DOCUMENTACIÓN
PROYECTO FIN DE CURSO

1º DAM

Jesús Ceacero Jimeno
Cads918@gmail.com

Parte 1: Descripción de la APP. Que hace y que gestióna.

Mi aplicación gestiona un club cervecero. Esta gestionara el alta de los usuarios, los eventos relacionados con el club (incluidos sus costes) y el listado de las cervezas disponibles en el club.

Todos los usuarios podrán acceder a la página principal de la app y al listado de cervezas y eventos. Pero solo los usuarios registrados podrán reservar los eventos.

Añadir, modificar y eliminar cervezas y eventos solo los podrán realizar los administradores. Al igual que el alta y la baja de usuarios.

Parte 2: Toma de requisitos.

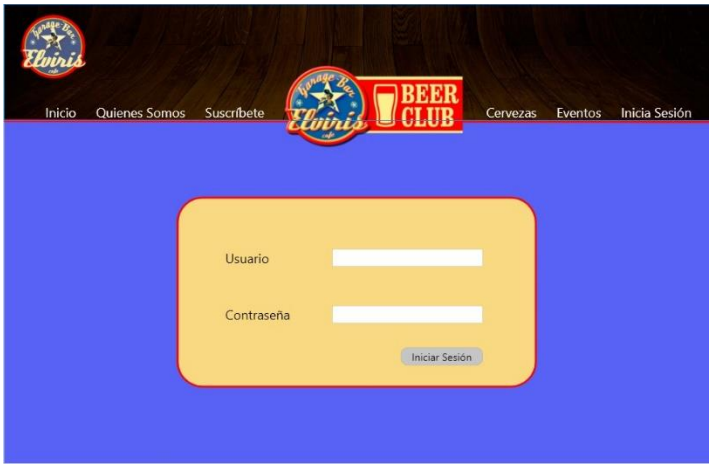
1. Añadir usuarios (administradores).
2. Eliminar usuarios (administradores).
3. Modificar usuarios (todos los usuarios registrados).
4. Buscar usuarios (administradores).
5. Añadir cervezas (administradores).
6. Eliminar cerezas (administradores).
7. Modificar cervezas (administradores).
8. Buscar cervezas (todos los usuarios registrados).
9. Añadir eventos (administradores).
10. Eliminar eventos (administradores).
11. Modificar eventos (administradores).
12. Buscar eventos (todos los usuarios registrados).
13. Reservar eventos (todos los usuarios registrados).
14. Navegar por la aplicación (todos los usuarios).
15. Calculo de contabilidad (administrador).

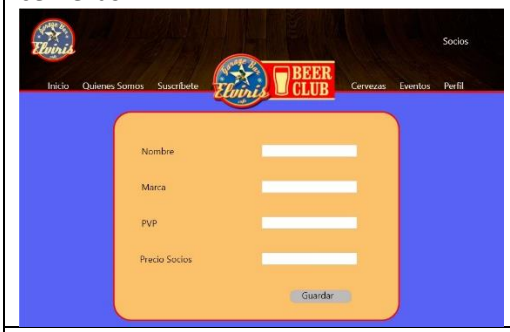
Parte 3: Sketching -> Pintear la parte gráfica.

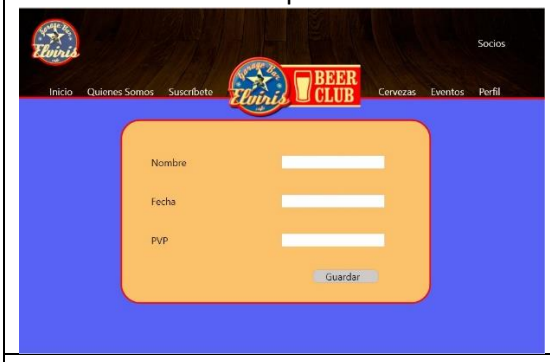
Al iniciar nuestra aplicación saldrá una página principal para todos los usuarios.

Sin iniciar sesión los usuarios podrán rellenar un formulario para mandar la petición de subscripción.

Todos los socios podrán iniciar sesión en la aplicación, ya sea clientes o administradores.



A partir de aquí mostrare las distintas vistas de la aplicación dependiendo si son de administrador o socio.	
Administrador	Socio
El apartado de cervezas, será un listado de las mismas, todas las que estén en la base de datos junto su foto y datos. El administrador podrá borrar, añadir y modificar las cervezas.	
	
Esta será la vista del formulario para añadir cervezas. 	
El apartado de eventos será un listado de las mismos, todos los que estén en la base de datos. El administrador podrá borrar, añadir y modificar los eventos. Todos los usuarios registrados podrán reservar los eventos.	

<table>
<tr><td> 
</td><td> 
</td></tr>
<tr><td>Así sería el formulario para añadir eventos.
 
</td><td></td></tr>
<tr><td colspan="2">El carrito de reservas de eventos tendrá la misma vista para todos los usuarios registrados.</td></tr>
<tr><td></td><td> 
</td></tr>
<tr><td>El listado de los socios solo será visible para los administradores.</td><td></td></tr>
<tr><td> 
</td><td></td></tr>
<tr><td>Este sería el formulario para añadir socios.</td><td></td></tr>
</table>

Parte 4: Requisitos Clases Modelo.

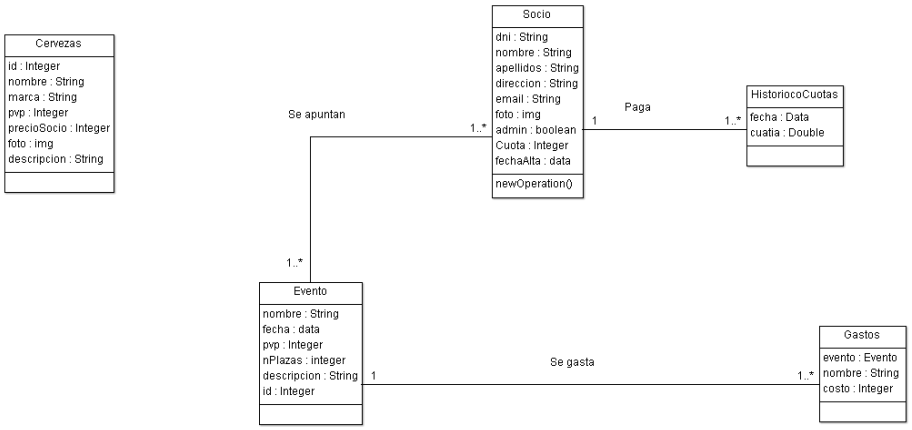
Socio (DNI, nombre, apellidos, dirección, email, foto, permisos)

Eventos (nombre, fecha, PVP, nPlazas, descripcion)

Cervezas (ID, nombre, marca, PVP,password, precioSocio, descripcion)

ReservasEventos (nombreEvento, fecha, DNI, nombreSocio)

Parte 5: Diagrama de clases.



Parte 6: Casos de prueba.

CP-E4-01	Test_buscarPorNombre()
Objetivo	Este test comprueba el método buscarPorNombre de la clase EventoServicio, creamos un nuevo evento con el nombre “Test", la guardamos en la bbdd y comprobamos llamando al método si la encuentra pasándole el nombre.
Datos de Entrada	Evento e = new Evento (“Test”,LocalDate.of(2018,5,24),15.00,25,”Cata de cervezas artesanales con catering”);
Condiciones previas	
Procedimiento	eventoServicio.save(e);
Resultados Esperados	Si le pasamos el nombre correcto debe devolver el evento creado.
Validado	

CP-E4-01	Test_totalGastos()
Objetivo	Este test comprueba el método totalGastos de la clase GastoServicio, creamos un listado con todos los gastos de la base de datos. En un bucle los sumamos y llamamos al método para comprobar que dan el mismo resultado.
Datos de Entrada	List<Gasto> listado = gastoServicio.findAll();
Condiciones previas	
Procedimiento	assertEquals(total,gastoServicio.totalGastos(),0.001);
Resultados Esperados	Debe de dar el mismo total de gastos.
Validado	

CP-E4-01	Test_totalCuotas()
Objetivo	Este test comprueba el método totalCuotas de la clase ingresoCuotaServicio, creamos un listado con todas las cuotas de la base de datos. En un bucle los sumamos y llamamos al método para comprobar que dan el mismo resultado.
Datos de Entrada	List<ingresoCuota> listado = ingresoCuotaServicio.findAll();
Condiciones previas	
Procedimiento	assertEquals(total,ingresoCuotaServicio.totalGastos(),0.001);
Resultados Esperados	Debe de dar el mismo total de ingresos por cuotas.
Validado	

CP-E4-01	Test_buscarPorEmail()
Objetivo	Este test comprueba el método buscarPorEmail de la clase SocioServicio, creamos un nuevo evento con el email "adminnnnn@admin.com", la guardamos en la bbdd y comprobamos llamando al método si la encuentra pasándole el email.
Datos de Entrada	Socio s= new Socio ("23963211F","Agel","Naranjo González", "adminnnnn@admin.com", true , 20.00, LocalDate.of(2019,2,2), "admin");
Condiciones previas	
Procedimiento	socioServicio.save(e);
Resultados Esperados	Si le pasamos el email correcto debe devolver el socio creado.
Validado	

CP-E4-01	Test_buscarPorNombre()
Objetivo	Este test comprueba el método buscarPorEmail de la clase SocioServicio, creamos un nuevo evento con el nombre “alfredo”, la guardamos en la bbdd y comprobamos llamando al método si la encuentra pasándole el email.
Datos de Entrada	Socio s= new Socio (“23963211F”,”alfredo”,”aaaaaaaaaa”, “alfredo@admin.com”, true , 20.00, LocalDate.of(2019,2,2), “admin”);
Condiciones previas	
Procedimiento	socioServicio.save(e);
Resultados Esperados	Si le pasamos el nombre correcto debe devolver el socio creado.
Validado	