

# PROYECTO DRITTSEKK



Diseñado y hecho por Pablo, Benjamín y Chari.

# INDICE

- Contenido
  - Resumen
  - Contextualización
  - Objetivos
    - Requisitos funcionales / no funcionales
    - Prototipo
    - Casos de uso
    - Diagrama de clases de análisis
    - Diagrama entidad Relación

# Contenido

## Resumen

Nuestro proyecto se trata de una red social llamada DrittSekk pensada para que los usuarios se molesten entre ellos, siempre desde el humor. No queríamos que fuera simplemente otra red social como cualquier otra, sino un espacio donde la incomodidad sea un juego compartido.

La idea surge entre bromas y la gracia de construir amistades a base de chincharse (con cariño), que, al fin y al cabo, suelen ser las más reales y duraderas.

DrittSekk ofrece funciones como elementos irritantes, menús confusos, notificaciones falsas y un largo etcétera de herramientas para desesperar al usuario. Todo acompañado por algunas medidas básicas de seguridad para que la experiencia no se nos vaya de las manos.

El resultado es un espacio original que mezcla lo absurdo con lo funcional, dirigido a un público que comparta nuestro sentido del humor, incluso llegando a reírse de uno mismo.

## Contextualización

La semilla de nuestra idea comienza a desarrollarse mucho antes de tener conocimiento sobre este proyecto, o incluso sobre el desarrollo del módulo de Entornos de Desarrollo. Tras los primeros días de clase comenzamos a forjar amistad entre los integrantes de este grupo y de ahí nace nuestra idea de molestar al usuario. Nos dimos cuenta de que las amistades son mucho más divertidas cuando puedes chincar, molestar o hacer de rabiar a aquel compañero que te acompaña durante el trayecto del curso. Ahí nació la idea base de nuestra aplicación, queríamos desarrollar algo que molestase al usuario. Avanzando ya en el transcurso del año, nos topamos con la incógnita de no saber cómo íbamos a emplearla, y con la ayuda de aplicaciones populares que usamos a diario y fuentes de inspiración como la serie El Consultor, llegamos a la conclusión de que molestar al usuario no era suficiente, necesitábamos convertirlo en una actividad social. Y nació el proyecto DrittSekk, una red social basada en irritar a los usuarios, permitiendo que los usuarios también se fastidien entre ellos.

## Objetivos

El objetivo principal de nuestro proyecto es desarrollar una red social que transforme la incomodidad en una experiencia compartida, divertida y original.

DrittSekk busca una pequeña innovación en el mundo de las redes sociales al convertir la molestia en una herramienta de socialización, con el humor como protagonista.

Queremos diseñar y construir una red social funcional que permita la interacción entre usuarios a través de elementos intencionadamente irritantes, como formularios inservibles, calendarios desactualizados, notificaciones falsas, audios aleatorios, giros de pantalla inesperados o cierres automáticos, entre otros.

Nuestra prioridad es garantizar la seguridad del usuario, por lo que incluimos medidas como la verificación de identidad al registrarse, asegurándonos de que se trata de personas reales y reduciendo así el riesgo de un uso inapropiado de la plataforma.

Queremos generar una incomodidad creativa y controlada, sin comprometer la estabilidad ni el correcto funcionamiento interno del sistema.

## Expansión de nuestro trabajo:

### 1.Requisitos funcionales / no funcionales.

Los requisitos funcionales de la web incluyen: una página de inicio de sesión; un menú del tipo hamburguesa disfuncional; una barra de búsqueda de usuarios; una bandeja de entrada de notificaciones que emita notificaciones falsas; un calendario disfuncional; las herramientas que se le dan al usuario para molestar a otros; los distintos tipos de chats; los ajustes del usuario y la opción de bloquear o reportar a otros usuarios. Como requisitos no funcionales hemos listado: una base de datos para guardar los datos del usuario, un JavaScript para cerrar la web automáticamente, un JavaScript para hacer un giro de la pantalla de 180°, una base de datos para el calendario, un JavaScript para invertir el scroll del ratón, un JavaScript para el formulario que modifique la base de datos de los usuarios, una base de datos para los chats, un JavaScript para el formulario de bloqueo o reporte de usuarios, un JavaScript para el menú, una base de datos más un JavaScript para la interacción de la barra de búsqueda y los usuarios, un JavaScript para crear la cola de espera falsa, un JavaScript para el correo falso que nunca llegará, un JavaScript para la bandeja de entrada que esté relacionado con la base de datos de los chats, y por último todos los recursos que requiera darle los permisos adecuados a los usuarios administrados, para poder gestionar las notificaciones de bloqueo o reporte y acceder a los registros de los chats.

### 2.Prototipo

El prototipo que hemos realizado muestra un ejemplo muy básico de cómo tenemos pensado que funcione nuestra “red social”. Al principio hicimos un archivo .PNG para guardar un boceto de cómo queríamos que sea la página visualmente en cuanto la

estructura base:



Luego de hacer esta imagen, continuamos haciendo un archivo HTML para crear la base de nuestra “red social”. Este es nuestro prototipo en **HTML** ya finalizado:

Tiene todos los apartados que queríamos hacer como el menú de arriba, el calendario, la bandeja de entrada, los chats y la búsqueda de usuarios. Mediante CSS hemos logrado hacer que se vea como el boceto. Todos los apartados que tenemos utilizan JS de algún modo. Como se puede ver en el <head> de nuestro HTML, tenemos 5 scripts; Uno para el menú, otro para el calendario, otro para la búsqueda de los usuarios, otro para las solicitudes que se enviarán entre los usuarios y finalmente uno para “almacenar” los chats.

El fichero **CSS** que hemos creado para esta página no lo voy a mostrar como imagen ya que son unas 150 líneas y ocuparía todo el documento. Mediante posicionamiento y alineaciones y el uso de los ID del HTML hemos centrado (y descentrado a posta para que diese TOC) los elementos necesarios en cada apartado.

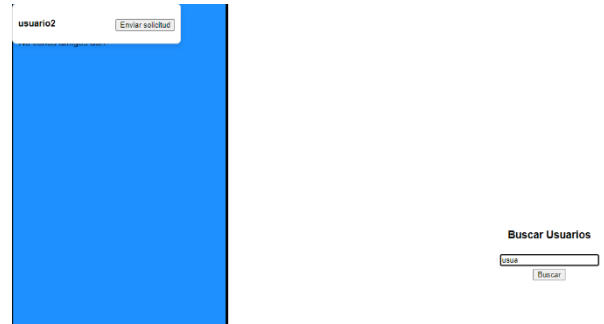
```

1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4    <meta charset="UTF-8">
5    <title>Proyecto DriftSekK</title>
6    <link rel="stylesheet" href="prototipo.css">
7    <script src="MenuBurguer.js"></script>
8    <script src="Calendario.js"></script>
9    <script src="BuscarUsuarios.js" defer></script>
10   <script src="SolicitudesAmistad.js" defer></script>
11   <script src="Chats.js" defer></script>
12 </head>
13 <body>
14   <div id="MenuBurguer">Menu burguer</div>
15   <div id="Contenido">
16     <div id="Chats">Chats</div>
17     <div id="BusquedaUsuarios">
18       <h3>Buscar Usuarios</h3>
19       <input type="text" id="inputBusqueda" placeholder="Buscar...">
20       <button onclick="buscarUsuario()">Buscar</button>
21       <div id="resultadosBusqueda" class="ventana-sugerencias"></div>
22     </div>
23     <div id="MenuDerecha">
24       <div id="Notificaciones">
25         <h3>Bandeja de entrada</h3>
26         <div id="contenedorNotificaciones"></div>
27       </div>
28       <div id="Calendario">
29         <h3>Calendario</h3>
30         <div id="calendario-contenedor"></div>
31       </div>
32     </div>
33   </div>
34 </body>
35 </html>

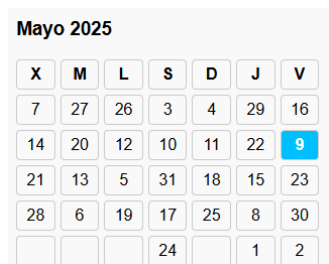
```

El **JavaScript** se divide en 5 (6) ficheros (explicaremos lo que hace cada uno, pero no mostraré fotos ya que ocuparían nuevamente todo el documento). Cada uno para un apartado de la web:

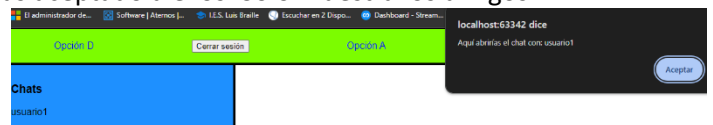
**BuscarUsuarios** este fichero identifica mediante un `<input>` del usuario qué usuarios existen en el sistema (Cómo no tenemos base de datos para nuestro prototipo todos los nombres de los usuarios se almacenan en una variable local.) Y muestra las coincidencias posibles. Este también te permite enviar solicitud a dicho usuario y le llegará la notificación en su bandeja de entrada.



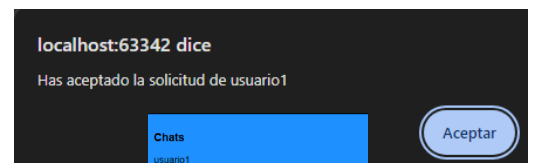
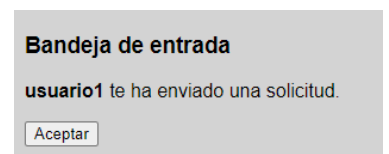
**Calendario** este fichero crea un calendario disfuncional; Muestra el mes y año actual, pero en un caos total: No te dice qué día es hoy (escoge uno aleatorio y lo selecciona como si fuese el actual), Cada vez que refrescas la página, el orden de los días está aleatorizado y la posición de números de este también (La posición de los días sigue siendo la misma; Si el Día 1 de este mes fue jueves, el 1 sigue siendo jueves).



**Chats** este fichero muestra los chats disponibles en el apartado de chats en nuestro HTML. Hace uso de el almacenamiento local del navegador para guardar y leer los datos de los amigos que te han aceptado la solicitud o les has aceptado a ellos. Sólo muestra los amigos disponibles para chatear y te muestra una alerta para simular que te aparecería el chat de dicho usuario.



**SolicitudesAmistad** este fichero envía o acepta solicitudes de amistad entre usuarios, utilizando el almacenamiento local del navegador envía una solicitud a `<usuario>` y se lo envía a su bandeja de entrada. Las notificaciones las muestra escalonadamente mediante el orden de envío en el apartado de notificaciones del HTML.



**MenuBurguer** este fichero es el que muestra el menú de arriba, de momento solo tenemos enlaces hacia el login de los usuarios; pero aquí estarían los enlaces a todo tipo de perrerías y maneras de molestar a tus amigos. Este menú se genera de manera aleatoria cada vez que se refresca la página. Dependiendo de si estás logueado o no, te muestra un mensaje de bienvenida y un “cerrar sesión” o un “Login”.

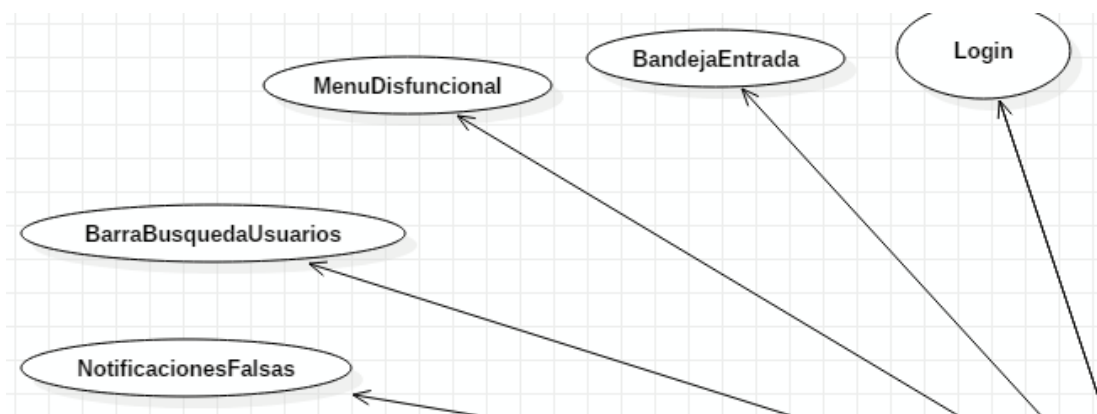


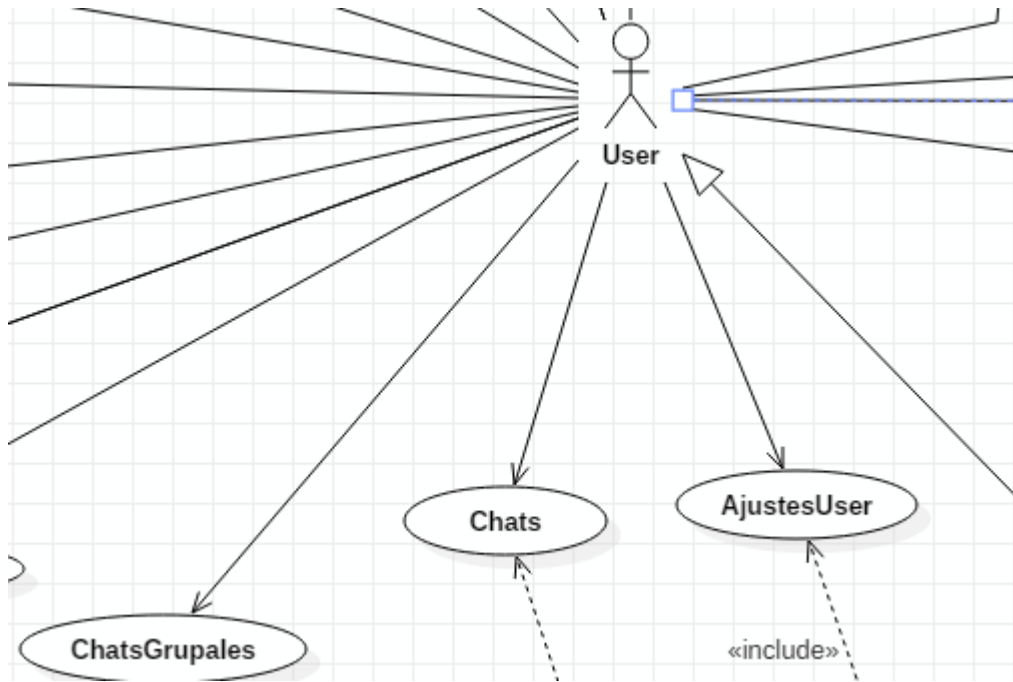
Open Y	Open D	Open Z	Login	Open B	Open C	Open A
Open A	Open D	Open Y	Open Z	Open B	Open C	Login

**Login.js y página de Login** estos ficheros están apartados en nuestro proyecto ya que no tienen mucho que ver con la idea del proyecto, son solo necesarios; estos ficheros sirven para almacenar localmente (ya que no disponemos de base de datos) los usuarios permitidos para hacer login (actualmente hay 2 usuarios y un administrador, cada uno tiene su usuario y contraseña. No pueden buscar como amigo al admin; el admin sí puede enviarles solicitud de amistad pero los usuarios no)

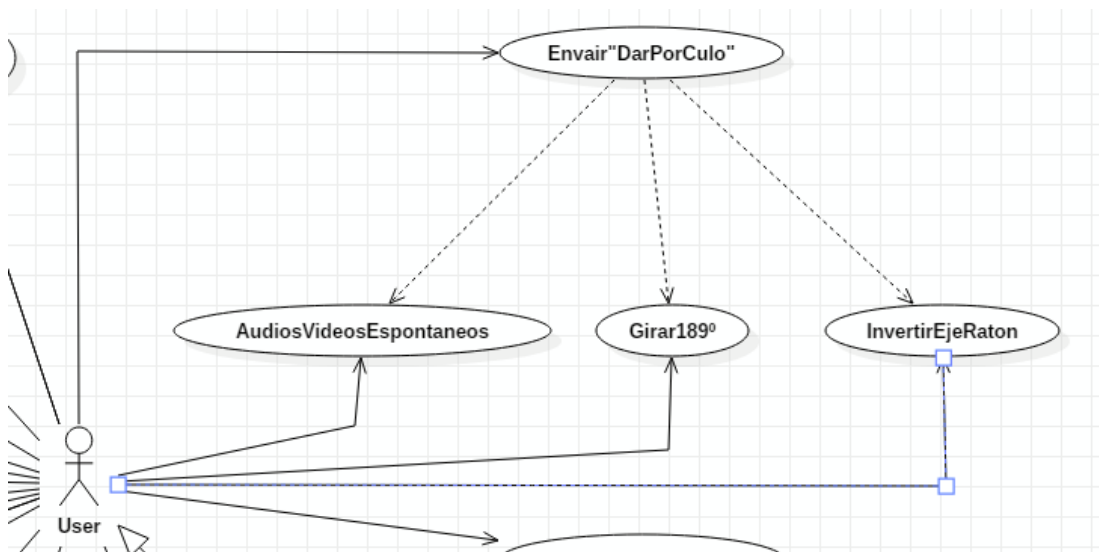
### 3.Casos de uso

Usando StarUML como herramienta, hemos hecho diversos esquemas para ayudar en la construcción de nuestro proyecto. El diagrama de casos de uso se compone de posibles acciones e interacciones entre el usuario y la web. Tenemos una primera pestaña de login, seguido de nuestra pestaña principal. En esta ventana principal tenemos los chats del usuario, la barra de búsqueda de otros usuarios, una bandeja de entrada, un calendario y los ajustes del usuario.



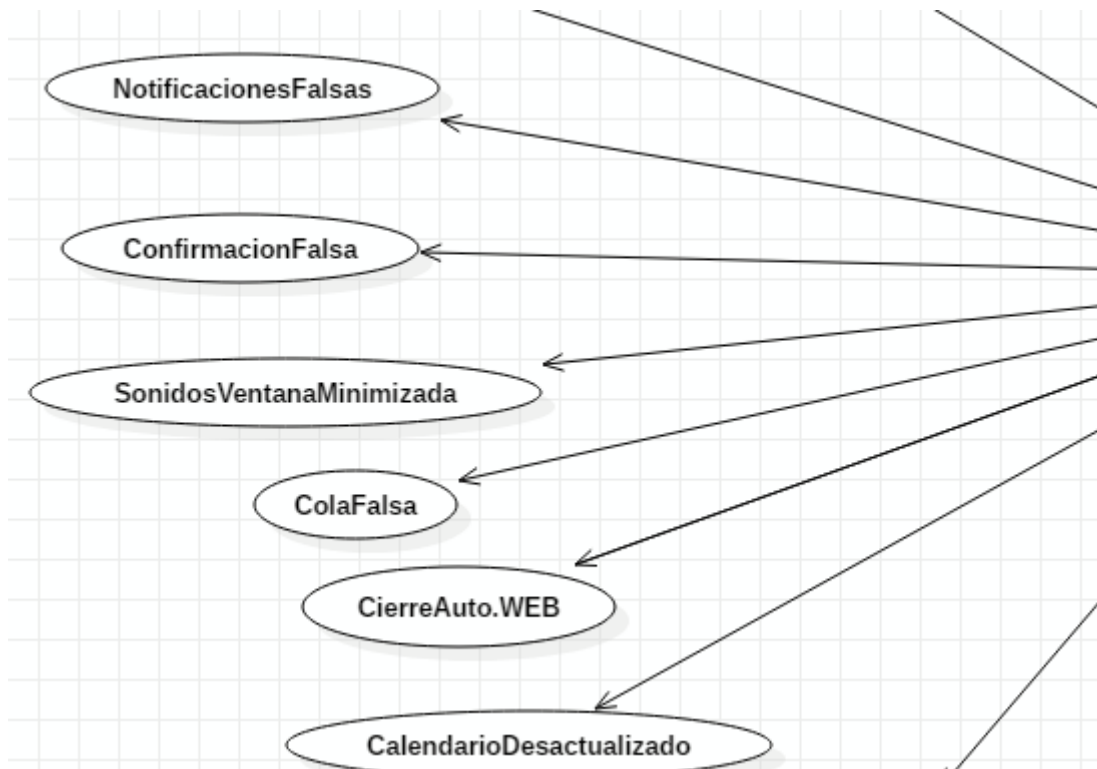


Dentro de estos campos visuales se encontrarán los casos de uso que se podrán dar. La interacción fundamental en nuestra web es que los usuarios puedan molestar entre sí y eso es lo que hace los casos de uso de enviar audios y videos espontáneos como, girar 189º e invertir eje ratón. A futuro nos gustaría implementar otros casos de uso para la interacción entre usuarios porque es la esencia de la web.

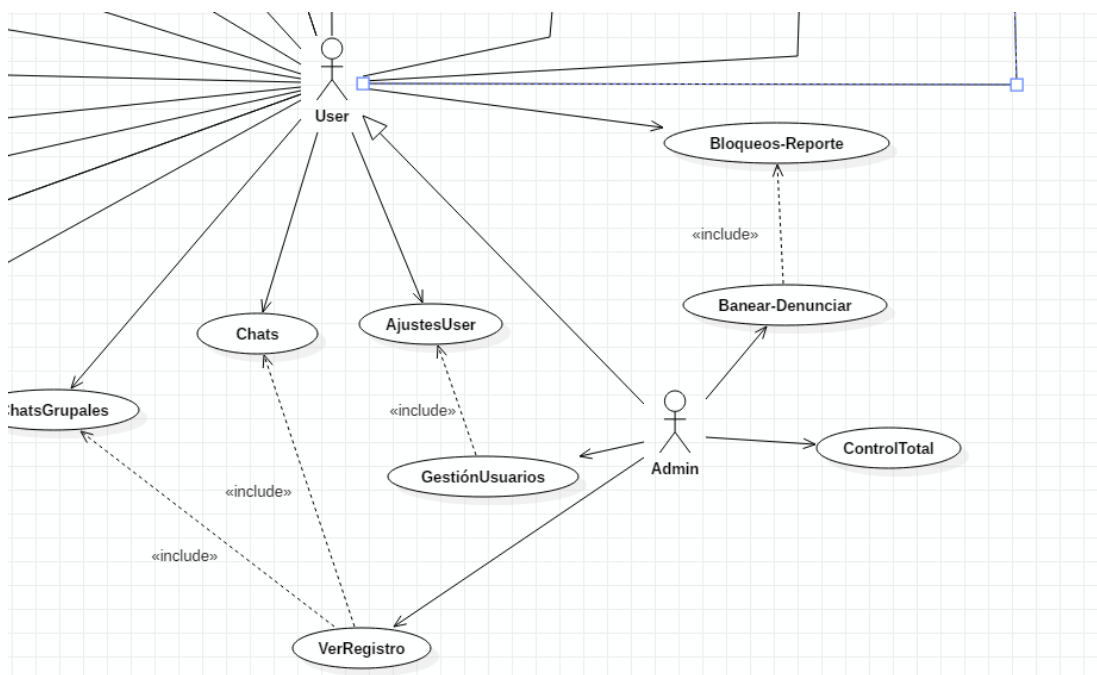


Por supuesto la propia web tendrá sus propias formas de molestar a los usuarios individualmente, como una cola de espera falsa para entrar a la web o notificaciones falsas en la bandeja de entrada entre otras cuantas.



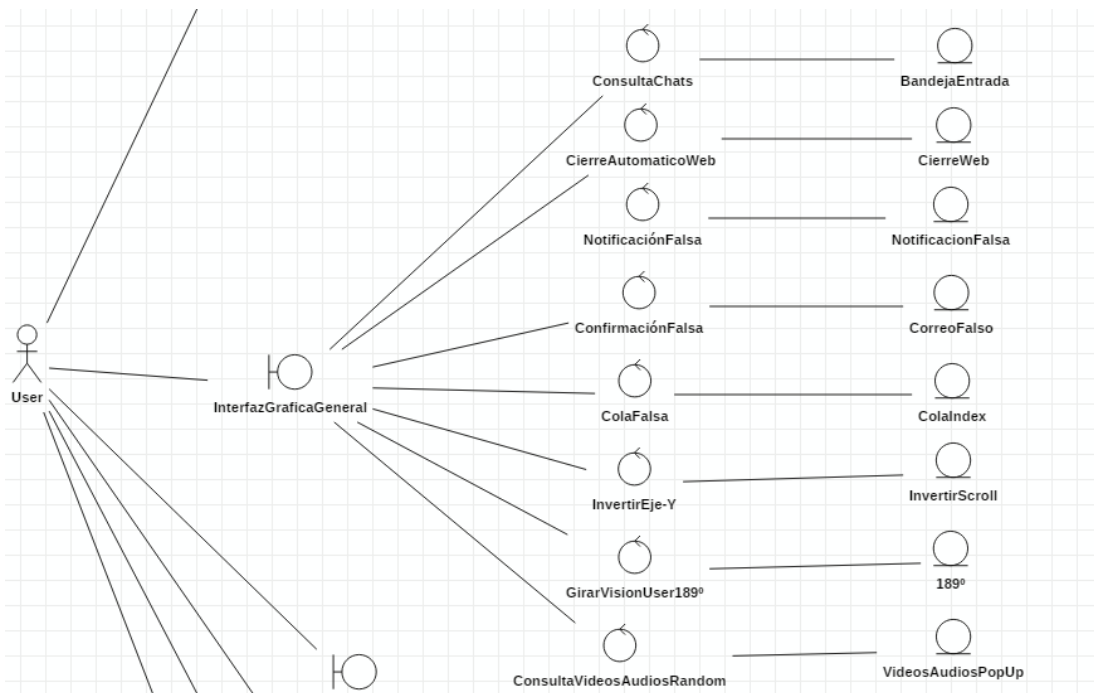
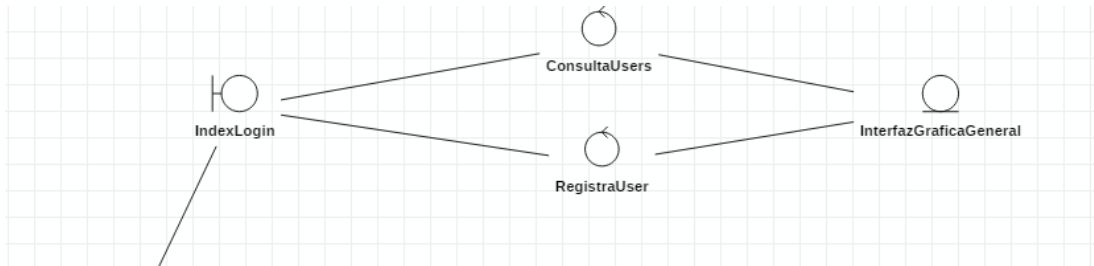


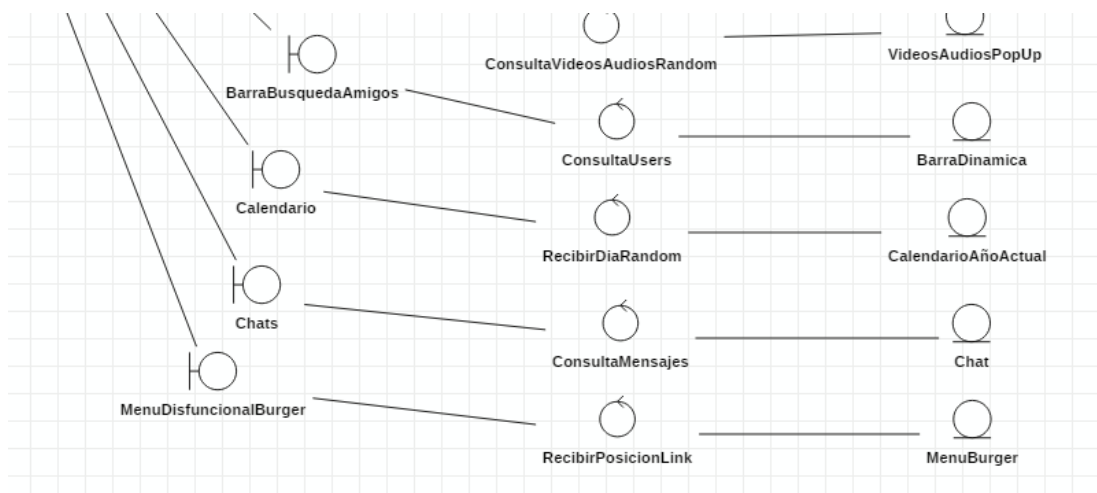
Como casos de uso enfocados en la gestión de nuestra web, le damos al usuario administrador la capacidad de acceder al registro de los chats, y la capacidad de recibir notificaciones de queja de los usuarios sobre otros usuarios que estén practicando algún tipo de actividad poco ética.



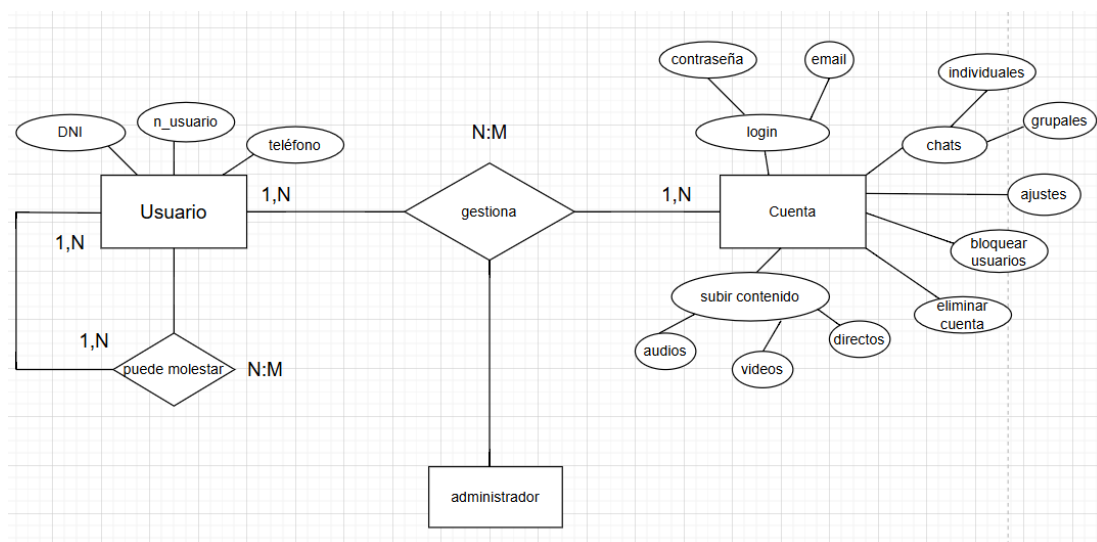
#### 4. Diagrama de clases de análisis

El diagrama de clases de análisis se enfoca en las operaciones que realiza la web en las distintas interfaces gráficas y que es lo que devuelven. Las interfaces gráficas son la página para el login del usuario y una gran ventana después del inicio de sesión en el que se incluirán el resto de las interfaces gráficas.





## 5. Diagrama entidad-relación



Este **diagrama** representa cómo se organizan y relacionan los distintos elementos de nuestra red social.

El **usuario** es la persona que utiliza nuestra red social. Tiene un nombre, un DNI y un teléfono. Cada usuario puede molestar a mínimo un usuario o a varios a la vez y ser molestado por uno y varios también.

Cada **cuenta** es gestionada por los usuarios. Un usuario puede gestionar una cuenta o varias y una cuenta puede estar gestionada por varios usuarios. La cuenta tiene sus propios atributos como el login con su email y contraseña además de algunas funcionalidades como disponer de chats tanto individuales como grupales, el permiso del usuario para subir contenido, bloquear usuarios, eliminar su propia cuenta o realizar algunos ajustes.

También contamos con el **administrador**, un usuario especial que gestiona cuentas también, pero con permisos especiales para controlar todo lo que ocurre en la plataforma. Tiene acceso a más herramientas y puede revisar el comportamiento de los usuarios.

## Bibliografía

Hemos utilizado prompts en ChatGPT para los siguientes apartados del prototipo:

Después de haber hecho el JS del [menú disfuncional](#), Le envié este prompt y después le enseñé el HTML y JS del prototipo. “la opcion X del array la cambio por esta: { nombre: "Login", enlace: "./Otras/prototipoLogin.html" } como verás quiero hacer un login, pero como es un prototipo no quiero hacer una base de datos, voy a tener ya 2 usuarios predeterminados y quiero hacer que sean accesibles, me ayudas a hacer el js necesario? Aquí te envío la estructura del login.html”.

### RESPUESTA

```
// Usuarios predeterminados
const usuarios = [
  { usuario: "usuario1", clave: "1234" },
  { usuario: "admin", clave: "admin123" }
];

document.getElementById("loginForm").addEventListener("submit", function(event) {
  event.preventDefault();

  const user = document.getElementById("username").value.trim();
  const pass = document.getElementById("password").value.trim();
  const errorMsg = document.getElementById("error");

  const encontrado = usuarios.find(u => u.usuario === user && u.clave === pass);

  if (encontrado) {
    // Acceso permitido
    localStorage.setItem("usuarioActual", user);
    window.location.href = "../prototipoPrincipal.html"; // Cambia esto según tu página principal
  } else {
    errorMsg.textContent = "Usuario o contraseña incorrectos.";
  }
});
```

### PROMPT

“ahora me gustaría hacer el apartado de buscar usuarios, pero como solo es un prototipo, solo voy a poder buscar a un usuario, que es uno de los que creamos en el login; agregué otro más para que haya dos usuarios y un admin: usuario1 y usuario2, quiero que dependa de si busca usuario1 la palabra "usuario" le salga el otro usuario y viceversa, si el admin inicia sesión, que pueda buscar a ambos”

### RESPUESTA

```
function buscarUsuarios() {
  const input = document.getElementById("inputBusqueda").value.toLowerCase();
  const resultados = document.getElementById("resultadosBusqueda");
  resultados.innerHTML = "";

  const usuarioActual = localStorage.getItem("usuarioActual");
  if (usuarioActual) {
    resultados.innerText = "Inicia sesión para buscar usuarios.";
    return;
  }

  const usuarios = ["usuario1", "usuario2"];
  let visibles = [];

  if (usuarioActual === "usuario1") {
    visibles = ["usuario2"];
  } else if (usuarioActual === "usuario2") {
    visibles = ["usuario1"];
  } else if (usuarioActual === "admin") {
    visibles = usuarios;
  }

  const encontrados = visibles.filter(u => u.includes(input));

  if (encontrados.length > 0) {
    resultados.innerHTML = "<ul>" + encontrados.map(u => "<li>{u}</li>").join("") + "</ul>";
  } else {
    resultados.innerText = "No se encontraron usuarios.";
  }
}
```

**ESTE CÓDIGO LO MODIFICO PARA QUE NO APAREZCA EN FRENTE DE TODO. SOLO ME SIRVIÓ DE PLANTILLA**