Paquetes y sus clases:

Los nombres en negrita se refieren a los paquetes. El siguiente párrafo contiene contexto acerca de las clases y los elementos de las clases que hemos encontrado pertenecen al paquete. Aquí es donde estaría todo el html, el código que se encarga de hacer visible la página en sí.

- **Gestión de usuarios:** Gestionar la autenticación, las preferencias y el registro de usuarios. Cuando una persona entre por aquí desde la web base, debería salir en el paquete que le corresponde, clasificado correspondientemente. Ya sean personal, clientes normales o socios, etc.
    - Usuario (abstracta, el resto heredan de ella)
        - Administrador
        - Voluntario
        - Cliente

- **Gestión de pagos:** Todo lo relacionado con el dinero. Tenerlo aislado del resto del sistema ayudará a tener mayor seguridad. Aparte, también parece importante que concretamente esto no tenga repetidos. Es un área crítica. Separarla de las demás a nivel de arquitectura es lo prudente.
    - Clase recibo (atributos de los datos fiscales)
        - Clase compraEntrada (atributos de los datos fiscales, atributo para referirse a la entrada comprada)
        - Clase compraRifa(atributos de los datos fiscales, atributo para referirse al boleto comprado)
        - Clase donación (atributos de los datos fiscales)
    - Clase Certificado (Atributo de identificación , atributos de identificación de usuario, Importe donación)

- **Gestión de comunicaciones:** Se encarga de todos los mensajes del sistema. Envío de certificados de donaciones, recordatorios de eventos, etc.
    - Clase Mensaje (Contiene atributo de contenido y destinatarios (puede ser uno o n))

Necesitaríamos un atributo en la clase usuario que se encargue de registrar el método de preferencia de comunicación. Y un atributo de “si está suscrito” al newsletter, para enviarlo o no. También se requiere acceso a las clases certificado de donación y recibo, para enviarlas a los usuarios. No generarlas, eso lo hace el paquete de gestión de pagos, solo manipularlas.

- **Administración:** Se encarga de crear eventos, modificarlos, ver los que hay y validar entradas. Todo en función del tipo de usuario que esté accediendo.
    - Clase evento (atributo fecha y atributo nombre)
    - Clase rifa (atributo fecha de fin y atributo nombre)
    - Clase boleto (atributo identificador, atributo refiriendo la rifa correspondiente)

- Clase entrada (atributo identificador, atributo refiriendo la el evento correspondiente, atributo qr)

Necesitamos una clase evento y una clase rifa que no herede de esta (más bien que actúe de manera independiente y, llegase a ser necesario, como complemento). También necesitamos una clase entrada con un atributo qr y un bool de asistencia. Una clase boleto para las rifas, independiente de entrada. Son cosas similares pero diferentes, mejor no heredar por aquí.

-**Conclusión a la que hemos llegado en ambigüedad de paquetes**
Se había planteado un paquete llamado gestión de clientes cuya función era la de encargarse de mostrar todo lo relacionado con el cliente. Después de pensarlo hemos llegado a la conclusión de que es mejor idea implementar el paquete Front-end. Ya que de esta forma se puede gestionar la entrada de usuarios a la web sin necesidad de estar registrados. De igual forma es un ligero cambio acompañado de un cambio de nombre pero creemos que simplifica las cosas. De esta forma tenemos en un solo paquete todo lo relevante la visualización de las cosas. El HTML, la cara de la web.

Enumeración + Value Objects

Usuario

Atributos

- Nombre de usuario. String private
- Correo electrónico. String private
- Contraseña_hash. String private
- Nacimiento Date private
- SuscritoNewsletter boolean private
- NIF String (NIF/NIE)(opcional) private
- MetodoComunicacionPref String protected
- Mensajes Gestion de comunicaciones::Mensaje[] protected

Métodos

getter de todos los datos a excepción de contraseña.
verificarContraseñaHash(constraseña_hash: String) Boolean public
verificarContraseña(constraseña: String) Boolean public

Cliente extends usuario

Atributos

- EsSocio boolean private
- EntradasEvento ReciboEntrada[] private
- RecibosRifa ReciboRifa[] private
- ReciboDonación ReciboDonacion[] private

Administrador extends usuario

Atributos

- AdminRole (grado de privilegios) enum private

Voluntario extends usuario

Métodos

- escanearQR(Imagen) boolean private

Recibo

Atributos

- Fecha date private
- Precio Int private
- identificador String private

ReciboEvento extends Recibo

Atributos

- Comprador enum(datos) private

ReciboRifa extends Recibo

Atributos

- Ganador enum(datos) private

ReciboDonación extends Recibo

Atributos

- Donador enum(datos) private

Certificado (para posibilitar la emisión del certificado el usuario debe de haber especificado su NIF o NIE)

Atributos

- persona Usuario private
- donacion Donacion private

Mensaje

Atributos

- contenido String private
- destinatarios Strings [] mínima private
- emisor Usuario private

Evento

Atributos

- id Int private
- nombre String private
- localización tupla(longitud float, latitud float) private
- Fecha Date private
- información String private
- entrada Entrada private

Rifa

Atributos

- id Int private
- ganador Int private
- nombre String private
- Fecha de fin Date private
- información String private

Boleto

Atributos

- idBoleto String private
- idRifa String private

Entrada

Atributos

- id Int private
- Precio Float private
- Asiento String puede ser nulo private
- EnlaceEvento String private
- Código qr PNG private

Creemos que tenemos bien gestionados los atributos mediante herencia. Por eso, creemos que no hay necesidad del uso de value objects.

Restricciones y Reglas de Negocio

1. Seguridad y Arquitectura de Alto Nivel

Estas reglas definen la base sobre la que se construye el sistema y no son negociables.

Comunicaciones Seguras Obligatorias (HTTPS)

Regla: Toda la funcionalidad web, sin excepción, debe operar sobre el protocolo HTTPS.

Aislamiento Crítico del Paquete de Pagos

Regla: El paquete Gestión de pagos debe estar arquitectónicamente aislado del resto del sistema.

2. Integridad de Datos y Restricciones del Modelo

Estas reglas aseguran que los datos almacenados sean coherentes, correctos y únicos.

Prohibición de Duplicados Financieros

Detalle: Esta regla de negocio se traduce en restricciones técnicas. El Identificador: String de la clase Recibo debe ser una clave única en la base de datos. Lo mismo aplica para los identificadores de CompraEntrada , CompraRifa , Donacion y el "Atributo de identificación" de Certificado. Esto es vital para la contabilidad, evitar dobles cobros y garantizar la validez de los certificados de donación.

Regla: No pueden existir registros repetidos dentro del paquete Gestión de pagos.

Independencia Estructural (No Herencia)

Regla: Las clases Evento y Rifa deben ser independientes y no heredar una de la otra. Lo mismo aplica para Entrada y Boleto.

3. Lógica de Negocio y Flujos de Proceso

Estas reglas definen cómo debe operar el sistema en escenarios específicos.

Proceso de Clasificación de Usuarios

Regla: Un usuario nuevo (abstracto) debe ser clasificado en un tipo concreto (Cliente, Voluntario, Administrador).

Gestión de Preferencias de Comunicación

Regla: El sistema debe consultar y respetar las preferencias de comunicación del usuario antes de enviar cualquier mensaje.

Proceso de Validación de Asistencia (Check-in)

Regla: Debe existir un proceso para marcar la asistencia de una Entrada a un Evento.

El usuario presenta su Entrada (con el QR).

Un usuario autorizado (probablemente Administrador o Voluntario) escanea el QR.

El sistema ejecuta la acción válida, que cambia el atributo asistencia de false a true.

4. Autorización y Separación de Responsabilidades

Estas reglas definen quién puede hacer qué y dónde.

Permisos Granulares Basados en Rol

Regla: Las acciones dentro del paquete Administración están estrictamente controladas en función del tipo de usuario que esté accediendo.

Permisos: El diagrama muestra que Administrador tiene una relación de gestión con Rifa y Evento. Esto implica permisos totales (Crear, Modificar, Eliminar). Sin embargo, un Voluntario no tiene esta relación. Su permiso podría estar limitado solo a la acción de validar una Entrada, pero no a crear o modificar el Evento en sí. El sistema debe implementar esta lógica de autorización detallada.

Separación de Responsabilidades (Pagos vs. Comms)

Regla: El paquete Gestión de pagos crea registros financieros, y Gestión de comunicaciones solo los lee para enviarlos.

Pagos (Creador): Es el único responsable de generar Certificado y Recibo.
Comunicaciones (Lector/Manipulador): Necesita acceso a esas clases para enviarlas. Gestión de comunicaciones tendrá permisos de solo lectura sobre las tablas de Certificado y Recibo, impidiendo que pueda crear o alterar registros financieros, lo cual es responsabilidad exclusiva de Gestión de pagos.

5. Getters y setters para atributos privados

Ya que no los vamos a poner en el diagrama por cuestiones de limpieza y simplicidad, queríamos aclarar que todos los atributos privados cuentan con getters y setters.

6. Vistas de la aplicación

Las vistas a continuación se refieren a lo que ven los usuarios antes y después de identificarse. Partiremos con los eventos y rifas que hayan. Para toda la funcionalidad web, usamos https para que la web sea segura. Acceso a inicio de sesión y registro.

- Vista base (Tiene Header, Navbar, footer…)
- Vista de pago
- Vista de donación
- Vista de ver eventos/rifas
- Vista administrar eventos/rifas
- Vista formulario de usuario log in
- Vista formulario de usuario registro
- Vista perfil de usuario
- …