

Proyecto curso

"Formación School of Tech Software Capital Market" [Full Stack]

LucaTicket

[Desarrollo de la gestión Web de venta online de entradas a eventos]

[Lucatic][Accenture]

26/01/2026 (Online)

ÍNDICE DE CONTENIDOS

Idea de referencia	3
Trabajo a realizar a lo largo del proyecto.....	3
Normas.....	4
ANEXO 01 – Proceso de compra	5
ANEXO 02 – Microservicio Banco.....	6
ANEXO 02a – Microservicio Banco. Información básica	6
ANEXO 02b – Microservicio Banco. Validar el usuario.....	7
ANEXO 02c – Microservicio Banco. Validar el pago	8
ANEXO 03 – Orden acciones	10

Idea de referencia

LucaTicket quiere convertirse en la referencia de las web de venta de tickets para eventos (sobre todo festivales de música)

En la primera parte del desarrollo de este software, se van a implementar las aplicaciones de gestión de eventos, y venta de tickets y se ha pensado en una arquitectura de microservicios para montarlo.

El modelo se va a simplificar mucho. Solo se quiere trabajar con los elementos principales.

Trabajo a realizar a lo largo del proyecto

Este proyecto tendrá 2 servicios básicos. El grueso del proyecto **se centrará de forma exclusiva en la creación de microservicios** y dispondrá de las siguientes tareas:

a. Gestión de eventos

1. (C) Alta de eventos
2. (R) Detalle de un evento
3. (U) Modificación de eventos
4. (D) Baja de eventos
5. Listado de eventos

b. Venta de entradas

1. Se emplea para simular la venta de entradas
2. Solo dispone de un método llamado “compraEntradas” (Ver Anexo 1)

NOTA 01: El microservicio de eventos emplea una BBDD MySQL

NOTA 02: El sistema de ventas/pago se encarga de realizar la compra y validar la venta para asociar el usuario y la entrada comprada. Debería incluir un sistema que valide la compra a través de “pasarela de pago”.

Ese servicio, externo y ajeno a nuestro sistema, devolverá un JSON con la información y los códigos HTTP de lo que ha ocurrido con la compra.

El servicio ya existe. No debe implementarse. Solo usarlo (ver Anexos 01 y 02)

NOTA 03: Del evento interesa: nombre, descripción, fecha evento, hora evento, precio mínimo, precio máximo, localidad, género, nombre del recinto

NOTA 04: El recinto será un String para simplificar

Normas

Para este proyecto hay una serie de normas o prioridades. Cuantas más cumplas mejor:

- **Usar Java + Spring + SpringBoot + Spring DATA + REST + Microservicios+ Angular**
- Emplear **logs de sistema** para mostrar información
- **Documentación:** Generada a partir de los scripts. La documentación del servicio REST se haría con OpenApi.
- **Uso de repositorios:** para compartir las versiones de código (GitHub, GitLab, Bitbucket)
- **BBDD para la parte de eventos con MySQL.**
 - Utiliza la IA para obtener al menos 20 registros para pruebas.
 - Además, se permite usar la IA para crear las tablas siempre y cuando primero se piense lo que se vaya a necesitar
- **Pruebas unitarias:** define, de forma previa, varias pruebas unitarias.
 - Pueden ser sencillas.
 - Es erróneo si no se planifican las pruebas antes de crear el método o funcionalidad (por lo menos hay que pensarla de forma previa)
 - Puedes usar JUnit, Mockito, AssertJ, HamCrest, REST Assured, etc.
 - Recuerda validar los JSON (posiblemente pruebas de integración en algún caso).
 - Por cada prueba que pienses y realices, se permiten hacer 2 pruebas extras con IA

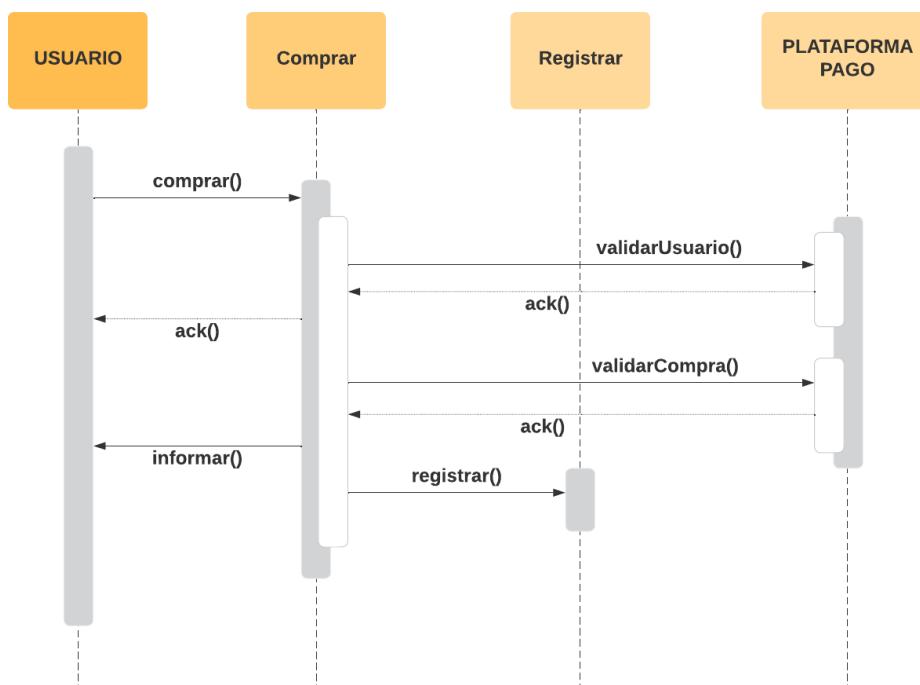
ANEXO 01 – Proceso de compra

Para realizar la tramitación de una compra, se realizará el siguiente esquema:

- El Servicio de compra tiene un método (“compraEntradas”) que va a recibir solo tres elementos/objetos de entrada:
 - Mail (String) al que se asociarán las entradas compradas.
 - Datos (ID) del evento a realizar la compra
 - Datos de la tarjeta del titular

Datos Tarjeta	
nombreTitular	Persona a la que se realiza el cargo Debe tener el formato estándar que figura en la tarjeta
numeroTarjeta	Número de tarjeta valido. Solo admitimos tarjetas Visa y Mastercard
mesCaducidad	Mes en el que caduca la tarjeta
yearCaducidad	Año en el que caduca la tarjeta
cvv	Código Valor de verificación de la tarjeta

- Para validar la compra, se pedirá autorización a la plataforma de pago (microservicio externo que ya está implementado), según los datos especificados del Anexo 02b y 02c
- Tanto si la compra ha resultado un éxito o un fracaso, debe avisarse al usuario
- (***) En el caso de éxito, además, se registrará la compra de una entrada al evento, su precio, fecha de compra y mail registrado. Se simulará el envío de un mail



ANEXO 02 – Microservicio Banco

ANEXO 02a – Microservicio Banco. Información básica

LucaTicket tiene contratado un servicio para facilitar la compra de estas entradas a todos sus clientes. Ese servicio lo proporciona LucaBanking, una empresa que permite validar las compras realizadas. Dispone de un API que permitirá controlar los pagos emitidos por el correspondiente comercio (en este caso LucaTicket).

Como se ha comentado, para validar la compra, se pedirá autorización a la plataforma de pago (microservicio externo que ya está implementado). Ese proceso consta de dos pasos: autenticación y autorización de la compra.

- PASO 01: Autenticación
 - La plataforma necesita unos datos para garantizar que es un usuario con permisos (ver Anexo 02b). Ésta sería la dirección

HOST	http://
ENDPOINT	/pasarela/validaruser (POST)

- PASO 02: Validar compra
 - Una vez validado el usuario, se podrán enviar datos para validar la compra (ver Anexo 2c). Ésta sería la dirección

HOST	http://
ENDPOINT	/pasarela/validación (POST)

NOTA DEL PROFESOR:

Por decisiones formativas, la implementación del proceso anterior completo -que de hecho sería el proceso lógico-, se va a retrasar en el orden de las acciones a acometer por parte del alumno (será la FASE 06). En primera instancia (FASE 01) el alumno no va a implementar la validación del usuario y directamente validará la compra (tal y como se verá en el Anexo 2c).

En este caso la dirección sería

HOST	http://
ENDPOINT	/pasarela/compra (POST)

ANEXO 02b – Microservicio Banco. Validar el usuario**DATOS DE ENTRADA**

Para garantizar un servicio exclusivo, el servicio de validación de pagos se encuentra securizado. La única forma de acceder a él es incluyendo en la cabecera de la petición un token válido para garantizar la autenticación del individuo que realiza la compra.

Para este paso primero (autenticación), la plataforma necesita unos datos para garantizar que es un usuario con permisos. Una vez validado el usuario, se podrán enviar datos para validar la compra (tal y como se desarrolla en el Anexo 2c)

Para pedir el token inicial, el individuo debe realizar una petición POST que incluya en el Body un valor de user y otro de password

HOST DE ACCESO	http://
ENDPOINT	/pasarela/validaruser
user	yofuilechon
password	AntoniosRules

El token que se reciba de vuelta deberá emplearse en las peticiones al endPoint (/pasarela/validación) para validar la futura compra.

ANEXO 02c - Microservicio Banco. Validar el pago

DATOS DE ENTRADA

Para poder validar la compra, es necesario proporcionar una serie de datos en la entrada que corresponde a la información de la transacción a realizar. Todos son obligatorios.

Todos los datos se proporcionan como String en la petición, aunque, lógicamente, datos como la cantidad (por ejemplo) se trabajen de forma interna en LucaBanking como números.

Datos Tarjeta	
nombreTitular	Persona a la que se realiza el cargo Debe tener el formato estándar que figura en la tarjeta
numeroTarjeta	Número de tarjeta valido. Solo admitimos tarjetas Visa y Mastercard
mesCaducidad	Mes en el que caduca la tarjeta
yearCaducidad	Año en el que caduca la tarjeta
cvv	Código Valor de verificación de la tarjeta
emisor	Entidad o comercio que emite la venta
concepto	Concepto a indicar que figurará en la compra
cantidad	Cantidad en euros de la compra realizada. El precio se simulará como un valor random entre el mínimo y el máximo

Un ejemplo para realizar una prueba correcta sería el siguiente

```
{
    "nombreTitular": "Antonio Santos Ramos",
    "numeroTarjeta": "4624-0071-8793-4978",
    "mesCaducidad": "12",
    "yearCaducidad": "2026",
    "cvv": "123",
    "emisor": "LucaTicket",
    "concepto": "Entrada a concierto",
    "cantidad": "100.5"
}
```

DATOS DE SALIDA

Al realizar la transacción pueden ocurrir distintas situaciones

CÓDIGO	DESCRIPCIÓN
200.0001	Transacción correcta
400.0001 (+)	No hay fondos suficientes en la cuenta
400.0002 (+)	No se encuentran los datos del cliente
400.0003	El número de la tarjeta no es válido
400.0004	El formato del cvv no es válido
400.0005	El mes (caducidad) no es correcto
400.0006	El año (caducidad) no es correcto
400.0007	La fecha de caducidad debe ser posterior al día actual
400.0008	El formato del nombre no es correcto
500.0001 (+)	El sistema se encuentra inestable

(+) Las desgracias a veces ocurren sin que las esperes... 😞

Tras la petición realizada se devolverá un JSON con información de lo que ha ocurrido en la transacción.

Si los datos son incorrectos con respecto al formato y las validaciones de la tarjeta, se obtendrá información para solucionar el problema.

Como se observa en la imagen de la derecha, a esos datos se le añadirá la información proporcionada en la entrada.

NOTA: LucaBanking tiene fama contrastada (basta con conocer a su CEO 😞) de que la información que proporcionan tiene cierto carácter ofensivo y no puede utilizarse de forma directa. Consultada al respecto, la empresa nos ha llamado lechones y ha declinado cualquier comentario al respecto.

```
{
  "timestamp": "",
  "status": "",
  "error": "",
  "message": [],
  "info": {
    "nombreTitular": "",
    "numeroTarjeta": "",
    "mesCaducidad": ↴,
    "yearCaducidad": ↴,
    "cvv": ↴,
    "emisor": "",
    "concepto": "",
    "cantidad": "",
  },
  "infoadicional": ""
}
```

Si el proceso se hace sin validación de usuario, la dirección sería

HOST	http://
ENDPOINT	/pasarela/compra (POST)

En el caso de que hayamos validado el usuario y tengamos un token , la dirección sería

HOST	http://
ENDPOINT	/pasarela/validación (POST)

ANEXO 03 – Orden acciones

En este proyecto no se trata de realizar todo de golpe, sino de ir realizando distintas acciones. Por este motivo, el proyecto se realiza en varias fases

1. FASE 01: **Microservicio** (solo back) de gestión de eventos. CRUD completo
 - a. Eureka
 - b. Pruebas de funcionamiento
2. FASE 02: **Microservicio** de compra (versión sencilla sin validación usuarios)
 - a. URL /pasarela/compra (POST)
 - b. Realizar Pruebas de funcionamiento
3. FASE 03: Realizar un Front (Con **Angular**) para el Listado de eventos
4. FASE 04: Elegir algunos (o todos) estos elementos
 - Gateway
 - Servidor de configuraciones
 - Circuit Breaker
5. FASE 05: Completar el CRUD de eventos para **Angular**
 - Realizar pruebas de funcionamiento
6. FASE 06: Realizar proceso de compra con validación
 - Validar usuario: /pasarela/validaruser (POST)
 - Validar compra: /pasarela/validación (POST)
7. FASE 07: Registrar compra de la entrada
8. FASE 08: Dar un mensaje de compra a medida/custom
9. FASE 09: Implementar un sistema de caché para optimizar las peticiones sobre el microservicio de eventos