Challenge Backend SSR - Serverless Task API

Tiempo estimado: 3-4 horas
Nivel: Semi-Senior
Stack: Node.js + Arquitectura Serverless

Contexto

Necesitas construir una API serverless para gestión de tareas personales. La API debe ser stateless, eficiente y seguir buenas prácticas de arquitectura serverless/cloud-native.

Stack Técnico Requerido

- **Node.js** (v18+)
- **Arquitectura serverless** (AWS Lambda, Serverless Framework, o simulación local)
- **JWT** para autenticación
- **Almacenamiento** a tu elección (DynamoDB, SQLite, en memoria, etc.)

Nota: Puedes trabajar completamente en local sin necesidad de desplegar a AWS.

Requisitos Funcionales

1. Autenticación

Endpoint: `POST /auth/login`

Implementar login con credenciales estáticas:

- Usuario: `admin`
- Password: `password123`

Debe retornar un JWT válido que incluya información del usuario. Este token se usará para proteger el resto de los endpoints.

Todas las rutas posteriores deben estar protegidas y validar el JWT. Retornar `401 Unauthorized` si el token es inválido o está ausente.

---

2. CRUD de Tareas

Cada tarea debe contener al mínimo:

- `id` (único)
- `title`
- `description`
- `completed` (boolean)
- `userId` (del usuario autenticado)
- `createdAt` (timestamp)

Endpoints requeridos:

`GET /tasks`

- Retorna todas las tareas del usuario autenticado

`POST /tasks`

- Crea una nueva tarea
- Body: `{ "title": "...", "description": "..." }`
- El campo `completed` debe inicializarse en `false`

`PATCH /tasks/:id`

- Actualiza una tarea existente (title, description, o completed)
- Debe validar que la tarea pertenezca al usuario autenticado

`DELETE /tasks/:id`

- Elimina una tarea
- Debe validar ownership

---

3. Integración con API Externa

`POST /tasks/import`

Debe realizar lo siguiente:

1. Llamar a la API: `https://jsonplaceholder.typicode.com/todos`
2. Filtrar y tomar las primeras **5 tareas** que correspondan al usuario autenticado (puedes mapear `admin` → `userId: 1` de la API externa)

3. Guardarlas en tu base de datos como tareas del usuario actual
4. Retornar cuántas tareas se importaron y el listado

Requisitos Técnicos

Arquitectura Serverless

- La solución debe seguir principios serverless (stateless, event-driven)
- Considera optimizaciones típicas de este tipo de arquitecturas (cold starts, conexiones, etc.)

Manejo de Errores

- Responses HTTP consistentes y apropiados
- Manejo adecuado de errores y edge cases
- Validación de inputs

Código

- Código limpio, organizado y mantenible
- Separación de responsabilidades
- Evitar duplicación de lógica

Entregables

Repositorio Git que incluya:

1. **Código fuente completo**
2. **README.md** con:
    - Instrucciones claras de instalación
    - Cómo ejecutar el proyecto localmente
    - Ejemplos de uso de los endpoints (curl, Postman, o similar)
    - Decisiones técnicas importantes que tomaste y por qué
3. **Archivo de configuración** necesario para levantar el proyecto (serverless.yml, SAM template, o lo que hayas usado)
4. **Collection de Postman/Insomnia** (opcional pero recomendado)

Criterios de Evaluación

Evaluaremos tu solución considerando:

- **Arquitectura y diseño** (30%) - Estructura del proyecto, separación de concerns, patrones aplicados
- **Código limpio** (25%) - Legibilidad, mantenibilidad, principios SOLID
- **Funcionalidad** (25%) - Todos los endpoints funcionan correctamente según especificaciones
- **Manejo de errores** (20%) - Robustez, validaciones, responses consistentes

---

Extras Opcionales

Si te sobra tiempo y quieres destacar, puedes agregar:

- Tests (unitarios o de integración)
- Paginación en `GET /tasks`
- Filtros avanzados (por completed, fecha, etc.)
- Documentación de API (Swagger/OpenAPI)
- Logging estructurado
- Deploy funcional en AWS (capa gratuita)

Nota: Los extras son opcionales y NO son requisitos. Prioriza una solución sólida de los requisitos principales.

Consideraciones Finales

- **Tiempo:** No deberías necesitar más de 3-4 horas. Si te estás excediendo, simplifica.
- **Preguntas:** Si algo no está claro en los requerimientos, documenta tus asunciones en el README.
- **Realismo:** Piensa en esto como un proyecto real - decisiones pragmáticas sobre arquitectura perfecta.

¡Éxito con el challenge! 🚀