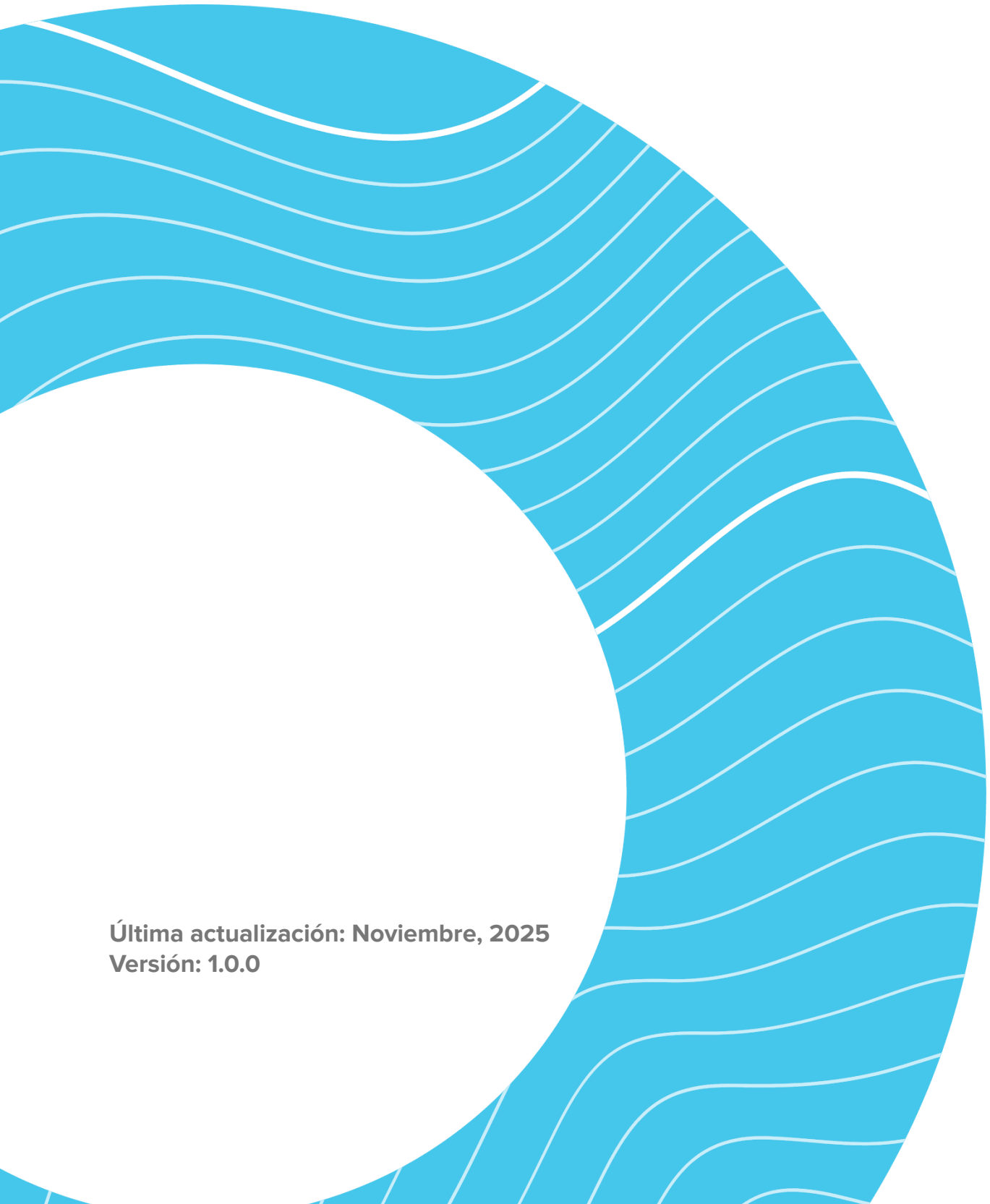
ONEBITTECH

Examen Ingreso Backend

Onebittech

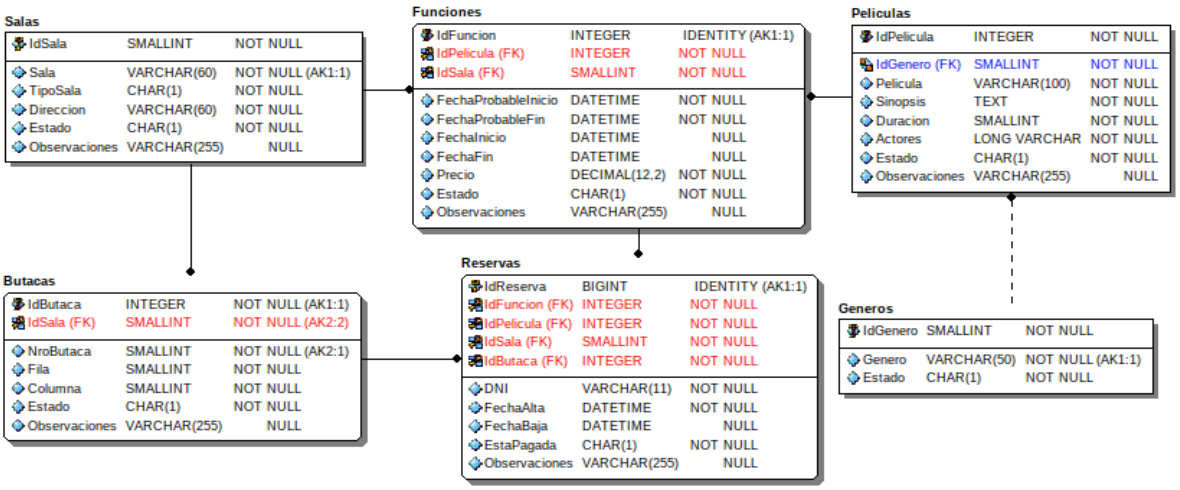


Última actualización: Noviembre, 2025
Versión: 1.0.0

ONEBITTECH

Ejercicio

A continuación se detalla el ejercicio que debe realizar el candidato:
El candidato tendrá que crear una base de datos para una sala de cine y una API que permita gestionar películas, horarios, disponibilidad de entradas y reservas.



1. Modelo de Base de Datos:

El modelo lógico se encuentra en la imagen provista. Se deberán crear las tablas siguiendo exactamente las relaciones del diagrama, **sin eliminar ni agregar columnas**.
Se deberán cumplir además las siguientes reglas:

Reglas obligatorias de integridad

1. El campo `Precio` de la tabla **Funciones** debe ser **mayor a 0**.
2. El `Estado` en todas las tablas es **'A'** (Activo) o **'I'** Inactivo).
3. En **Funciones**, las fechas deben cumplir:
    - `FechaInicio < FechaFin`
    - `FechaProbableInicio <= FechaProbableFin`
4. No pueden existir **salas, películas, géneros ni butacas** con nombres repetidos.
5. Crear índices en:
    - `Funciones(FechaInicio)`
    - `Butacas(IdSala, NroButaca)`
    - `Reservas(IdFuncion, IdButaca)`
    - `Otros índices que considere necesario`

2. Implementar en la BD:

2.1 – SP: Determinar Precio de Entrada

Entrada: pIdFuncion INT
Salida: Precio calculado DECIMAL(12,2) o detalle de error.
Reglas:

1. Si la función es para un género específico (ej. 'Estreno' o '3D', que deberías insertar previamente), aplicar un recargo del 10% sobre el Precio base de la función.
2. Si la función tiene un IdSala específica (ej. IdSala = 1, la sala VIP), aplicar un recargo adicional del 5%.
3. El precio final no puede ser menor al Precio base de la función.
4. Solo funciones activas (Estado='A') que no hayan finalizado.

2.2 – SP: Reporte de Ocupación por Película

Entrada: pIdPelicula INT, pFechaInicio DATE, pFechaFin DATE
Salida: Listado con:
IdFuncion
FechaInicio (Hora de inicio)
IdSala, Nombre de la Sala
Total Butacas Vendidas (contando reservas)
Total Ingresos Recaudados (Butacas Vendidas * Precio de la función)
Reglas:
Considerar solo funciones con Estado='A' cuya FechaInicio esté dentro del rango pFechaInicio y pFechaFin.

2.3 – SP: Reservar Butaca con Validación de DNI Único

Entrada: pIdFuncion INT, pIdButaca INT, pDNI VARCHAR(11)
Salida: "OK" o detalle de error.
Reglas Adicionales a las Originales:

1. Regla Nueva: Una persona identificada por pDNI no puede tener más de 4 reservas activas (sin fecha de baja y pagadas) en total para la misma FechaInicio (solo la fecha, ignorando la hora) en todas las funciones. Si supera este límite, devolver el error.
2. (Se mantienen las reglas originales 1, 2, 3, 4 y 5 del examen original).

3. API:

Desarrollar una API en go o en el lenguaje que domine, que permita:

Endpoints:

- GET /precios/{idFuncion}: Utiliza el SP 2.1 para obtener el precio calculado de una función.
- GET /reporte/ocupacion: Utiliza el SP 2.2 para generar el reporte (los parámetros de fecha e idPelicula se pasan como query parameters o en el cuerpo de la solicitud POST, si lo prefieres).
- POST /reservas: Utiliza el SP 2.3 para realizar la reserva.
- Endpoint adicional a su elección.

Requisitos:

- Documentar con **Swagger/OpenAPI**.
- Manejo de errores consistente (HTTP 400/404/409/500).
- Paginación opcional

4. Entrega

Publicar código en un repositorio público (GitHub/GitLab).

Incluir un README con:

- Pasos para crear BD
- Pasos para ejecutar SPs
- Pasos para correr la API
- Ejemplos de llamadas con curl/postman

5. Defensa

- ☐ Se coordinará una reunión para la defensa del proyecto, en la cual se explicará la forma de diseñar y desarrollar el proyecto, y posterior a esto se realizarán diversas preguntas por parte de los miembros del equipo.