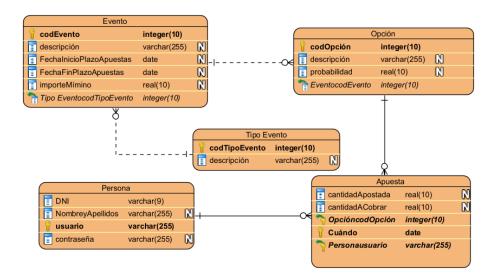
# **CRITERIOS DE CORRECIÓN**

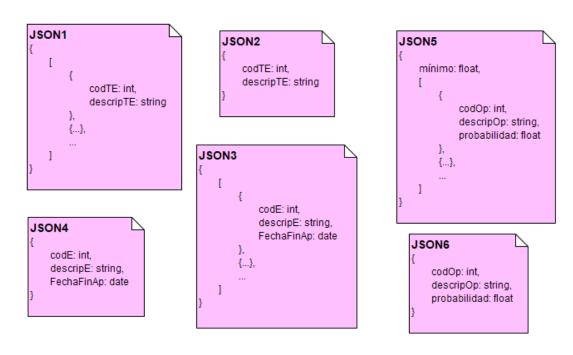
Criterio		Ptos.
Esquema relacional de la base de datos	$\rightarrow$	0,4
Relaciones del modelo de dominio en el diagrama de clases	$\rightarrow$	0,4
Gestores y atributos necesarios en el diagrama de clases	$\rightarrow$	0,2
Listar tipos de evento	$\rightarrow$	0,5
Listar eventos	$\rightarrow$	0,6
Comprobar plazo de apuesta de los eventos	$\rightarrow$	0,4
Listar opciones de apuesta	$\rightarrow$	1,0
Comprobar importe mínimo apuesta	$\rightarrow$	0,7
Comprobar usuario repetido	$\rightarrow$	0,6
Registrar usuario (Objetos)	$\rightarrow$	0,6
Registrar usuario (Base de datos)	$\rightarrow$	0,4
Identificación	$\rightarrow$	0,7
Calcular ganancias	$\rightarrow$	1,5
Almacenar apuesta (Objetos)	$\rightarrow$	0,6
Almacenar apuesta (Base de datos)	$\rightarrow$	0,4
Bucles	$\rightarrow$	0,2
Condiciones	$\rightarrow$	0,3
Parámetros	$\rightarrow$	0,25
JSON	$\rightarrow$	0,25

El diagrama de clases, la estructura de los JSON y los diagramas de secuencia que se presentan en esta solución son una de las múltiples posibilidades correctas de implementación del caso de uso.

#### **Esquema Relacional**



### Estructuras JSON (una de las muchas posibilidades correctas)



#### Diagrama de clases (una de las muchas posibilidades correctas)

Persona -DNI: string -NombreYApellidos: string

+Persona(dni : string, nomyap : string, usu : string, pwd : string) : Persona

-contraseña: string

+getUsuario(): string +comprobarContraseña(pwd: string): boolean

Evento

codEvento : string

FechalnicioPlazoApuestas: date importeMinimo: float

-susOpciones : Opción[] +estaAbierto(cuando : date) : boolean

getCodEvento():int

obtenerOpciones(): JSON5

+calcularApuestasResto(codOp : int, laPersona : Persona) : float +apostar(codOp : int, laPersona : Persona, cantidadapostar : float, cantidadcobrar : float, cuando : date)

Opción

-codOpción : int descripción : string probabilidad : float susApuestas : Apuesta[]

+getInfo(): JSON6 getCodigo(): int

apostar(laPersona : Persona, cantidadapostar : float, cantidadcobrar : float, cuando : date)

Tipo Evento

-codTipoEvento : int -descripción : string -susEventos : Evento[]

+obtenerInfo(): JSON2 +getCodTipoEvento(): int +obtenerEventosAbiertos(): JSON3

Apuesta

cuando : date -cantidadApostada: float cantidadACobrar: float

deQuién : Persona

+getDeQuien(): Persona +getCantidadApostada(): float

+Apuesta(laPersona : Persona, cantidadapostar : float, cantidadcobrar : float, cuando : date) : Apuesta

Gestor Tipo Eventos

todosTipos : Tipo Evento[] +obtenerTipoEventos(): JSON1

+buscarTipoEvento(codTE: int): Tipo Evento +obtenerEventosAbiertos(elTipoEvento: Tipo Evento): JSON3

+obtenerTipoEventos(): JSON1 +obtenerEventosAbiertos(codTE: int): JSON3 +obtenerOpciones(codE : int) : JSON5

+registrarPersona(DNI: string, nomyap: string, usu: string, pwd: string): boolean

+identificar(usu:string, pwd:string):boolean +calcularGanancias(codE:int, codOp:int, probOp:float, cantidad:float, usu:string):float

+apostar(codE: int, codOp, usu, cantidadapostar, cantidadcobrar)

Gestor Persona

-todasPersonas : Persona[] +registrarPersona(dni : string, nombreyap : string, usu : string, pwd : string) : void

+buscarPersona(usu : string) : Persona +comprobarContraseña(laPersona : Persona, pwd : string) : boolean

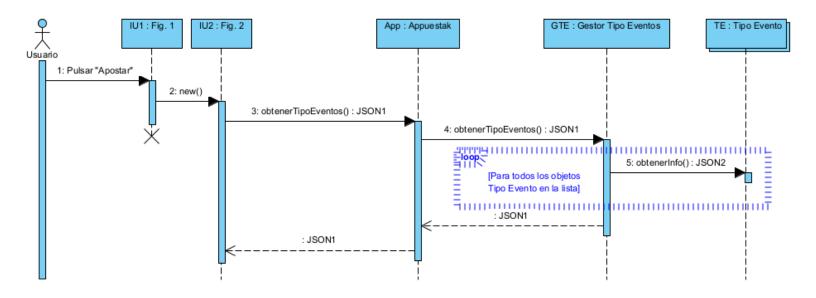
Gestor Eventos

-todosEventos : Evento[]

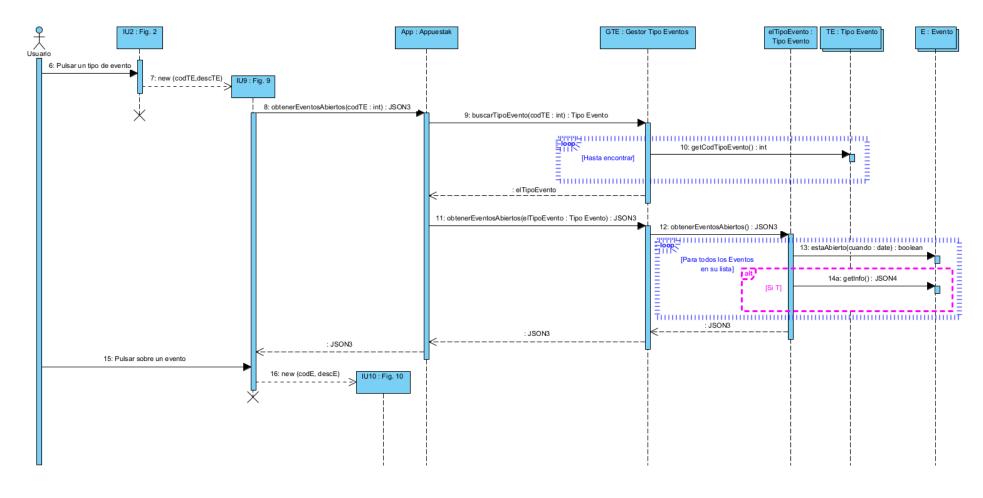
+obtenerOpciones(elEvento : Evento) : JSON5 +calcularApuestasResto(elEvento : Evento, codOp : int, laPersona : Persona) : float

apostar(elEvento: Evento, la Persona: Persona, usu: string, codOp: int, cantidadapostar: float, cantidadcobrar: float, cuando: date)

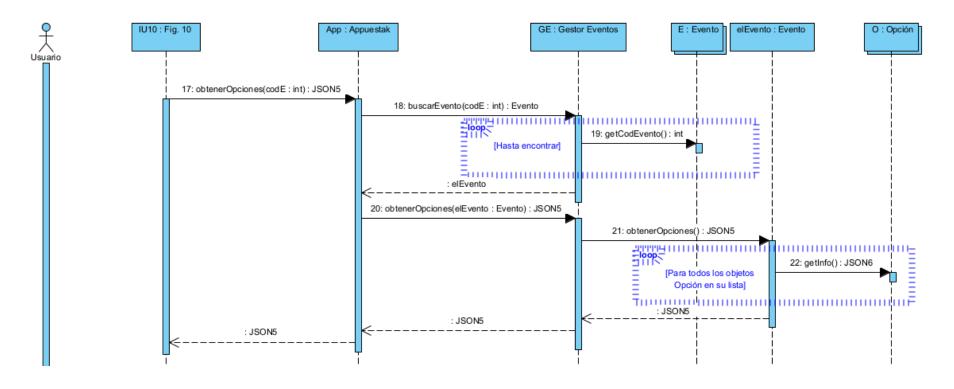
# Listar tipos de evento



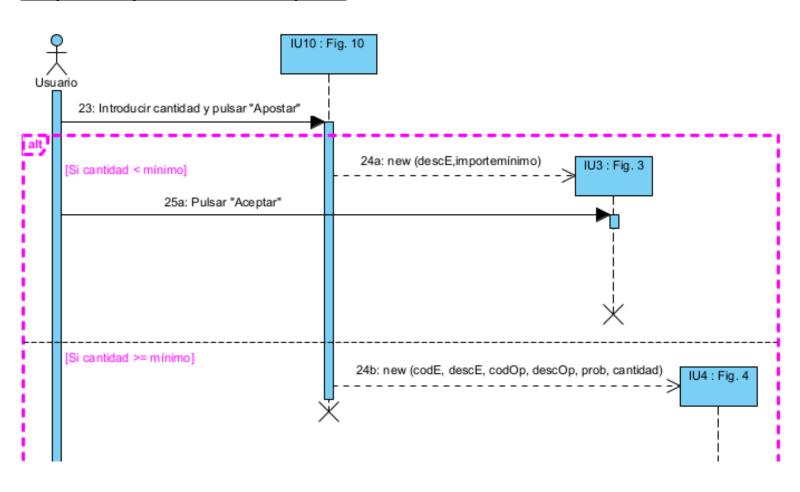
#### Listar eventos del tipo seleccionado en los que se pueda apostar



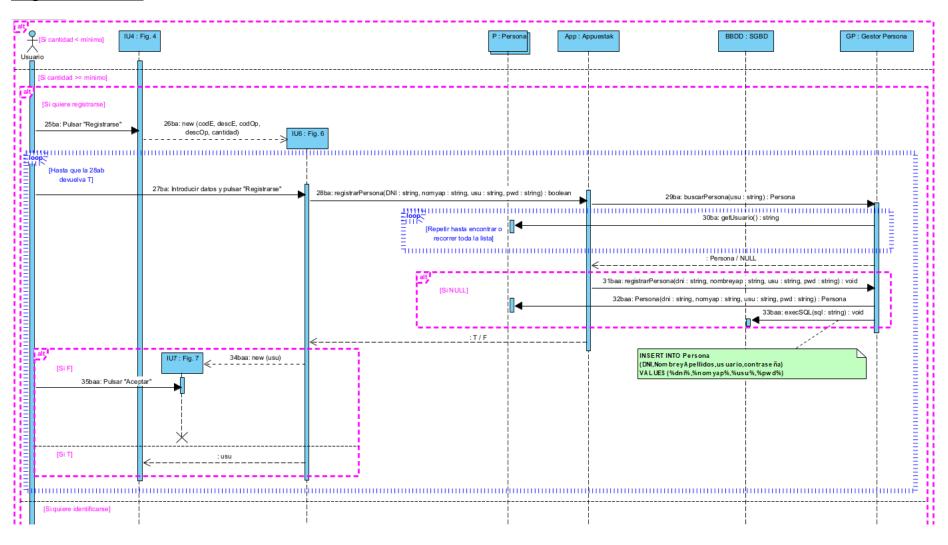
## Listar opciones de apuesta del evento seleccionado



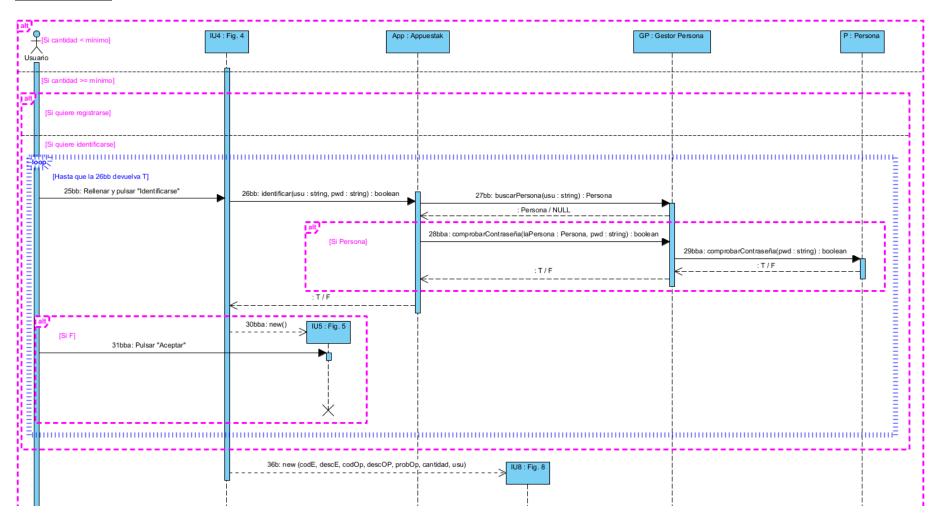
# Comprobar importe mínimo de la apuesta



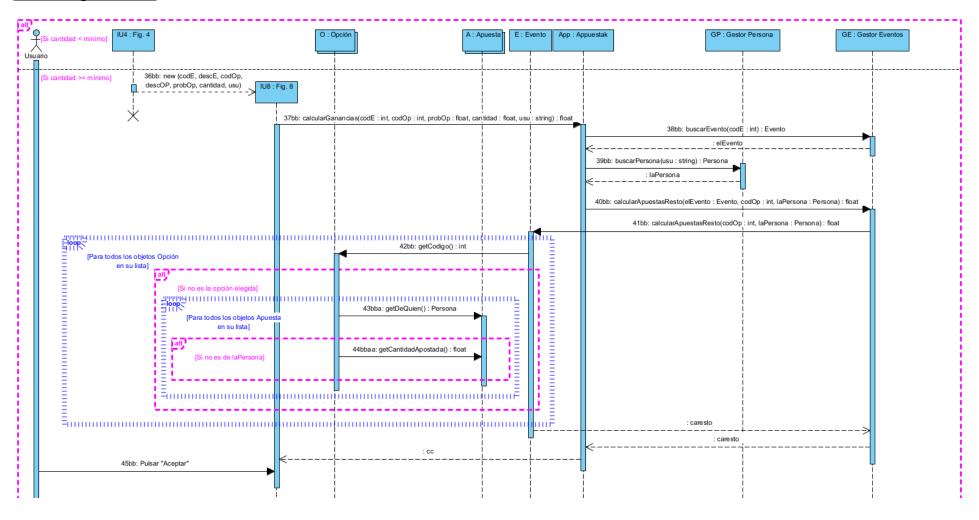
#### **Registrar usuario**



## **Identificarse**



#### **Calcular ganancias**



#### Almacenar apuesta

