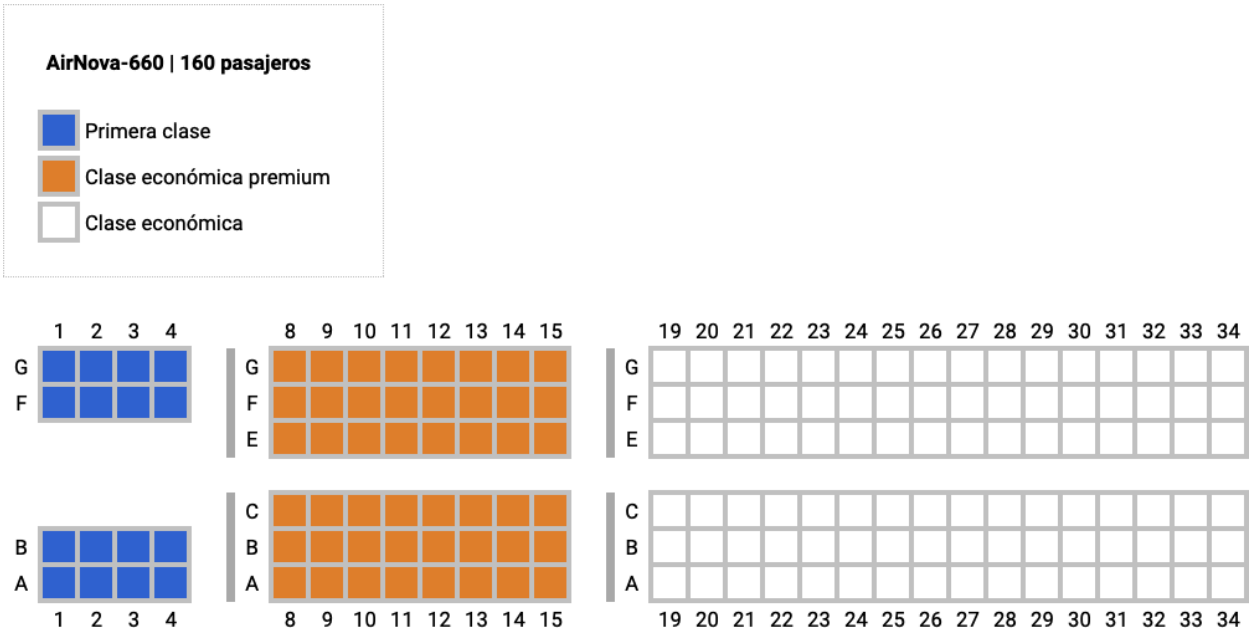
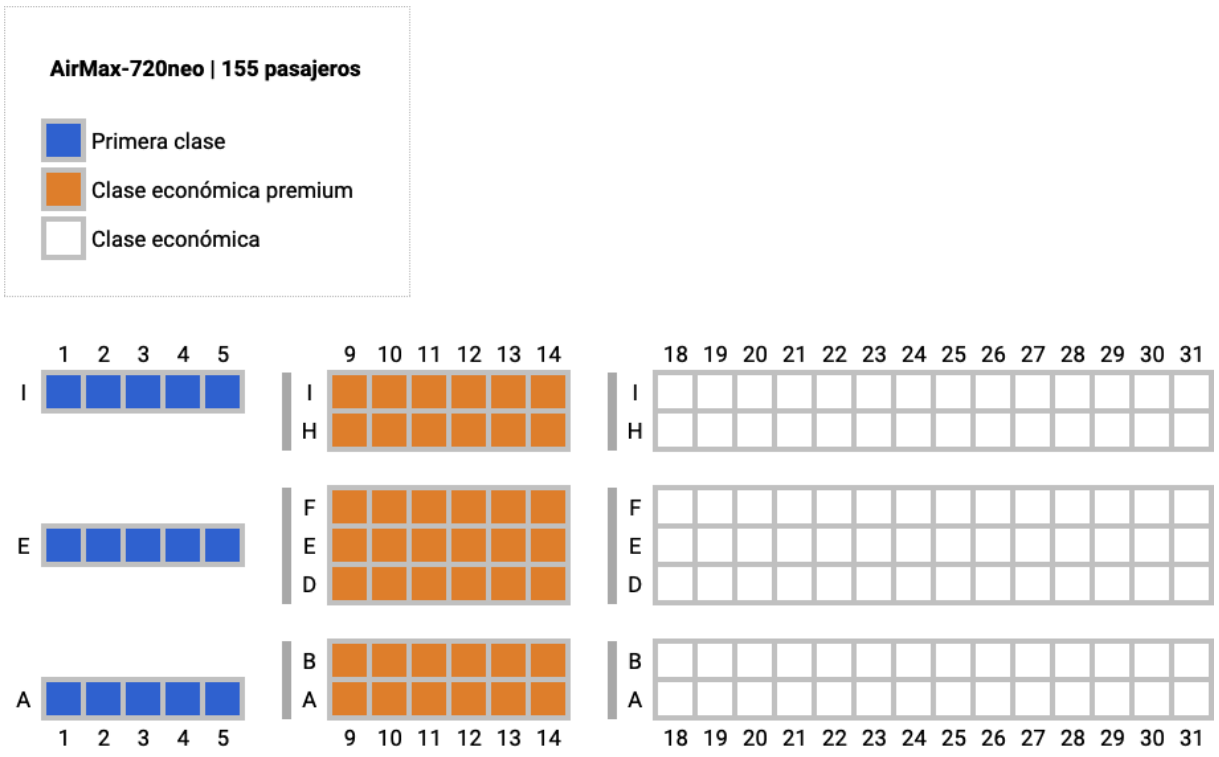
Ejercicio de postulación: Simulación check-in Aerolínea

Una aerolinea está evaluando comenzar sus operaciones en países de latinoamérica. Para esto, necesita saber si es posible realizar un check-in automático de sus pasajeros. Tal como se puede ver en las siguientes figuras la aerolínea cuenta con 2 aviones, de distinta capacidad y distribución de asientos:



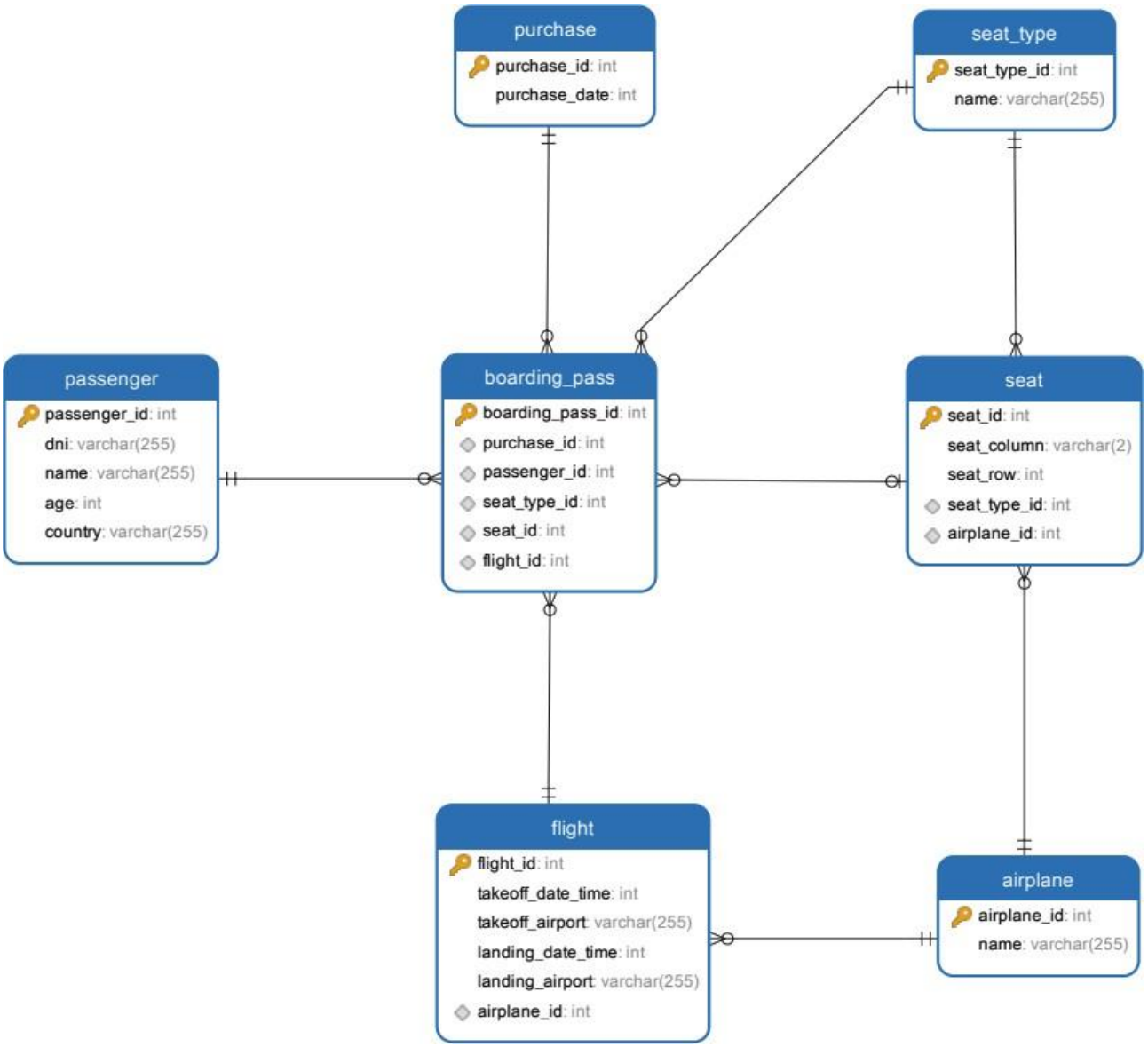


Desafío

Como futuro miembro del equipo de desarrollo, tu misión será crear un programa que realice esta simulación de check-in. Para ello contarás con una base de datos (solo lectura) que contiene todos los datos necesarios para la simulación.
Las credenciales de conexión son:

- `Motor: MySQL`
- `Host: ***`
- `Usuario: ***`
- `Contraseña: ***`
- `Nombre db: airline`

Tip para la conexión: El servidor está configurado para que todas aquellas conexiones inactivas por más de 5 segundos sean abortadas, por lo que se requiere controlar la reconexión.



Con esto, deberás crear una API REST, lenguaje y/o framework a libre elección, con un solo endpoint que permita consultar por el ID del vuelo y retornar la simulación. Tal como muestra el ERD, una compra puede tener muchas tarjetas de embarque asociadas, pero estas tarjetas pueden no tener un asiento asociado, siempre tendrá un tipo de asiento, por lo tanto, al retornar la simulación del check-in se debe asignar el asiento a cada tarjeta de embarque.

Los puntos a tomar en cuenta son:

- Todo pasajero menor de edad debe quedar al lado de al menos uno de sus acompañantes mayores de edad (se puede agrupar por el id de la compra).
- Si una compra tiene, por ejemplo, 4 tarjetas de embarque, tratar en lo posible que los asientos que se asignen estén juntos, o queden muy cercanos (ya sea en la fila o en la columna).
- Si una tarjeta de embarque pertenece a la clase “económica”, no se puede asignar un asiento de otra clase.

El servicio debe tener la siguiente estructura:

`Método: GET`

`Ruta: /flights/:id/passengers`

`Respuesta exitosa:`


```
{
 "code": 200,
 "data": {
   "flightId": 1,
   "takeoffDateTime": 1688207580,
   "takeoffAirport": "Aeropuerto Internacional Arturo Merino Benitez, Chile",
   "landingDateTime": 1688221980,
   "landingAirport": "Aeropuerto Internacional Jorge Cháve, Perú",
   "airplaneId": 1,
   "passengers": [
     {
       "passengerId": 90,
       "dni": 983834822,
       "name": "Marisol",
       "age":  44,
       "country": "México",
       "boardingPassId": 24,
       "purchaseId": 47,
       "seatTypeId": 1,
       "seatId": 1
     },
     {...}
   ]
 }
}
```

Vuelo no encontrado:

```
{
 "code": 404,
 "data": {}
}
```

En caso de error:

```
{
 "code": 400,
 "errors": "could not connect to db"
}
```

Muy importante

- Los campos en la bd están llamados en Snake case, pero en la respuesta de la API deben ser transformados a Camel case.
- Se debe entregar la misma estructura de respuestas pues la API será testeada de forma automática, y si no cumple con la estructura el ejercicio quedará descartado.
- Contemplar buenas prácticas de programación, una manera de medir esto es pensando que estás desarrollando una aplicación que será utilizada en productivo.
- Utilizar control de versiones.
- Este ejercicio **no tiene ningún fin comercial ni estratégico**, sólo queremos medir tus habilidades y conocimientos.

Entregables

- Archivo README.md que indique como ejecutar la API. Agregar documentación del proyecto que consideres relevante.
- Código fuente en repositorio **privado** de GitHub, y acceso al usuario ***
- Enviar mediante este formulario la siguiente información:
  - ☐ CV en formato PDF.
  - ☐ URL donde estará desplegada la API (AWS, Heroku u otro hosting).
  - ☐ URL del repositorio de GitHub.

¡Mucho éxito!

😎