.
- Una **mala distribución** de claves (por ejemplo, muchas entradas con la misma clave) puede causar "hot partitions", afectando el rendimiento.
- Se recomienda elegir una Partition Key que garantice **alta cardinalidad y distribución uniforme** de datos.
- La Sort Key permite **ordenar y consultar eficientemente** dentro de una partición.

3. ¿Qué ventaja ofrece la integración de DynamoDB con AWS Lambda?

- Permite construir **aplicaciones serverless completamente desacopladas**.
- Se pueden ejecutar funciones automáticamente en respuesta a cambios en la base de datos (inserciones, actualizaciones, eliminaciones) usando **streams de DynamoDB**.
- Facilita la **automatización de flujos de trabajo**, validaciones, envío de notificaciones o sincronización con otros sistemas.
- Reduce costos y complejidad de infraestructura al no necesitar servidores para manejar lógica backend.