Trabajo de regularización 2025

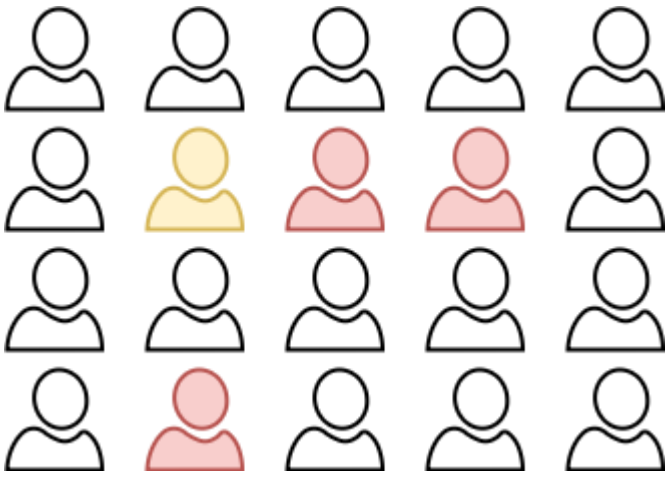
1. Introducción

Este trabajo práctico se desarrollará a lo largo del año y será utilizado para **regularizar la materia**. La nota final obtenida será la que se registre en actas durante las **mesas de examen**.
Fecha límite de entrega: el día del **segundo llamado de las mesas de diciembre**.

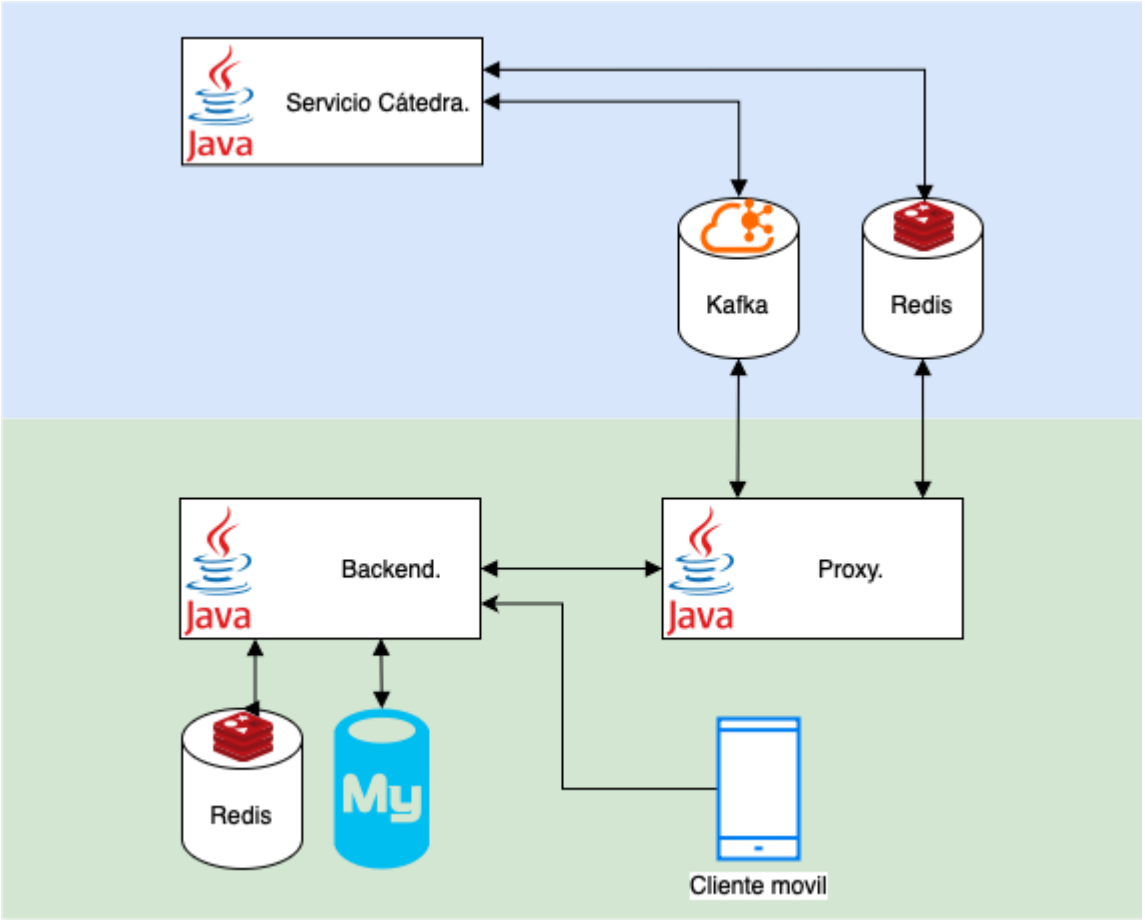
El desarrollo se realizará de forma **individual**.

2. Objetivo General

El objetivo es construir un sistema que permita **registrar la asistencia a eventos únicos**, tales como charlas, cursos, obras de teatro, etc.
Cada evento contará con un título, descripción, fecha y hora, organizador (quien organiza), presentadores, cantidad de asientos totales, y una **distribución de asientos en filas y columnas**.



3. Arquitectura del Sistema



El sistema está dividido en **cuatro componentes**:

3.1. Servicio de la Cátedra (cátedra)

Desarrollado por la cátedra. Expone una serie de endpoints y servicios que el alumno deberá consumir para realizar diferentes actividades como ser: listados, ventas, bloqueos, etc.

3.2. Servicios adicionales de la cátedra (servicios)

Se agregan 2 servicios adicionales. Estos servicios se encuentran asociados al servicio de la cátedra pero serán consumidos por el servicio proxy del alumno.

3.2.1 Kafka

Este servicio se utiliza para notificar cambios en la información de los eventos, cuando en el servicio de la cátedra se produce un cambio en un evento será registrado en Kafka y el alumno deberá leer información desde allí o puede ser utilizado como notificación de cambios.

3.2.2 Redis

Este servicio servirá para mantener la información actualizada de las ubicaciones de los asientos de cada evento. El backend deberá consultar el servicio de redis para mantenerse actualizado respecto de los asientos.

3.3. Backend del Alumno (backend)

Desarrollado en Java utilizando springBoot (preferentemente JHipster). Es el backend que interactúa con el cliente móvil y el servicio de la cátedra.

3.4. Servicio Proxy del Alumno (proxy)

Servicio adicional, también desarrollado por el alumno. Es el **único que tiene acceso a Kafka y Redis de la cátedra**. Sirve como intermediario entre el backend y estos servicios.

3.5. Cliente móvil (móvil)

El cliente móvil será desarrollado por el alumno en **Kotlin Multiplatform (KMP)**.

Este cliente será la interfaz gráfica del backend.

4. Funcionalidad global

4.1 Manejo de eventos

El servicio de la cátedra almacena eventos que serán modificados en cualquier momento, estos cambios pueden ser:

- agregar nuevos eventos
- eventos que expiran por tiempo. Desaparece del listado una vez que expira.
- eventos que se cancelan.
- eventos donde se cambian algunos datos.

El servicio del backend debe mantener sincronizado los eventos que se encuentran en el servicio de la cátedra en su almacenamiento de datos local.

Cada cambio que se produzca de los eventos en el servicio de la cátedra será notificado en Kafka (que es una cola que irá almacenando los cambios en orden cronológico). Se notificará el dato cambiado en el evento.
Cada alumno tendrá su consumer group id para poder leer los datos de forma independiente de los demás alumnos.

Cada cambio que se realice en los datos de los eventos el servicio del alumno deberá actualizar sus datos locales para tener los datos perfectamente sincronizados.

El alumno puede usar el contenido del evento para actualizar sus datos locales o simplemente utilizarlo para enterarse que ha ocurrido un cambio y poder sincronizar los datos llamando al endpoint de información de eventos.

4.2 Proceso de selección de eventos y ubicaciones

La siguiente explicación está relacionada a la actividad del cliente móvil y el backend durante el proceso de selección de eventos.

a- El cliente móvil debe poder obtener una lista de eventos desde el backend, esta lista es resumida y sirve para poder seleccionar algún evento que nos interese.
b- Al seleccionar el evento se muestran todos los detalles del mismo incluyendo la cantidad de asientos disponibles y las ubicaciones.
c- En este momento se genera una sesión que dura X cantidad de minutos que debe mantenerse incluso si se inicia sesión desde otro cliente con el mismo usuario.
d- Inicialmente se elige 1 solo asiento y se puede incrementar hasta 4, cuando se seleccionan los asientos y se decide continuar se deben bloquear los asientos (contra el servicio de la cátedra)
e- A continuación se debe cargar el nombre y apellido de cada uno de los asientos. En este punto se puede volver hacia atrás para agregar/quitar asientos o reasignarles un lugar.

4.3 Venta

Una vez seleccionados los nombres de los asientos se puede realizar la venta confirmando la compra. También se le puede volver hacia atrás para cambiar el nombre de los asientos.

Al realizar la venta se debe registrar localmente en el sistema y notificar al servidor de la cátedra.

4.4 Resumen y sesión

Resumiendo el proceso se siguen los siguientes pasos:
a- Listado de todos los eventos
b- Selección de evento
c- Selección de asientos
d- Carga de nombres de los asientos
e- Venta

Si se inicia sesión en otro cliente móvil, se debe retomar en el punto donde había quedado en la interfaz anterior. Por ejemplo, si en un cliente se había seleccionado los asientos y estaba en la pantalla para cargar los nombres, el cliente nuevo deberá continuar desde ese punto.

4.5 Cierre de sesión / expiración de sesión

Si el cliente cierra sesión en el cliente móvil se deberá invalidar los datos y al iniciar sesión nuevamente deberá comenzar el proceso nuevamente.
Si la sesión expira el cliente al iniciar sesión nuevamente arrancará desde el paso 1.

5. Funcionalidades por componente

5.1. Servicio de la Cátedra (cátedra)

Estas son las funcionalidades del servicio

- **Registro de alumnos:** se debe registrar el alumno para obtener un ID de sesión y un token de JWT para poder consumir el servicio de la cátedra.
- **Gestión de eventos**: creación, modificación y baja. Esto lo manejan los profesores de la cátedra.
- **Bloqueo de asientos**: al recibir la notificación del backend, bloquea temporalmente los asientos (actualizando Redis).
- **Registro de ventas**: al confirmar una compra, el backend notifica la venta y se actualiza el estado en Redis.
- **Notificaciones vía Kafka**: cuando ocurre un cambio en un evento será registrado en el tópico de kafa para que cada alumno tome las medidas necesarias.
- **Exposición de datos de eventos**:
    - Listado completo de eventos activos.
    - Detalles individuales por ID.
- **Exposición de datos de ventas**:
    - Listado resumido de todas las ventas.
    - Detalle individual de cada venta.

Endpoints para los alumnos

- Registro de sesión del alumno. http://SERVIDOR:PUERTO/api/v1/agregar_usuario . Ver payload 1.
- Login de usuario. http://localhost:8080/api/authenticate . Ver payload 2.
- Listado completo de eventos (datos resumidos). http://SERVIDOR:PUERTO/api/endpoints/v1/eventos-resumidos. Ver payload 3.
- Listado completo de eventos (con todos los datos). http://SERVIDOR:PUERTO/api/endpoints/v1/eventos. Ver payload 4.
- Datos completos de un evento. http://SERVIDOR:PUERTO/api/endpoints/v1/evento/{id} . Ver payload 5.
- Bloqueo de asiento por evento. http://SERVIDOR:PUERTO/api/endpoints/v1/bloquear-asientos . Ver payload 6.
- Venta de asientos por evento. http://SERVIDOR:PUERTO/api/endpoints/v1/realizar-venta . Ver payload 7.
- Listado completo de ventas por cada alumno (datos resumidos). http://SERVIDOR:PUERTO/api/endpoints/v1/listar-ventas . Ver payload 8.
- Ver datos de una venta en particular. http://SERVIDOR:PUERTO/api/endpoints/v1/listar-venta/{id}. Ver payload 9.

5.2. Servicio Backend del Alumno (backend)

- **Registro de backend contra el servidor de cátedra**:
    - Este proceso es manual y no lo hace el servicio. Al hacerlo manual se obtiene un token que debe utilizarse para configurar el servicio para acceder a los endpoints del servicio de la cátedra.
- **Registro de usuarios/clientes:**
    - Cada usuario o cliente es alguien que quiere anotarse en un evento.

    - JHipster tiene una interfaz web para la creación de usuarios que es parte de lo que se genera al crear la aplicación. Esta interfaz puede utilizarse para crear nuevos usuarios, por lo que no es necesario realizar el registro de usuarios nuevos desde la aplicación móvil.
- **Inicio de sesión**:
    - El cliente móvil (o utilizando algún cliente web tipo insomnia, postman o cURL) inicia sesión contra el backend.
    - Mantiene sesiones activas utilizando Redis local.
    - Las sesiones expiran a los **30 minutos de inactividad** (valor **parametrizable**).
    - Si se inicia sesión en otro cliente simultáneamente ambos verán lo mismo de la sesión.
- **Manejo de eventos y asientos**:
    - Lista eventos consumiendo los endpoints de la cátedra.
    - Al seleccionar un evento, solicita al `proxy` los datos de asientos en Redis.
    - Al seleccionar asientos y avanzar al paso de cargar datos de los asientos , los bloquea temporalmente (5 minutos) notificando a `cátedra` (a través del proxy).
    - Antes de la venta, verifica que los asientos aún estén disponibles.
    - Al confirmar la venta, notifica a `cátedra` y espera confirmación.
- **Persistencia**:
    - Debe almacenar información local sobre eventos y ventas en base de datos.
    - Si no puede confirmar la venta localmente, esta debe **quedar pendiente** y reintentarse hasta concretarse.
    - Se puede dar un caso de ambigüedad si otra persona compró el mismo asiento, pero este conflicto no será resuelto en esta instancia.
- **Sesiones concurrentes**:
    - El sistema debe permitir múltiples sesiones activas simultáneamente.
    - Redis local debe garantizar persistencia ante reinicios del backend o despliegues con múltiples instancias.
- **Actualizaciones**:
    - El `proxy` recibe notificaciones desde Kafka y debe comunicarse con el backend para informar los cambios.

5.3. Servicio Proxy del Alumno (proxy)

- Accede a **Redis** de la cátedra para:
    - Consultar el estado de los asientos cuando lo solicite el backend.
- Está **suscripto al tópico Kafka**:
    - Recibe notificaciones de cambios en eventos.
    - Debe comunicarse con el backend para informar los cambios.
- Se debe resolver la forma en que el proxy notificará al backend de algún cambio en los eventos.

5.4. Cliente móvil (`móvil`)

- **Iniciar sesión contra el backend:**
  - Se debe ingresar el usuario y contraseña para generar la sesión.
- **Listar eventos:**
  - Se debe mostrar una lista de eventos y dar lugar a seleccionarlos.
  - Los datos que se muestran son aquellos relevantes para poder entender de qué se trata el evento y que no ocupe demasiado lugar en pantalla.
- **Detalles del evento seleccionado:**
  - Visualizar la información detallada de un evento.
  - Visualizar el mapa de asientos.
  - Agregar una opción para volver al paso anterior.
- **Selección de asientos:**
  - Seleccionar hasta **cuatro asientos por sesión**.
  - Visualmente se deben ver los asientos libres, vendidos, seleccionados por mi y bloqueados por otro. Se puede hacer cambiando de color el elemento que representa al asiento.
  - Se puede indicar la cantidad de asientos a reservar (de 1 a 4)
  - Se pueden seleccionar los asientos que deseamos.
  - Agregar una opción para volver al paso anterior.
- **Cargar datos de las personas:**
  - Seleccionar cada asiento y cargarle el nombre.
- **Compra de entradas:**
  - Habrá un botón que permita hacer la compra una vez que se decida realizarla.
  - Agregar una opción para volver al paso anterior.

6. Autenticación y Comunicación

- Toda la comunicación entre servicios se hace mediante **JSON**.
- Los servicios deben estar autenticados mediante **tokens JWT**, ya sea entre el backend y el servicio de la cátedra como el backend contra el servicio proxy.
- Cada servicio debe estar configurado con sus credenciales para poder consumir otros servicios.

7. Payloads

Aquí se encontrarán ejemplos de payloads de los mensajes que maneja el servicio de la cátedra.

Payload 1 - Registro de sesión del alumno

Este endpoint se utiliza para registrar un usuario en el sistema de la cátedra. Durante el registro se obtiene el token de conexión. Si no se anotó el token se puede hacer inicio de sesión y obtener el token.

URL: http://SERVIDOR:PUERTO/api/v1/agregar_usuario
Método: POST
JSON entrada:

```
{
  "username": "juan",
  "password": "juan123",
  "firstName": "Juan",
  "lastName": "Perez",
  "email": "juan@perez.com.ar",
  "nombreAlumno": "Juan Perez",
  "descripcionProyecto": "Proyecto de Juan Perez"
}
```

JSON salida:

```
{
  "creado": true,
  "resultado": "Usuario creado",
  "token":
"eyJhbGciOiJITzUxMiJ9.eyJzdWIiOiJmZXJuYW5kbyIsImV4cCI6MTc1ODIyMTY5
MSwiYXV0aCI6IlJPTEVfVVNFUiIsImlhdCI6MTc1NTYyOTY5MX0.DtQ5-CqQj_tysA
KeqS6OS6IcdWkPgdjbNjjT-GROHkymvTxRmz1NKlf89awBnsyoODQPXBbXmqfBuHut
vmLX5w"
}
```

Payload 2 - Login de usuario

Este endpoint sirve para iniciar sesión con un usuario ya creado. El resultado es un mensaje con el token

URL: http://SERVIDOR:PUERTO/api/v1/agregar_usuario
Método: POST
JSON entrada:

```
{
  "username": "juan",
  "password": "juan123",
  "rememberMe": false
}
```

JSON salida:

```
{
  "id_token":
"eyJhbGciOiJITzUxMiJ9.eyJzdWIiOiJmZXJuYW5kbyIsImV4cCI6MTc1ODIyMTY5
MSwiYXV0aCI6IlJPTEVfVVNFUiIsImlhdCI6MTc1NTYyOTY5MX0.DtQ5-CqQj_tysA
KeqS6OS6IcdWkPgdjbNjjT-GROHkymvTxRmz1NKlf89awBnsyoODQPXBbXmqfBuHut
vmLX5w"
}
```

Payload 3 - Listado completo de eventos (datos resumidos)

Este endpoint devuelve una lista completa de eventos activos pero con datos resumidos para utilizar en una lista simple de eventos.
URL: http://SERVIDOR:PUERTO/api/endpoints/v1/eventos-resumidos
Método: GET

JSON salida:


```
[
  {
    "titulo": "Conferencia Nerd",
    "resumen": "Esta es una conferencia de Nerds",
    "descripcion": "Esta es una conferencia de prueba para
verificar que los datos están correctos",
    "fecha": "2025-11-10T11:00:00Z",
    "precioEntrada": 2500.00,
    "eventoTipo": {
      "nombre": "Conferencia",
      "descripcion": "Conferencia"
    },
    "id": 1
  },
  {
    "titulo": "Otra Conferencia Nerd",
    "resumen": "Esta es otra conferencia de Nerds",
    "descripcion": "Esta es una conferencia de prueba para
verificar que los datos están correctos version 2",
    "fecha": "2025-12-12T14:00:00Z",
    "precioEntrada": 4500.00,
    "eventoTipo": {
      "nombre": "Obra de teatro",
      "descripcion": "Obra de teatro"
    },
    "id": 2
  }
]
```


Payload 4 - Listado completo de eventos (con todos los datos)

Este endpoint devuelve una lista completa de eventos activos.
URL: http://SERVIDOR:PUERTO/api/endpoints/v1/eventos
Método: GET

JSON salida:


```
[
  {
    "titulo": "Conferencia Nerd",
    "resumen": "Esta es una conferencia de Nerds",
```

```
    "descripcion": "Esta es una conferencia de prueba para
verificar que los datos están correctos",
    "fecha": "2025-11-10T11:00:00Z",
    "direccion": "Aula magna de la Universidad de Mendoza",
    "imagen":
"https://scontent-scl2-1.xx.fbcdn.net/v/t1.6435-9/78167441_1316997
891841734_2833734909829316608_n.jpg?stp=dst-jpg_p960x960_tt6&_nc_c
at=105&ccb=1-7&_nc_sid=127cfc&_nc_ohc=1EcS_p0lVGUQ7kNvwE1jrdM&_nc_
oc=AdnCq7-RyHdhVpcPvW46Wehv10cjB9rcujwllkpiPP4H0OWUdbvCG0ygHh4zuXM
QuIY&_nc_zt=23&_nc_ht=scontent-scl2-1.xx&_nc_gid=_fzHXJUwDiUHBphj9
6cIcw&oh=00_AfUm0lIje5462zK3WbGACoar67RKotbEXLqM6MeGpJdPoA&oe=68B9
E878",
    "filaAsientos": 10,
    "columnAsientos": 20,
    "precioEntrada": 2500.00,
    "eventoTipo": {
      "nombre": "Conferencia",
      "descripcion": "Conferencia"
    },
    "integrantes": [
      {
        "nombre": "María",
        "apellido": "Corvalán",
        "identificacion": "Dra."
      }
    ],
    "id": 1
  },
  {
    "titulo": "Otra Conferencia Nerd",
    "resumen": "Esta es otra conferencia de Nerds",
    "descripcion": "Esta es una conferencia de prueba para
verificar que los datos están correctos version 2",
    "fecha": "2025-12-12T14:00:00Z",
    "direccion": "Aula magna de la Universidad de Mendoza",
    "imagen":
"https://scontent-scl2-1.xx.fbcdn.net/v/t39.30808-6/498246488_1010
495731278033_6589798781026206970_n.jpg?_nc_cat=100&ccb=1-7&_nc_sid
=127cfc&_nc_ohc=nNqMZ6Q01LwQ7kNvwFwn-Dp&_nc_oc=Adka8XbkxUljf0sqN4f
az1zUFFIKEAD5XKS-gZEumNPJNcSkMTfB92azVHLQYnKxqeI&_nc_zt=23&_nc_ht=
scontent-scl2-1.xx&_nc_gid=WzFrlM29lDxhAEfynHPxFQ&oh=00_AfVoCftNVc
xpatawm7mG3mA-FqqiwqSa-66iHFYp9Hw65Q&oe=689863E1",
    "filaAsientos": 12,
    "columnAsientos": 18,
    "precioEntrada": 4500.00,
    "eventoTipo": {
      "nombre": "Obra de teatro",
      "descripcion": "Obra de teatro"
```

```
    },
    "integrantes": [
      {
        "nombre": "María",
        "apellido": "Corvalán",
        "identificacion": "Dra."
      },
      {
        "nombre": "Ricardo",
        "apellido": "Tapia",
        "identificacion": "Profesor"
      }
    ],
    "id": 2
  }
]
```


Payload 5 - Datos completos de un evento

Este endpoint devuelve el dato completo de un evento pasando como parámetro el Id del evento.
URL: http://SERVIDOR:PUERTO/api/endpoints/v1/evento/{id}
Método: GET

JSON salida:


```
{
  "titulo": "Conferencia Nerd",
  "resumen": "Esta es una conferencia de Nerds",
  "descripcion": "Esta es una conferencia de prueba para verificar
que los datos están correctos",
  "fecha": "2025-11-10T11:00:00Z",
  "direccion": "Aula magna de la Universidad de Mendoza",
  "imagen":
"https://scontent-scl2-1.xx.fbcdn.net/v/t1.6435-9/78167441_1316997
891841734_2833734909829316608_n.jpg?stp=dst-jpg_p960x960_tt6&_nc_c
at=105&ccb=1-7&_nc_sid=127cfc&_nc_ohc=1EcS_p0lVGUQ7kNvwE1jrdM&_nc_
oc=AdnCq7-RyHdhVpcPvW46Wehv10cjB9rcujwllkpiPP4H0OWUdbvCG0ygHh4zuXM
QuIY&_nc_zt=23&_nc_ht=scontent-scl2-1.xx&_nc_gid=_fzHXJUwDiUHBphj9
6cIcw&oh=00_AfUm0lIje5462zK3WbGACoar67RKotbEXLqM6MeGpJdPoA&oe=68B9
E878",
  "filaAsientos": 10,
  "columnAsientos": 20,
  "precioEntrada": 2500.00,
  "eventoTipo": {
    "nombre": "Conferencia",
    "descripcion": "Conferencia"
  },
```

```
  "integrantes": [
    {
      "nombre": "María",
      "apellido": "Corvalán",
      "identificacion": "Dra."
    }
  ],
  "id": 1
}
```

Payload 6 - Bloqueo de asientos en un evento

A este endpoint le tenemos que mandar los datos del evento y los asientos que queremos bloquear. El JSON respuesta puede tener “resultado” verdadero o falso si los asientos pudieron ser bloqueados o no.
Si el bloqueo es falso puede deberse a tratar de reservar un asiento ya reservado (“Bloqueado”) o ya vendido (“Ocupado”)
URL: http://SERVIDOR:PUERTO/api/endpoints/v1/bloquear-asientos
Método: POST
JSON entrada:

```
{
  "eventoId": 1,
  "asientos": [
    {
      "fila": 2,
      "columna": 1
    },
    {
      "fila": 2,
      "columna": 2
    }
  ]
}
```

JSON salida (ejemplo 1):

```
{
  "resultado": false,
  "descripcion": "No todos los asientos pueden ser bloqueados",
  "eventoId": 1,
  "asientos": [
    {
      "estado": "Ocupado",
      "fila": 2,
      "columna": 1
    },
    {
      "estado": "Ocupado",
```

```
      "fila": 2,
      "columna": 2
    }
  ]
}
```

JSON salida (ejemplo 2):

```
{
  "resultado": true,
  "descripcion": "Asientos bloqueados con exito",
  "eventoId": 1,
  "asientos": [
    {
      "estado": "Bloqueo exitoso",
      "fila": 2,
      "columna": 3
    },
    {
      "estado": "Bloqueo exitoso",
      "fila": 2,
      "columna": 4
    }
  ]
}
```

Payload 7 - Venta de asientos por evento

A este endpoint le tenemos que mandar los datos del evento y los asientos que queremos vender. El JSON respuesta puede tener “resultado” verdadero o falso si la venta es exitosa o no.
Si la venta es falsa puede deberse a tratar de vender un asiento no reservado (“Libre”) o ya vendido (“Ocupado”)
Si la venta es verdadera el estado de los asientos es “Vendido”
URL: http://SERVIDOR:PUERTO/api/endpoints/v1/realizar-venta
Método: POST
JSON entrada:

```
{
  "eventoId": 1,
  "fecha": "2025-08-17T20:00:00.000Z",
  "precioVenta": 1400.10,
  "asientos": [
    {
      "fila": 2,
      "columna": 3,
      "persona": "Fernando Galvez"
    },
    {
```

```
      "fila": 2,
      "columna": 4,
      "persona": "Carlos Perez"
    }
  ]
}
```


JSON salida (ejemplo 1):


```
{
  "eventoId": 1,
  "ventaId": null,
  "fechaVenta": "2025-08-24T23:18:07.541151Z",
  "asientos": [
    {
      "fila": 2,
      "columna": 3,
      "persona": "Fernando Galvez",
      "estado": "Libre"
    },
    {
      "fila": 2,
      "columna": 4,
      "persona": "Carlos Perez",
      "estado": "Libre"
    }
  ],
  "resultado": false,
  "descripcion": "Venta rechazada. Alguno de los asientos no se
encontraban bloqueados para la venta.",
  "precioVenta": 1400.0
}
```


JSON salida (ejemplo 2):


```
{
  "eventoId": 1,
  "ventaId": 1506,
  "fechaVenta": "2025-08-24T23:18:41.974720Z",
  "asientos": [
    {
      "fila": 2,
      "columna": 3,
      "persona": "Fernando Galvez",
      "estado": "Vendido"
    },
    {
      "fila": 2,
```

```
      "columna": 4,
      "persona": "Carlos Perez",
      "estado": "Vendido"
    }
  ],
  "resultado": true,
  "descripcion": "Venta realizada con exito",
  "precioVenta": 1400.0
}
```


Payload 8 - Listado completo de ventas por cada alumno (datos resumidos)

Este endpoint devuelve un listado con datos resumidos de las ventas (exitosas y fallidas) que ha realizado el alumno.

URL: http://SERVIDOR:PUERTO/api/endpoints/v1/listar-ventas
Método: GET

JSON salida:


```
[
  {
    "eventoId": 1,
    "ventaId": 1503,
    "fechaVenta": "2025-08-23T22:51:02.574851Z",
    "resultado": false,
    "descripcion": "Venta rechazada. Alguno de los asientos no se
encontraban bloqueados para la venta.",
    "precioVenta": 1200.1,
    "cantidadAsientos": 0
  },
  {
    "eventoId": 1,
    "ventaId": 1504,
    "fechaVenta": "2025-08-23T22:51:15.101553Z",
    "resultado": true,
    "descripcion": "Venta realizada con exito",
    "precioVenta": 1200.1,
    "cantidadAsientos": 2
  },
  {
    "eventoId": 1,
    "ventaId": 1505,
    "fechaVenta": "2025-08-24T23:18:07.541151Z",
    "resultado": false,
    "descripcion": "Venta rechazada. Alguno de los asientos no se
encontraban bloqueados para la venta.",
    "precioVenta": 1400.1,
```

```
    "cantidadAsientos": 0
  },
  {
    "eventoId": 1,
    "ventaId": 1506,
    "fechaVenta": "2025-08-24T23:18:41.974720Z",
    "resultado": true,
    "descripcion": "Venta realizada con exito",
    "precioVenta": 1400.1,
    "cantidadAsientos": 2
  },
  {
    "eventoId": 1,
    "ventaId": 1507,
    "fechaVenta": "2025-08-24T23:23:05.622646Z",
    "resultado": false,
    "descripcion": "Venta rechazada. Alguno de los asientos no se
encontraban bloqueados para la venta.",
    "precioVenta": 1400.1,
    "cantidadAsientos": 0
  }
]
```


Payload 9 - Ver datos de una venta en particular

Este endpoint devuelve el dato detallado de una venta particular identificada por el Id.

URL: http://SERVIDOR:PUERTO/api/endpoints/v1/listar-venta/{id}
Método: GET

JSON salida 1 (venta fallida):


```
{
  "eventoId": 1,
  "ventaId": 1503,
  "fechaVenta": "2025-08-23T22:51:02.574851Z",
  "asientos": [],
  "resultado": false,
  "descripcion": "Venta rechazada. Alguno de los asientos no se
encontraban bloqueados para la venta.",
  "precioVenta": 1200.1
}
```


JSON salida 2 (venta exitosa):


```
{
  "eventoId": 1,
  "ventaId": 1504,
  "fechaVenta": "2025-08-23T22:51:15.101553Z",
```

```
  "asientos": [
    {
      "fila": 2,
      "columna": 1,
      "persona": "Fernando Villarreal",
      "estado": "Ocupado"
    },
    {
      "fila": 2,
      "columna": 2,
      "persona": "Carlos Perez",
      "estado": "Ocupado"
    }
  ],
  "resultado": true,
  "descripcion": "Venta realizada con exito",
  "precioVenta": 1200.1
}
```