Vamos a realizar **una aplicación web para encontrar pareja** (nombre a tu elección). Será una web con funcionalidades que permitan establecer amistad entre personas, rellenar y editar un perfil persona, sugerir personas para amistad según ese perfil y usando un algoritmo que se valorará especialmente y algunas otras funcionalidades que detallaremos a continuación. También deberemos idear algún sistema en el que si dos personas muestran interés se establezcan como amigos y puedan verse el uno al otro cuando ingresen en el sistema y puedan chatear si lo desean.

La **aplicación** constará de:

* Una **página de inicio** que permitirá entrar en el sistema. En esta página de inicio se permitirá también enlazar a la página de registro y a la página de restaurar contraseña.
* La **página de registro** permitirá dar los datos personales (incluyendo *email* que servirá como usuario de inicio de sesión).
* Una vez dentro del sistema como usuario estándar tendremos un ***dashboard*** con las siguientes funcionalidades:
  + Ver **usuarios de mi preferencia** lo más afines posible a mis gustos.
  + Permitir **establecer qué usuarios me gustan** (o dejan de gustar) con un simple click. En caso de que sea recíproco, nos convertiremos automáticamente en amigos.
  + Deberíamos poder **listar los amigos** que tenemos. El formato será libre.
  + **Enviar mensajes** a otros usuarios (amigos o no).
  + Ver si algunos de los usuarios que me gustan (y yo les gusto) están **en línea.**
  + **La primera vez que entramos al sistema** nos pedirá que rellenemos un formulario con nuestras preferencias. Estas preferencias se podrán editar siempre, así como nuestro perfil (foto, nick, clave de acceso…).
  + **Avisos de mensajería pendiente de leer**.
* Como **administrador** podemos, con un CRUD**, gestionar a los usuarios**: altas, bajas, modificaciones, activaciones… los usuarios estarán desactivados por defecto hasta que los active el administrador. También se podrán crear a otros administradores (o quitarles los privilegios a los actuales).
* Los usuarios podrán enviar mensajes a otros usuarios y ver los mensajes que otros usuarios les han enviado. A estos mensajes se les podrán adjuntar archivos, lo obligatorio es que se pueda adjuntar al menos uno; como optativo, pero muy valorado, es que permita más de un archivo adjunto. El tamaño del archivo estará limitado en tamaño.
* Las **preferencias** que un usuario podrá poner son las siguientes (se pueden añadir las que se crean necesrias y serán parametrizables, el administrador las puede editar y añadir/borrar):
  + Relación seria 🡪 indicarán si su finalidad es una relación seria o esporádica.
  + Deportivos 🡪 valor numérico que indica, de 0 a 100, su gusto por los deportes.
  + Artísticos 🡪 valor que indica su inquietud artística.
  + Políticos 🡪 interés por la política, valorando de 0 a 100.
  + Tiene/Quiere hijos o no 🡪 Indicando si tiene y/o si quiere hijos.
  + Interés en: hombres, mujeres o ambos.

Todos estos valores se tendrán en cuenta a la hora de mostrar a posibles parejas candidatas en la pantalla principal del usuario y siempre podrán ser editados por los usuarios.

* Los **usuarios** verán, por lo tanto, a los candidatos más afines a sus propios valores. La forma de hacer esta parte es libre, se valora el algoritmo ideado.
* Cuando un **usuario se conecte** se habilitará, de algún modo, **un sistema que indique los amigos que tiene y que están conectados en ese momento,** así como el número total de usuarios conectados (amigos o no).
* Las páginas de la aplicación deberán ser coherentes en estilo, con cabecera y pie de página. En la cabecera sería bastante útil y bien valorado que siempre esté la información básica del usuario conectado y las opciones que permitan editar el perfil de los mismos, tal y como se indicó anteriormente. En el footer podemos poner el nombre del programa y del creador (vuestro nombre).
* El **administrador podrá generar eventos** a los que se apuntan los socios que quieran asistir. Dicho evento tendrá una fecha de realización y se cerrará el plazo de inscripción un día antes. Los usuarios asistentes podrán descargarse en pdf un resumen informativo del evento. Dicho evento también constará de una geolocalización y será editable hasta el momento de su cierre. El administrador podrá borrar a gente del evento antes de su cierre.

Se tendrá en cuenta a nivel general:

* Programación clara, robusta y funcional.
* Diseño de la BD que recoja toda la información de forma correcta.
* Que lo programado se ajuste a lo pedido.
* Una planificación de tareas coherente.
* Que los sprints de desarrollo cumplan lo planificado.
* El proyecto deberá estar sincronizado con un repositorio Git y tener una rama, mínimo, de desarrollo.
* La planificación debe estar siempre accesible por los profesores; se recomienda usar el *board* de Github.

Cada profesor indicará en su tarea los criterios que estime oportunos.

**Requisitos básicos cliente:**

* Controlar introducir el token a los servicios mediante un interceptor.
* Validaciones en todos los campos introducidos por el usuario (controlando nulos, correos bien formados etc).
* Implementación de un websocket para el servicio de mensajería y para avisar que hemos encontrado a una persona que también le gustamos.
* Protección de rutas mediante guards o comprobación de rol en los componentes.
* Tratamiento correcto de los errores por parte del servidor. Mostrar de forma correcta mediante mensajes genéricos que se ha producido un error en el servicio.
* Llevar a cabo una organización del sistema de carpetas en el proyecto de manera uniforme, si se establece un criterio de ordenación, respetarlo hasta el final del desarrollo.

**Requisitos básicos servidor:**

* Rutas, buena elección, organización y protección.
* Middlewares y validators. Token de seguridad y *abilities*.
* Controladores bien organizados y optimizados.
* Conexiones encapsuladas y organizadas correctamente.
* Migrations, Seeders y factories que permiten desplegar la base de datos de la aplicación con datos funcionales.
* Modelos con las asociaciones de Sequelize bien definidas y usadas en los lugares convenientes. Si la consulta es más óptima realizada en SQL sin mapeo también será correcto, en aquellos accesos a bd que así lo requieran.
* Gestión de imágenes.
* Websockets.
* Porcentaje desarrollado.
* Funcionalidades desarrolladas.
* Planificación personal.
* Tests.