GRUPO LEONCITOS

MÉTODO DEL CASO: CUDECA

Miembros del grupo

Assunçao Ocampo, Arturo Francisco
González Díaz, Sergio
Leal Huertas, Álvaro
León Becerra, Clara
Moreno Peralta, Héctor
Ramírez Girón, José Antonio
Salas Trujillo, Yago
Sánchez Alarcón, Javier
Soler Ortega, Andrea
Torres Méndez, Pablo
Torres Ovalle, Samuel
Velasco Soto, Regina

ÍNDICE

1. IDENTIFICAR ACTORES

En el sistema se identifican los siguientes actores principales y secundarios, que actúan con las distintas funcionalidades de dicho sistema:

- Administrador:
    - Es el encargado de la gestión global del sistema. Tiene acceso a las principales gestiones del sistema, como sería la creación, modificación y control de los diferentes eventos, así como la gestión de los tickets disponibles y de sus distintos precios. Su rol es garantizar el correcto funcionamiento de la plataforma y mantener actualizada la información de los eventos.
- Usuario:
    - Representa a cualquier persona que utiliza la plataforma para consultar los diferentes eventos, hacerse miembro, comprar entradas, realizar donaciones o participar como voluntario en algún evento. Su interacción con el sistema se centra en el consumo de servicios y la participación en las distintas actividades ofrecidas.
- Sistema de pago:
    - Este actor representa un servicio externo que se encarga de procesar los pagos que hacen los usuarios al comprar entradas o hacer donaciones. No pertenece al sistema principal, pero se conecta con él para completar las transacciones.

2. LÍMITES DEL SISTEMA

El sistema abarca sólo las funciones necesarias para organizar y participar en eventos. Las tareas que dependen de servicios externos, como los pagos, no están incluidas dentro del propio sistema.

Dentro del sistema, se incluyen aquellos procesos de administración y uso de la plataforma, como sería, la gestión de eventos, gestión de tickets, registro e inicio de sesión, compra de entradas, donaciones, voluntarios, generación de certificados o ser socio.

Fuera del sistema está el sistema de pago, que no forma parte del sistema principal. Su función será gestionar los cobros y los procesos de pago de los usuarios. Trabaja como un servicio externo que se conecta cuando se necesita realizar un pago.

3. ALCANCE

El alcance del sistema abarca desde la gestión administrativa de los eventos como la interacción del usuario con la plataforma, de forma que se pueden visualizar los diferentes eventos hasta la participación en los mismos.

Por ello, el sistema permite al administrador realizar las siguientes acciones: iniciar sesión, ver lista de eventos, crear, modificar o eliminar eventos y consultar la disponibilidad y modificar precios de los tickets.

Mientras que en el usuario el sistema permite al mismo registrarse e iniciar sesión, consultar la lista de eventos, comprar entradas, donar y obtener un certificado de donación, hacerse socio y suscribirte a la newsletter y unirse como voluntario a un evento.

4. REQUISITOS

4.1 REQUISITOS FUNCIONALES

- **RF-01: Venta de entradas por tipo de boleto**

  Como: Comprador.

  Quiero: Comprar entradas que se diferencien entre conciertos, eventos, sorteos, rifas, y recibir mi boleto por correo o Whatsapp.

  Para: Tener claro qué tipo de entrada compré y poder acceder al evento sin confusión.

- **RF-02: Identificación completa**

  Como: Comprador.

  Quiero: Rellenar mis datos personales (nombre, apellidos, DNI, teléfono, correo y código postal) al comprar.

  Para: Que la fundación tenga mi información para contacto, certificados, …

- **RF-03: Casillas opcionales**

  Como: Comprador.

  Quiero: Ver casillas opcionales en el formulario (suscripción a newsletter, voluntariado, socio).

  Para: Poder decidir si quiero recibir información, colaborar, registrarme como socio.

- **RF-04: Gestión clientes**

  Como: Fundación Cudeca.

  Quiero: Datos de usuario que se contrastan automáticamente con el CRM y se autocompleta si son socios.

  Para: Evitar duplicados y mantener la información centralizada.

- **RF-05: Generación automática de certificado de donación**

  Como: Donante.

  Quiero: Genere automáticamente certificado de donación con los datos fiscales del donante.

  Para: Tener documento válido para Hacienda.

- **RF-06: Reembolso por cancelación**

  Como: Comprador.

  Quiero: Si el evento se cancela, que devuelvan automáticamente el importe completo y recibir aviso.

  Para: Estar informado y no perder dinero.

- **RF-07: Múltiples métodos de pago**

  Como: Comprador.

  Quiero: Pagar con Bizum, PayPal o tarjeta crédito.

  Para: Usar el método de pago que resulte más cómodo.

- **RF-08: Opción de donación**

  Como: Donante.

  Quiero: Poder aportar dinero al evento durante la compra de la entrada o sin necesidad de comprar entrada.

  Para: Contribuir y apoyar a la fundación.

- **RF-09: Leer/validar códigos QR**

  Como: Cliente.

  Quiero: Validar con QR de entrada desde mi móvil.

  Para: Marcar asistencia más rápido y sencillo.

- **RF-10: Hacerse socio**

Como: Cliente.

Quiero: Poder hacerme socio de la fundación.

Para: Apoyar financieramente la fundación y no tener que rellenar mis datos en cada compra.

4.2 REQUISITOS NO FUNCIONALES

- **RNF-01: Accesibilidad y usabilidad**

  Como: Usuario.

  Quiero: Interfaz sencilla con letra grande y botones visibles y poco cargada de información.

  Para: Poder comprar entradas y donar fácilmente de manera sencilla.

- **RNF-02: Rendimiento**

  Como: Fundación Cudeca.

  Quiero: Soportar múltiples usuarios en picos de demanda.

  Para: Evitar caídas del sistema cuando se lancen entradas de un evento.

- **RNF-03: Protección de datos**

  Como: Fundación Cudeca.

  Quiero: Poder almacenar la información de los usuarios de forma segura cumpliendo con la ley de protección de datos.

  Para: Proteger la privacidad de los usuarios.

- **RNF-04: Robustez del sistema**

  Como: Fundación Cudeca.

  Quiero: Que el sistema se mantenga disponible y no se caiga.

  Para: Que los usuarios puedan comprar entradas cuando quieran.

- **RNF-05: Escalabilidad**

  Como: Fundación Cudeca.

  Quiero: Que la plataforma se adapte a nuevos tipos de entrada en el futuro.

  Para: No tener que rehacer el diseño si queremos expandir.

5. CASOS DE USO

5.1 CASOS DE USO ADMINISTRADOR

- **CU01: Iniciar sesión como administrador**

  El administrador introduce sus credenciales para acceder al sistema y poder realizar operaciones de creación, edición o supervisión de eventos.

- **CU02: Ver lista de eventos**

  El sistema muestra un listado con todos los eventos registrados, incluyendo estado (activo, pendiente, cancelado), tipo, fecha y recaudación.

- **CU03: Gestión de eventos**

  Caso de uso principal que engloba la creación, modificación, aprobación o denegación de eventos, así como su seguimiento general.

- **CU04: Crear evento**

  El administrador define nombre, tipo (concierto, cena, rifa, etc.), fecha, ubicación, precios, aforo y objetivo de recaudación.

- **CU05: Modificar evento**

  Permite cambiar datos del evento como fechas, aforo, precios, descripción o responsables, sin perder el histórico de ventas.

- **CU06: Gestión de tickets**

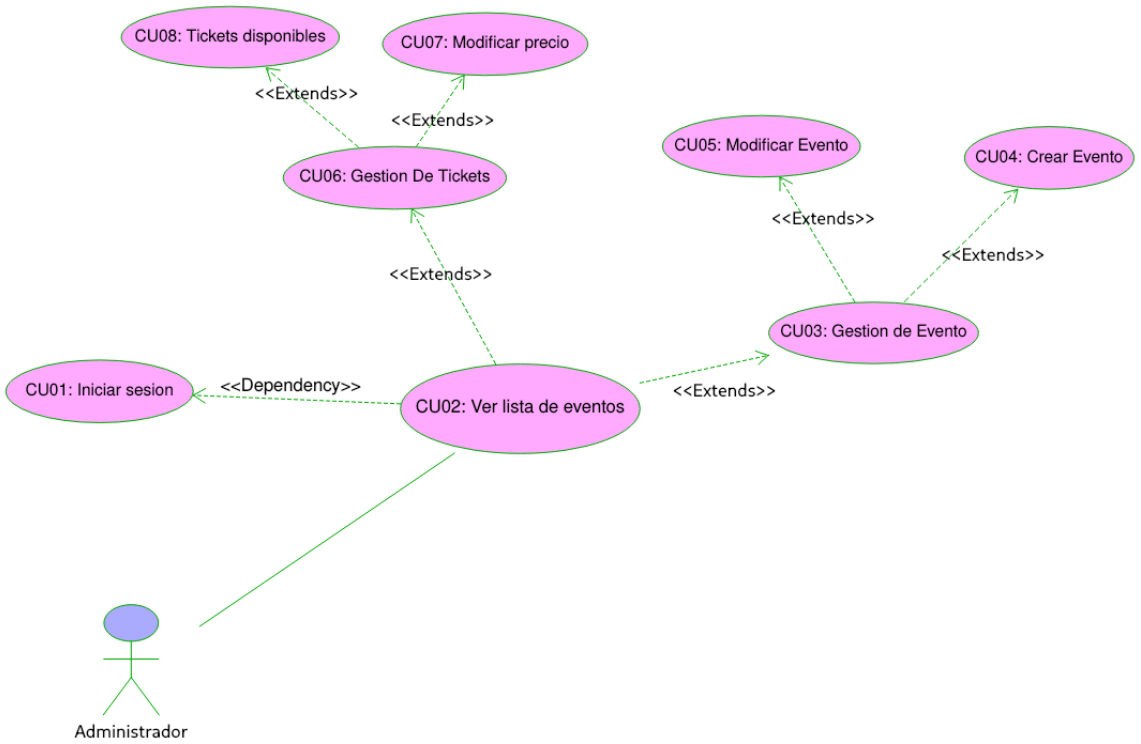
  Permite supervisar la cantidad de entradas disponibles, sus precios y el estado de las ventas por evento.

- **CU07: Modificar precio**

  El administrador ajusta el valor de los tickets según la estrategia de recaudación, los cambios de aforo o la zona en la que se ubican las butacas.

- **CU08: Tickets disponibles**

  El sistema muestra en tiempo real las entradas vendidas y las que quedan disponibles.



5.2 CASOS DE USO USUARIO

- **CU01: Iniciar Sesión**

  El usuario inicia sesión en una cuenta previamente registrada, si dicha cuenta no existe, se le exige el registro.

- **CU02: Registrar**

  El usuario crea una cuenta donde facilita la información que se solicite para el registro y acceso a la plataforma.

- **CU03: Ver Evento**

  El usuario accede a la información completa de un evento, donde se especifica cuánto ha recaudado el evento, si sigue activo o no, el tipo de evento, etc.

- **CU04: Voluntario**

  El usuario puede elegir apoyar a CUDECA presentándose como trabajador voluntario, si el evento visto así lo permite.

- **CU05: Comprar Entradas**

  El usuario adquiere entradas para algún evento visto anteriormente.

- **CU06: Ser Socio**

  Tras asistir a un evento, se le ofrece al usuario la opción de convertirse en socio de CUDECA. CUDECA utiliza la información que facilitó el usuario a la hora de registrar su cuenta para contactarte.

- **CU07: Newsletter**

  Al comprar las entradas, facilitando así su información a CUDECA, se le añade a un Newsletter donde el usuario recibirá notificaciones sobre los nuevos eventos que se creen. El usuario puede darse de baja de este Newsletter cuando así lo quiera.

- **CU08: Donar**

  El usuario no tiene porqué comprar entradas obligatoriamente para apoyar al evento de forma económica, también puede directamente donar a CUDECA para aportar a la recaudación.

- **CU09: Certificado**

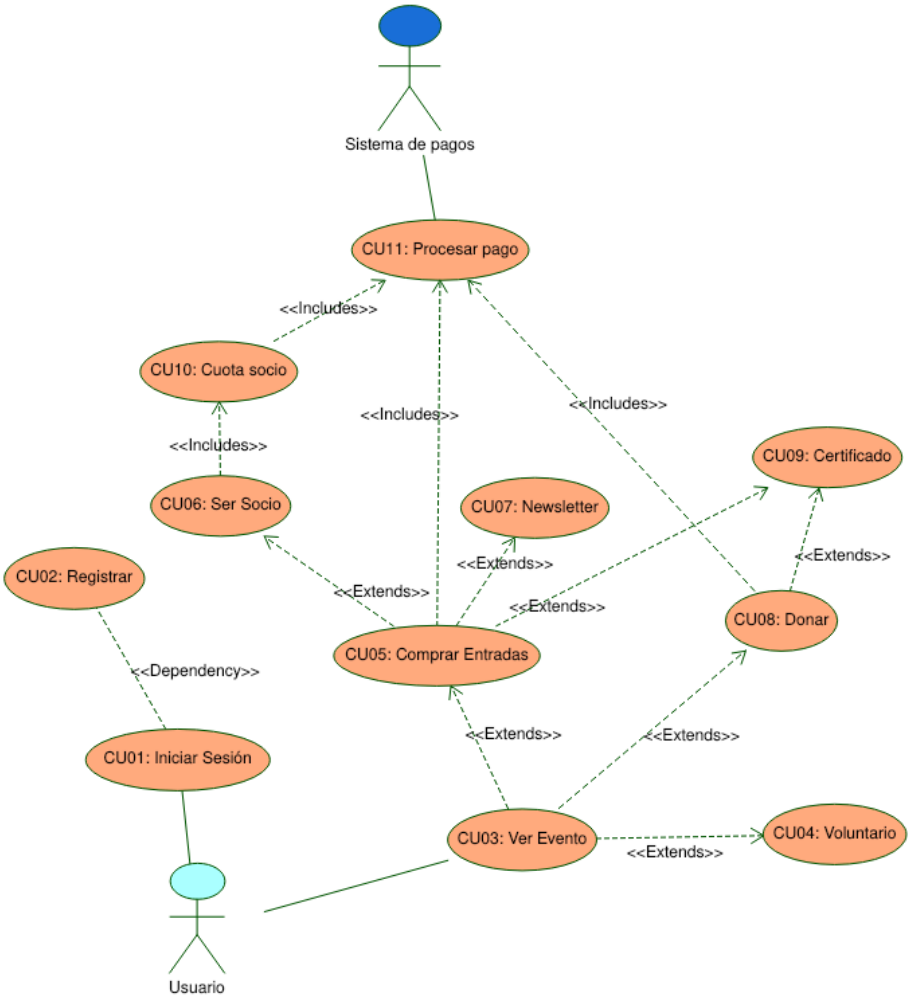
  Un usuario que haya donado o comprado entradas para el evento, es decir, que haya mostrado su apoyo de forma económica, recibe por parte de CUDECA un certificado de donación al evento.

- **CU10: Cuota socio**

  Un usuario que se haya hecho socio, ha de pagar una cuota para seguir manteniendo estos privilegios.

- **CU11: Procesar pago**

  Pagos como el de la donación, la compra de entradas y la cuota de socios han de procesarse en el sistema de pagos correspondiente.



6. DIAGRAMA DE CLASES

6.1 DESCRIPCIÓN

El diagrama de clases modela un sistema integral de gestión de eventos para la fundación CUDECA. En primer lugar, cualquier Usuario puede ser un Socio o un Voluntario, representado mediante relaciones de herencia. Cada Usuario puede realizar múltiples Compra de Entrada, y cada compra puede incluir una o más entradas para diferentes eventos.

Cada Evento está gestionado por un ManejadorEventos que permite al Administrador crear, modificar y cancelar eventos. Las Entrada están compuestas por un CódigoQR único que se utiliza para la validación en el acceso al evento. Cuando un usuario realiza una donación (Donativo), el sistema genera automáticamente un CertificadoDonación que valida fiscalmente la contribución.

Todo Donativo y Compra de entradas se procesa a través del Pagos, que maneja diferentes métodos de pago y estados de transacción. Además, los usuarios pueden recibir Notificación sobre eventos, recordatorios y actualizaciones importantes.

6.2 PAQUETES PRINCIPALES

- **GestionEventos**

  Descripción: Contiene todo lo relacionado con la creación, consulta, modificación y eliminación de eventos. Es el núcleo del dominio, ya que el sistema gira en torno a los eventos organizados. Los casos de uso del administrador (CU02, CU03, CU04, CU05) se centran exclusivamente en la gestión de eventos. Agrupar esta lógica en un solo paquete facilita su mantenimiento y evolución.

  Clases: Evento, ManejadorEventos, Administrador

- **VentasYEntradas**

  Descripción: Se encarga de la lógica de negocio para la venta de entradas, la gestión de precios, la verificación de disponibilidad y la generación de los códigos QR. Separa la complejidad de la transacción de compra (RF-01, CU06, CU07, CU08-Admin, CU05-Usuario) del concepto de "Evento" en sí.

  Clases: Entrada, Compra, AdministracionEntrada, CodigoQR

- **UsuariosYAcceso**

  Descripción: Gestiona todo lo referente a los actores del sistema: registro, inicio de sesión (CU01-Admin, CU01-Usuario, CU02-Usuario), datos personales (RF-02) y roles (Administrador, Usuario). También gestionaría la lógica de "hacerse socio" (RF-03, CU06-Usuario). Centraliza la seguridad y la gestión de la identidad.

  Clases: Usuario, Socio, Voluntario.

- **Contribuciones**

  Descripción: Maneja las funcionalidades que no son ventas directas pero que aportan a la fundación, como las donaciones (RF-08, CU08-Usuario), la gestión de voluntarios (RF-03, CU04-Usuario) y la generación de certificados de donación (RF-05, CU09-Usuario). Las donaciones y el voluntariado tienen reglas de negocio propias y distintas a la compra de una entrada.

  Clases: Donativo, CertificadoDonacion.

- **IntegracionPagos**

  Descripción: Actúa como una capa de abstracción para comunicarse con el sistema de pago externo. Define cómo procesar un pago (RF-07) o un reembolso (RF-06), independientemente del método de pago. El documento especifica claramente que el "Sistema de pago" es un actor externo. Este paquete aísla al resto del sistema de los detalles de implementación de la pasarela de pago, facilitando cambiar o añadir nuevos métodos en el futuro.

  Clases: Pagos

- **Notificaciones**

  Descripción: Centraliza el envío de comunicaciones a los usuarios, como el envío de entradas por correo o WhatsApp (RF-01), avisos de cancelación (RF-06) y la suscripción a la newsletter (RF-03, CU07-Usuario). La forma de notificar puede cambiar (de email a SMS, por ejemplo). Tener un paquete dedicado a ello permite modificar la lógica de comunicación sin alterar los paquetes de ventas, eventos o usuarios.

  Clases: Notificacion

6.3 CLASES Y ATRIBUTOS

- **Usuario**
  - nombre : string
  - apellidos : string
  - dni : string
  - email : string
  - numTelefono : int
  - codigoPostal : int
  - aceptaNewsletter : bool
  - fechaNacimiento : Date

- **Evento**
  - id : int
  - nombre : string
  - tipo : string
  - fecha : date
  - ubicacion : string
  - aforo : int
  - recaudacionActual : double
  - objetivoRecuadacion : double

    - tipoEvento : enum
    - estadoEvento : enum

- **Entrada**
    - id : int
    - zona : int
    - fila : int
    - asiento : int
    - precio : double
    - qrAsignado : CodigoQR
    - evento : Evento
    - comprador : Usuario

- **Donativo**
    - id : int
    - cantidad : double
    - fecha : Date
    - donante : Usuario
    - eventoAsociado : Evento

- **Pago**
    - id : int
    - metodoPago : enum
    - estadoPago : enum

- **Compra**
    - id : int
    - fechaCompra : Date
    - cantidad : double

- **Administrador**

- **Socio**
    - cuotaMensual : int
    - fechaDeAlta : date

- **Voluntario**
    - tareasAsignadas

- **Notificación**
  - id : int
  - mensaje : string
  - fechaEnvio : date
  - eventoNotificado : Evento

- **CodigoQR**
  - id: int
  - code: string

- **CertificadoDonacion**
  - idCertificado : int
  - fecha : Date
  - donacion : donativo
  - codigoValidacion : string

- **ManejadorEventos**
  - eventos: List<Evento>

- **AdministracionEntradas**
  - idAdministracion: int
  - entradas: List<Entrada>
  - evento: Evento

6.4 ENUMERACIONES Y VALUE OBJECTS

6.4.1 ENUMERACIONES

- TipoEvento
  - CONCIERTO, CENA, RIFA, SORTEO, EVENTO_GENERAL.

- EstadoEvento
  - PENDIENTE, ACTIVO, CANCELADO, FINALIZADO.

- MetodoPago
  - PAYPAL, BIZUM, TARJETA.

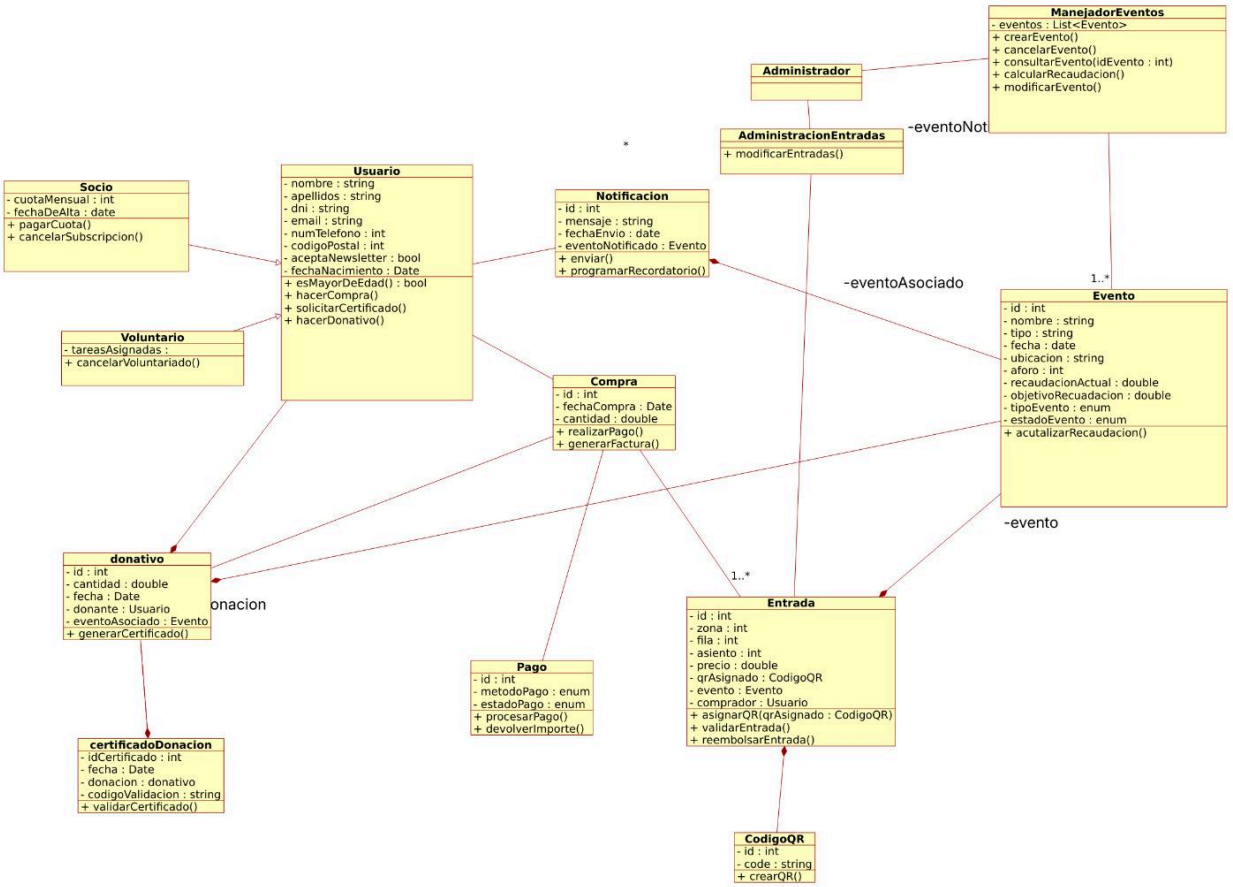
- EstadoPago
  - PENDIENTE, COMPLETADO, FALLIDO, REEMBOLSADO.

6.4.2 VALUE OBJECTS

- Dinero
  - Representa cantidad monetaria, cantidad (long / BigDecimal para evitar problemas de redondeo), divisa (String ; ‘EUR’/’USD’).

- DatosFiscales
  - Info necesaria para emisión certificado, nombreCompleto (String), dni (String), direccionFiscal (String).

6.5 DIAGRAMA UML DE CLASES



6.6 RESTRICCIONES Y REGLAS DE NEGOCIO

- **Unicidad de eventos**
  No pueden existir dos eventos con el mismo nombre, fecha y ubicación.
  El sistema debe validar que no se solapen eventos en la misma ubicación y fecha.

- **Integridad de aforo**
  El número total de entradas vendidas para un evento no puede superar el aforo máximo definido.

- **Validez temporal de eventos**
  No se pueden crear eventos con fecha anterior a la actual.
  Los eventos cancelados no pueden modificarse, solo consultarse.
  Los eventos finalizados permiten solo operaciones de consulta.

- **Generación de QR**

  Cada código QR debe ser único e intransferible y se generan solo después de pago COMPLETADO.

7. DIAGRAMA ENTIDAD-RELACIÓN

7.1 ENTIDADES PRINCIPALES

- *Usuario*: Entidad central que representa a la persona que interactúa con el sistema.
- *Socio*: Tipo de usuario, guarda la información de la cuota mensual que paga el usuario.
- *Voluntario*: Tipo de usuario, guarda la lista de tareas asignadas al usuario.
- *Evento*: Representa los eventos organizados.
- *Entrada*: Modela cada ticket individual vendido para un evento.
- *CodigoQR*: Entidad que almacena el código de validación único para cada entrada.
- *Donacion*: Representa una contribución económica.
- C*ertificadoDonacion:* Almacena los datos del certificado fiscal generado a raíz de una donación.
- *Pago*: Registra la transacción económica.
- *Notificacion*: Modela los mensajes y recordatorios enviados a los usuarios.

7.2 ATRIBUTOS RELEVANTES Y CLAVES

- **Usuario**
  - id (integer) [PK]
  - nombre (varchar)
  - apellidos (varchar)
  - dni (varchar, Unique)
  - email (varchar, Unique)
  - acepta_newsletter (boolean)

- **Socio**
  - id (integer) [PK]
  - user_id (integer) [FK] - referencia a Usuario.id
  - cuota_mensual (decimal)
  - fecha_de_alta (date)

- **Voluntario**
  - id (integer) [PK]
  - user_id (integer) [FK] - referencia a Usuario.id
  - tareas_asignadas (text)

- **Evento**
  - id (integer) [PK]
  - nombre (varchar)
  - descripcion (text)
  - fecha (date)
  - ubicacion (varchar)
  - aforo (integer)
  - precio_entrada (decimal)
  - recaudacion_actual (decimal)
  - objetivo_recaudacion (decimal)
  - estado_evento (enum)

- **Entrada**
  - id (integer) [PK]
  - zona (varchar, Nullable)
  - fila (varchar, Nullable)
  - numero (integer, Nullable)
  - precio (decimal)
  - id_comprador (integer) [FK] - Referencia a Usuario.id (relación comprador)
  - id_evento (integer) [FK] - Referencia a Evento.id (relación evento)
  - id_qr (integer) [FK] - Referencia a CodigoQR.id (relación qrAsignado)

- **CodigoQR**
  - id (integer) [PK]
  - codigoHash (varchar, Unique) - (Atributo inferido, necesario para almacenar el valor del QR)

- **Donacion**
  - id (integer) [PK]
  - cantidad (decimal)
  - fecha_donacion (date)
  - id_donante (integer) [FK] - Referencia a Usuario.id (relación donante)
  - id_evento (integer) [FK] - Referencia a Evento.id (relación eventoAsociado)

- **CertificadoDonacion**
  - id (integer) [PK]
  - fecha_emision (date)
  - codigo_validacion (varchar, Unique)
  - id_donacion (integer, Unique) [FK] - Referencia a Donacion.id (relación donacion)

- **Pago**
  - id (integer) [PK]
  - metodo_pago (enum)
  - estado_pago (enum)
  - id_donacion (integer, Unique) [FK], - Referencia a Donacion.id (basado en la relación 1:1 del diagrama)

- **Notificacion**
  - id (integer) [PK]
  - mensaje (text)
  - fecha_envio (date)
  - id_evento_asociado (integer) [FK] - Referencia a Evento.id (relación eventoNotificado)

7.3 RELACIONES ENTRE ENTIDADES Y MULTIPLICIDADES (EXPLICADAS)

- **Usuario y Entrada**
  - *Relación*: Un Usuario compra (0..N) Entradas. Cada Entrada es comprada por (1) Usuario.
  - *Multiplicidad*: Uno a Muchos (1:N).

- **Evento y Entrada**
  - *Relación*: Un Evento tiene (0..N) Entradas. Cada Entrada pertenece a (1) Evento.
  - *Multiplicidad*: Uno a Muchos (1:N).

- **Entrada y CodigoQR**
  - *Relación*: Una Entrada tiene asignado (1) un CodigoQR. Un CodigoQR válida (1) una Entrada.
  - *Multiplicidad*: Uno a Uno (1:1).

- **Usuario y Donacion**
  - *Relación*: Un Usuario realiza (0..N) Donaciones. Cada Donación es realizada por (1) Usuario.
  - *Multiplicidad*: Uno a Muchos (1:N).

- **Evento y Donacion**
  - *Relación*: Un Evento recibe (0..N) Donaciones. Una Donacion puede estar asociada a (1) un Evento.
  - *Multiplicidad*: Uno a Muchos (1:N) (Opcional en el lado de la Donación).

- **Donacion y CertificadoDonacion**
  - *Relación*: Una Donacion genera (1..1) un CertificadoDonacion. Un CertificadoDonacion certifica (1..1) una Donacion.
  - *Multiplicidad*: Uno a Uno (1:1).

- **Donacion y Pago**
  - *Relación*: Una Donacion se procesa mediante (1..1) un Pago. Un Pago corresponde a (1..1) una Donacion.
  - *Multiplicidad*: Uno a Uno (1:1).

- **Evento y Notificacion**
  - *Relación*: Un Evento es sujeto de (0..N) Notificaciones. Una Notificacion es sobre (1..1) un Evento.
  - *Multiplicidad*: Uno a Muchos (1:N).

- **Usuario y Notificacion**
  - *Relación*: Un Usuario Recibe (0..N) Notificaciones. Una Notificacion es enviada a (1..N) Usuarios.
  - *Multiplicidad*: Muchos a Muchos (N:M). Esta relación requiere una tabla intermedia (ej: Usuario_Notificaciones) con las claves id_usuario (FK) e id_notificacion (FK).

8. (OPCIONAL) DIAGRAMA DE ARQUITECTURA

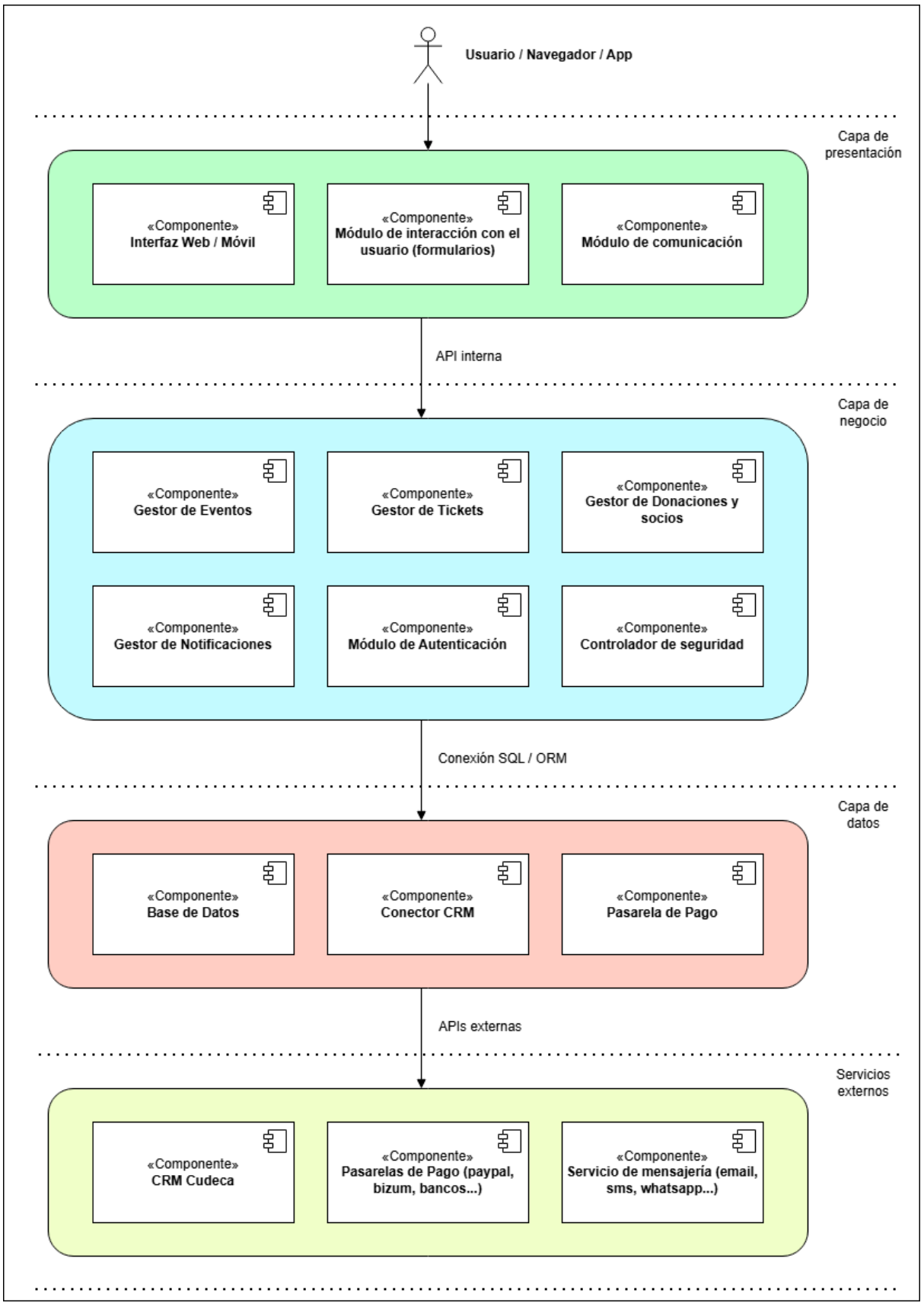
Hemos desarrollado el diagrama de arquitectura teniendo en cuenta las capas lógicas con las que el usuario tendrá que interactuar. Estas representan las posibles interacciones y acciones que puede tener el usuario con nuestro sistema.

La primera capa es la ***capa de presentación***, aquí es donde el usuario realizará las primeras interacciones con el sistema. Como hemos indicado consta de varios módulos los cuales son la representación de los componentes del sistema con los que el usuario podrá tener acceso en este nivel. Para acceder a este nivel no es necesario que el usuario esté registrado, ni que haya iniciado sesión. Se podría resumir en cómo el usuario interactúa con la UI.

En el segundo nivel nos encontramos con la ***capa de negocio***, la cual se accede a través de la API del sistema, por consiguiente el usuario no tiene acceso directo a esta zona del sistema. En esta capa se gestiona toda la parte de la compra/venta de entradas, gestión de eventos, y el control de las notificaciones/mensajes que enviaremos a nuestros clientes. Podríamos resumir esta sección como el núcleo del sistema, donde está el "cerebro".

La tercera parte, la ***capa de datos***, nos resume como el núcleo del sistema interacciona con la parte que gestiona los datos, es decir como nuestro sistema se comunica con la base de datos. Esta comunicación se realiza a través de un ORM de esta forma se jerarquiza y se puede organizar de mejor forma el cómo comunicar ambas partes. El sistema trabaja con la base de datos de forma que se pueda asegurar en todo momento la confidencialidad de los datos. Esta capa actúa como una **abstracción entre la lógica de negocio y las fuentes de datos**, de modo que la capa de negocio no depende directamente de cómo ni dónde se almacenan los datos. Esto reduce el acoplamiento entre capas y facilita el mantenimiento o sustitución de componentes (como la base de datos o conectores externos) sin afectar al funcionamiento general del sistema.

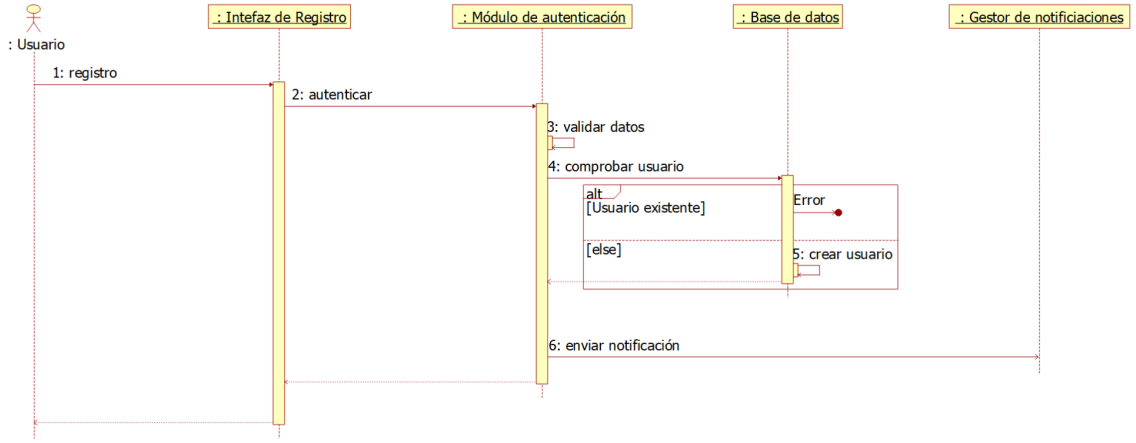
Y por último la parte donde nuestro sistema se comunica con herramientas externas como pueden ser la pasarela de pago, el CRM de cudeca, o realizar petición a un administrador de servicio de mensajería. La ***capa de servicios externos***. Esto tiene lugar con la API de esa herramienta externa, desde la capa de negocio se irán realizando peticiones a las distintas herramientas en caso de que sea necesario.


Usuario / Navegador / App
Capa de presentación
«Componente» Interfaz Web / Móvil
«Componente» Módulo de interacción con el usuario (formularios)
«Componente» Módulo de comunicación
API interna
Capa de negocio
«Componente» Gestor de Eventos
«Componente» Gestor de Tickets
«Componente» Gestor de Donaciones y socios
«Componente» Gestor de Notificaciones
«Componente» Módulo de Autenticación
«Componente» Controlador de seguridad
Conexión SQL / ORM
Capa de datos
«Componente» Base de Datos
«Componente» Conector CRM
«Componente» Pasarela de Pago
APIs externas
Servicios externos
«Componente» CRM Cudeca
«Componente» Pasarelas de Pago (paypal, bizum, bancos...)
«Componente» Servicio de mensajería (email, sms, whatsapp...)

9. (OPCIONAL) DIAGRAMAS DE SECUENCIA

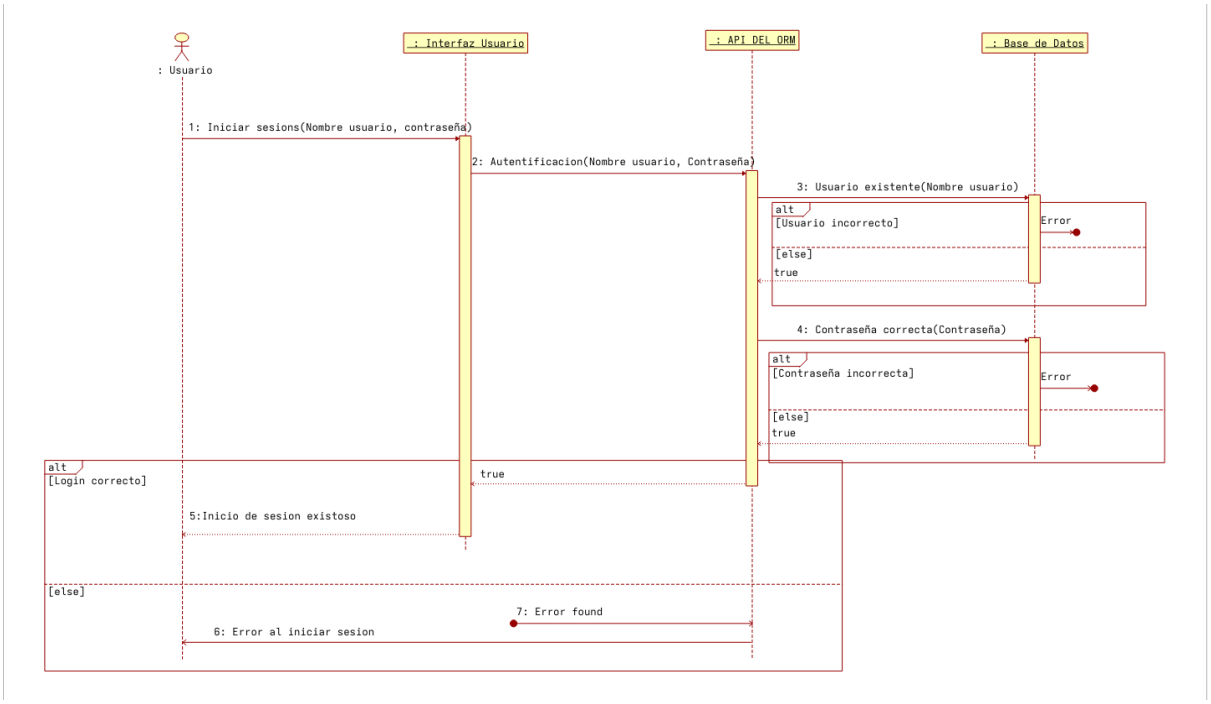
9.1 REGISTRO

Diagrama de secuencia que representa la interacción entre el usuario y el sistema a la hora de registrarse en este.



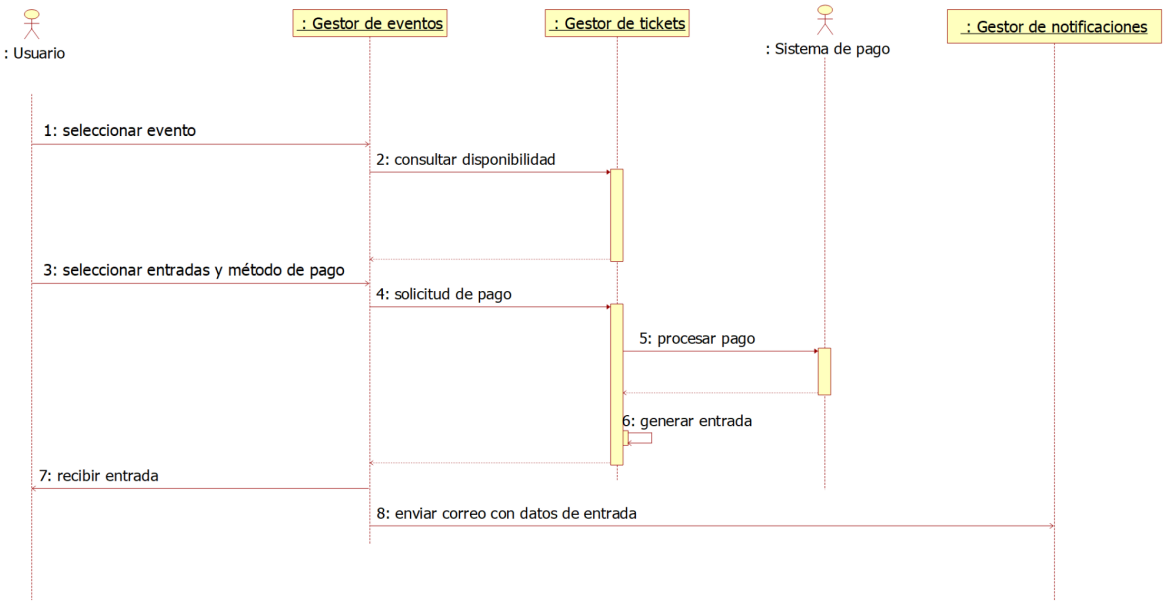
9.2 INICIO DE SESIÓN

Diagrama de secuencia que representa cómo el usuario interactúa con el sistema al intentar iniciar sesión.



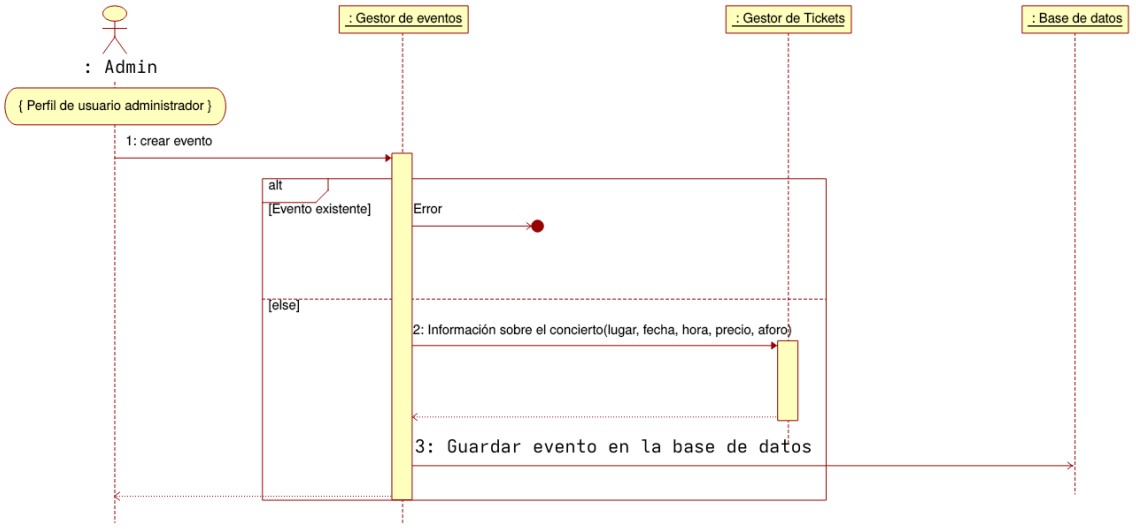
9.3 COMPRA DE ENTRADAS

Diagrama de secuencia que representa la interacción del usuario a la hora de comprar una entrada para un evento.



9.4 CREACIÓN DE EVENTOS

Diagrama de secuencia que representa como un usuario, que ha de tener un perfil de tipo administrador, genera un nuevo evento.



10. PROBLEMAS DURANTE EL PROCESO DE TRABAJO

1. Problemas con Trello, herramienta que usamos para poder establecer plazos para tener las diferentes entregas listas y para poder dividirlas entre los diferentes miembros del grupo. El problema radica en que no pueden estar más de 10 personas en el mismo tablero, ya que se pone en modo lectura por lo que no se puede modificar nada. Hemos considerado como alternativa el programa Superthread.

2. Problemas con umbrello, crasheos en linux, en windows, se borran partes del diagrama como líneas. Incompatibilidad entre sistemas y versiones. Un claro ejemplo del mal análisis y diseño de aplicaciones por parte de KDE.

3. Diferencias de dificultad entre la práctica y el trabajo del método del caso. La simplicidad del proyecto conlleva al sobre pensamiento de las diferentes partes/secciones.