



Socialine

An Intimate, Anonymous Social Network

Designed and Developed by **Arturo Rodríguez Romero**

Índice

Resumen, Abstract	3
Introducción	4
Tecnologías Utilizadas	6
Diseño Técnico	8
Alcance Funcional del Sistema	12
Organización del Proyecto	14
Código Documentado, Presupuesto	16
Manual de Instalación	17
Manual de Usuario	18
Análisis de Competencias	22
Conclusiones	23

1 – Resumen

En este documento voy a hablar sobre **Socialine**. Una aplicación web que llevo desarrollando desde noviembre de 2017. Hablaré sobre la motivación, la idea, el aspecto técnico y mi experiencia en el desarrollo.

2 – Abstract

In this document I will be talking about **Socialine**. A web application I've been developing since November of 2017. I will talk about the motivation, the idea, the technical aspect and my experience in the development.

3 – Introducción

Mi objetivo principal en el desarrollo de este proyecto era demostrarme a mí mismo que sería capaz de aprender un conjunto de tecnologías totalmente desconocidas para mí y desarrollar una aplicación (más o menos compleja) con ellas.

Dicho esto, desde el principio tuve claro las características que quería que tuviese el proyecto:

- Debía utilizar una base de datos **no relacional**.
- Debía utilizar una base de datos en **tiempo real**.
- Debía ser **escalable**.
- Debía priorizarse la **experiencia de usuario**.
- Debía ser usable tanto en **desktop** como en **mobile**.

Ahora, tenía que buscar una idea para la aplicación, que cubriese alguna necesidad o la crease.

Después de muchas consideraciones, llegue a la siguiente idea:

"Una aplicación que permita a personas que no se conocen de nada, pero viven, trabajan o estudian muy cerca, entablar en una conversación abierta, anónima, íntima y sin pretenpresiones sociales establecidas."

Esta idea viene de mi observación de que en un ámbito social, solo interactuamos con los que tienen más cosas en común con nosotros, o con los que ya conocemos.

En una universidad, la gente se junta con los de su clase o facultad.

En una empresa, la gente se junta con los de sus compañeros o la gente de su departamento.

Por lo general la gente no sale ni intenta salir de su grupo de confort. Ya sea por vergüenza, timidez, miedo o porque no saben que pueden tener en común con la otra persona.

Y este comportamiento me parece una lástima porque se puede aprender mucho de gente con la que uno no tiene nada en común a primera vista.

4 – Tecnologías Utilizadas

Ahora que ya tenía una idea, debía encontrar un conjunto de tecnologías que me ayudasen a alcanzarla.

Después de investigar muchos frameworks tanto como de backend como de frontend, llegue a los que se adaptaban a mis necesidades de la mejor manera posible:

Backend:

Node.js: es un entorno de ejecución para JavaScript construido con el motor V8 de Chrome. A diferencia de servidores más clásicos como Apache o Tomcat, funciona con un sistema de operaciones E/S sin bloqueo y orientado a eventos, que lo hace liviano y eficiente. Node y su ecosistema de librerías 'npm' es uno de los servidores por excelencia de las aplicaciones web.

NeDB: es una base de datos no relacional embebida (o en memoria) de Node.js basada en directorios y ficheros. La API es un subset de MongoDB y es muy rápida (43.290 operaciones de select por segundo).

Feathers.js: es un framework web de micro-servicios en tiempo real de Node.js que te permite crear y acceder a recursos a través de API REST, sockets y plugins. Feathers es un puente de conexión entre Express.js, socket.io y la base de datos elegida por el desarrollador. Esto le permite a la aplicación suscribirse a un servicio, que le notificará cuando este se actualize. Lo que le permite crear aplicaciones en tiempo real, y mantener el número de peticiones al mínimo.

Frontend:

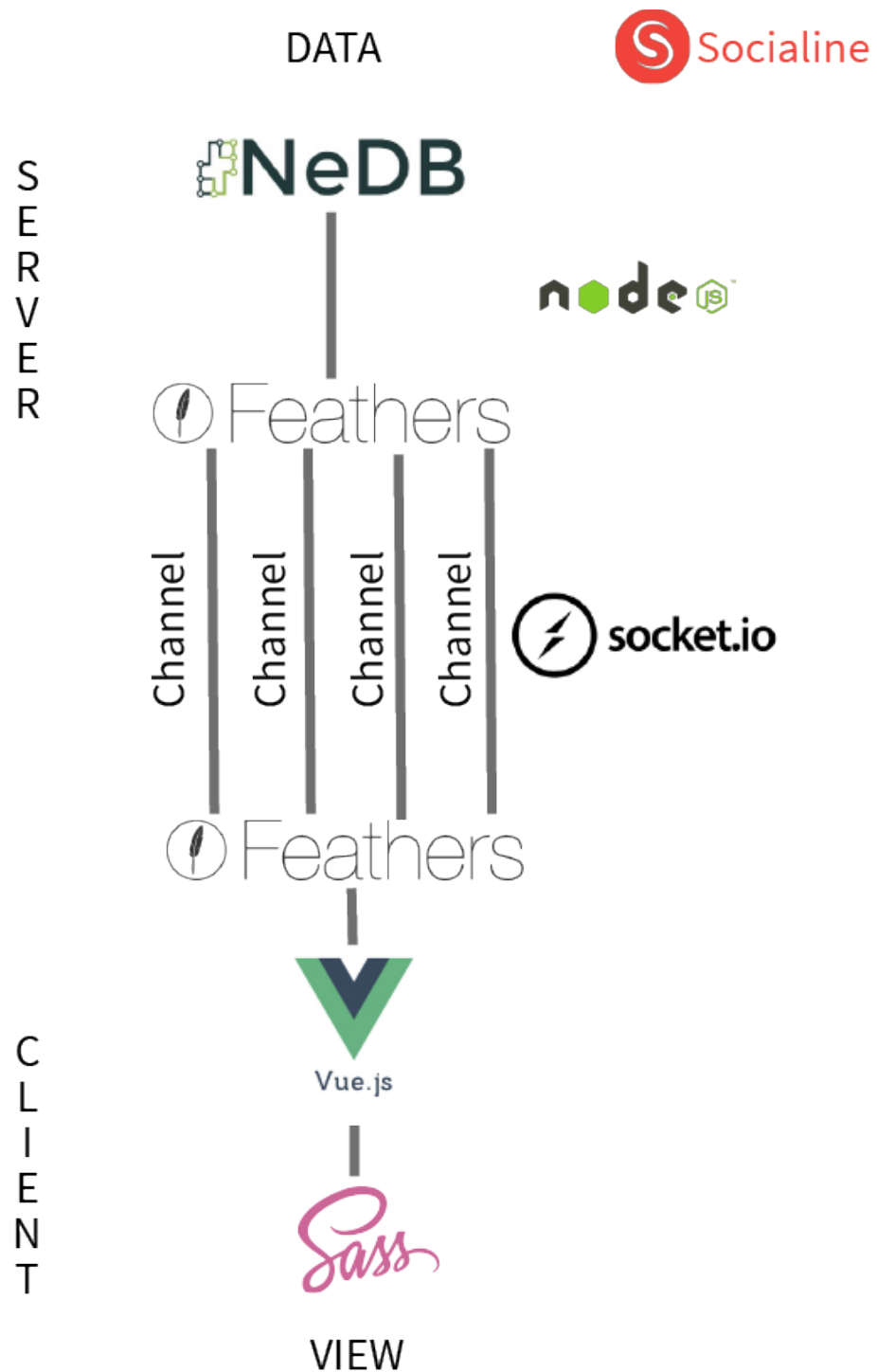
Vue.js: es un framework para crear interfaces web de forma progresiva. Permite la construcción de aplicaciones web complejas y en tiempo real de forma sencilla para el desarrollador. Es muy parecido a otros frameworks web como Angular o React.

Sass: es un preprocesador de CSS que le da al desarrollador más facilidades a la hora de diseñar una aplicación web.

A continuación explicaré como estas tecnologías enlazan entre si para crear Socialine.

5 – Diseño Técnico

Para exponer mejor como estas tecnologías se enlazan para crear Socialine, he creado el siguiente esquema:



Backend:

NeDB guarda los datos en ficheros de texto plano en formato JSON. Las tablas de Socialine son las siguientes:

- **Cuentas:** guardan la información de login de los usuarios.
 - Id.
 - Nombre.
 - Contraseña (Encriptada y hasheada).
- **Usuarios:** guardan los datos de los clientes.
 - Id.
 - Id cuenta.
 - Nombre.
 - Imágen perfil.
 - Color mensajes.
 - Imagen fondo.
 - Última conexión.
- **Mensajes:** guardan los mensajes de los clientes.
 - Id.
 - Id emisor.
 - Id receptor.
 - Texto.
 - Hora emisión.
 - Ha sido leído por receptor.

Feathers está compuesto por microservicios. Estos servicios son los encargados de esperar peticiones de usuarios y actualizar la persistencia. En el caso de Socialine, hay un servicio por cada tabla de la base de datos.

Todos estos servicios son realtime, lo que significa que los usuarios se suscriben al canal del servicio, y las actualizaciones

les llegarán a través del socket web.

La seguridad era un factor importante a la hora de desarrollar Socialine.

Por esto, se establecen unos **límites** a los que los usuarios pueden acceder:

- Un usuario solo puede modificar sus propios datos.
- Solo pueden acceder a la base de datos los usuarios que estén autenticados.
- Un usuario solo puede acceder a mensajes en los que participa, ya sea de emisor o de receptor.

```
app.service('messages').publish((data, hook) => {  
  return [  
    app.channel(`userId/${data.sender}`),  
    app.channel(`userId/${data.receiver}`)  
  ];  
});
```

El canal de mensajes asegura que un mensaje solo llega al emisor y al receptor del mensaje.

También se usa **JWT** (JSON Web Tokens) para el inicio de sesión.

Esto significa que cuando el usuario hace login con su nombre y contraseña, el servidor genera una cadena de caracteres aleatorios que le servirá como "llave".

Cada operación que haga, llevará incrustado el Token que le representa. Esto también hace que sea más seguro mantener abierta la sesión, sin tener que guardar el nombre y la contraseña.

Frontend:

Feathers.js también se ejecuta en el cliente, lo que facilita mucho la conexión con el servidor. El cliente se suscribe a un servicio, y cuando ese ocurra un cambio en el servicio, el cliente será actualizado.

```
messagesService.on('created', message => {
```

Cliente Feathers.js se suscribe al servicio de mensajes.

Vue.js permite al programador mantener la vista actualizada con los datos del modelo sin tener que programar un complicado sistema de actualización de elementos HTML. Esto significa que pude centrarme en añadir características y optimizarlas.

```
<div class="user-item" v-for="user in users"
```

El renderizado declarativo de Vue permite usar for o if dentro del propio HTML.

6 – Alcance Funcional del Sistema

En este punto explicaré las distintas funcionalidades que he implementado en la aplicación.

Creación de usuario

Permite a un nuevo usuario crear una cuenta con la que iniciar sesión.

Inicio de sesión:

Permite a un usuario registrado acceder a su cuenta.

Inicio de sesión automático

Si el usuario ha iniciado sesión hace menos de 24 horas, la sesión se mantendrá abierta, para que no tenga que volver a introducir los datos.

Ver Contactos Cercanos

Un usuario verá a los contactos cercanos, dentro de la distancia máxima elegida por el usuario. Se podrá bloquear o añadir a favoritos a cualquier usuario.

Mandar Mensajes

Un usuario puede mandar mensajes a cualquiera de los contactos que estén dentro del rango de distancia máxima elegido. Se puede mandar texto plano, una URL, o la URL de una imagen (Que mostrará una miniatura).

Recibir Mensajes

Un usuario recibirá todos los mensajes que otro usuario le mande. Estos le serán notificados dentro de la aplicación, en la pestaña del navegador y (si lo tiene activo) en las notificaciones Web nativas del navegador.

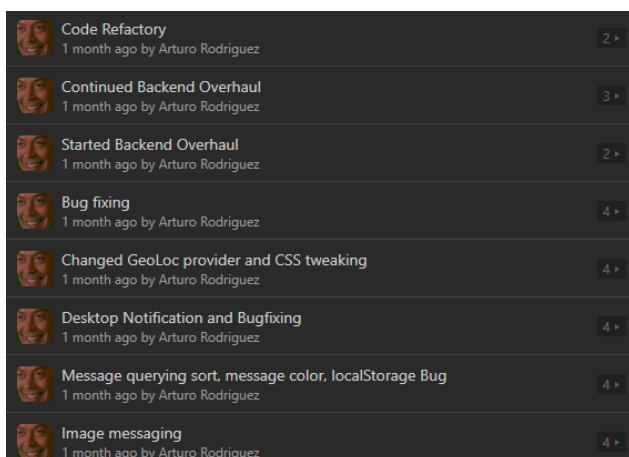
Personalizar Perfil

El usuario podrá personalizar su nombre, su estado, su foto de perfil, su imagen de fondo, su color de mensaje y la distancia máxima de los contactos que quiere que se le muestren.

7 – Organización del Proyecto

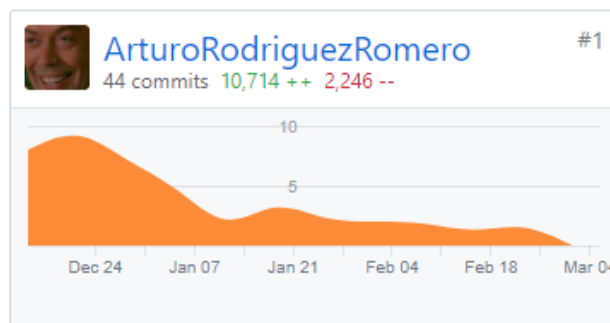
Para organizar el proyecto he utilizado Github y Trello.

Github.



Es un servicio de repositorios remotos que utiliza Git para llevar el control de versiones de tu proyecto. Te permite crear ramas secundarias a la principal para llevar un desarrollo paralelo, unir ramas distintas, volver a un estado anterior del proyecto, etc .

También te da algunas estadísticas interesantes, por ejemplo, desde el comienzo del proyecto, he hecho 44 commits y he añadido más de 10,000 líneas de código.

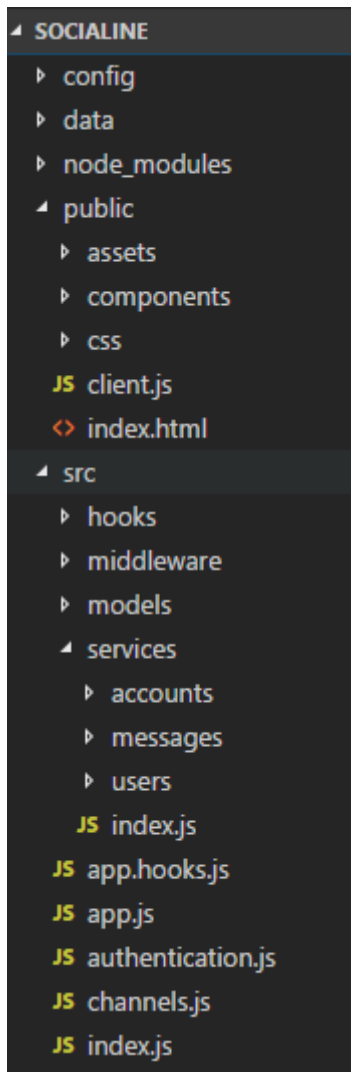


Trello.

Es una herramienta web de planificación de proyectos basada en la metodología ágil Kanban. Utiliza un sistema de columnas y tarjetas para mantener el proyecto organizado y saber qué tienes que hacer en todo momento.

El hecho de que todos estos servicios estén en la nube me permitió acceder a ellos desde cualquier lugar, y poder continuar el desarrollo en cualquier lugar.

Organización de las carpetas del proyecto.



- Config: contiene muchos configuraciones para el servidor de Node.
- Data: guarda los archivos .db de la base de datos.
- Node_modules: contiene todas las librerías importadas por npm.
- Public: guarda la parte de frontend de la aplicación.
- Src:
 - Services: contiene los servicios utilizados.
 - Models: guarda las clases de los servicios.
 - Hooks: contiene triggers del servidor.

8 – Código Documentado

El proyecto puede ser encontrado en este [link](#) de GitHub.

9 – Presupuesto

Coste de desarrollo:

Concepto	Coste
Visual Studio Code	0 €
Github	$(7€ / \text{Mes}) * 3 \text{ Meses} = 21€$
Personal	$(1600€ / \text{Mes}) * 3 \text{ Meses} = 4800€$
Total	4821€

Coste de mantenimiento:

Concepto	Coste
Servidor (Azure)	~100€ / 100.000 Usuarios
Personal	1600€
Github	$(7€ / \text{Mes}) * 3 \text{ Meses} = 21€$
Total Mensual	~1721€

10 – Manual de instalación

Para desplegar la aplicación, se deben seguir los siguientes pasos:

Instalar Node.js:

Descargar para Windows (x64)

8.10.0 LTS

Recomendado para la mayoría

9.9.0 Actual

Últimas características

Dentro del directorio de Socialline, abrir PowerShell:

```
PS F:\Proyectos\socialline-repo\socialline> npm install
```

Instalará las dependencias en el directorio.

```
PS F:\Proyectos\socialline-repo\socialline> npm start
> socialline@0.0.0 start F:\Proyectos\socialline-repo\socialline
> node src/

info: Feathers application started on http://localhost:3030
```

Despliega la aplicación en el puerto seleccionado.

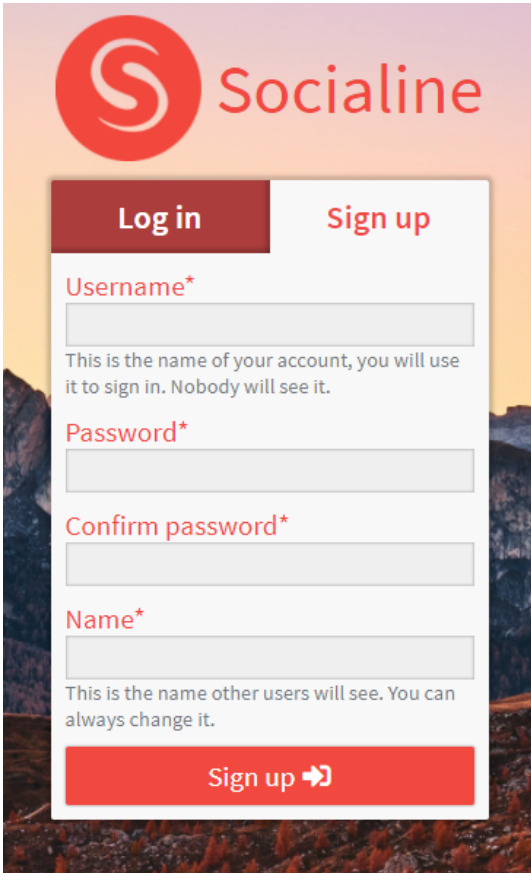
11 – Manual de Usuario

Al tratarse de una aplicación web, el usuario puede utilizarla desde cualquier plataforma que tenga un navegador Web. Ya sea Windows, MacOS, Linux, Android, iOS, etc.

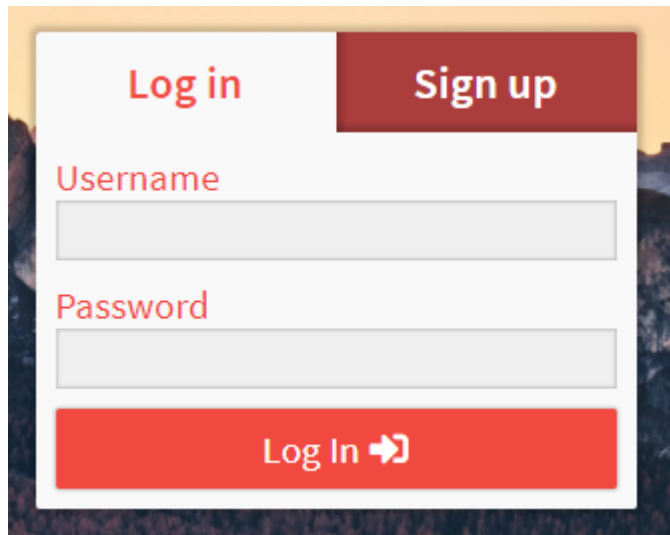
Crear usuario.

Un usuario puede crear una cuenta para acceder a la aplicación de forma sencilla.

La razón por la que no se pide correo electrónico es porque pienso que la gente suele ser reticente cuando una aplicación que no conocen les pide su correo personal. Al pedirles un nombre de usuario, el usuario entrará a la aplicación más positivamente.

The image shows a 'Sign up' form for 'Socialline'. At the top, there is a red circular logo with a white 'S' and the word 'Socialline' in red. Below the logo, there are two tabs: 'Log in' and 'Sign up', with 'Sign up' being the active tab. The form contains four input fields: 'Username*', 'Password*', 'Confirm password*', and 'Name*'. Each field has a corresponding text description below it. The 'Username' field description says 'This is the name of your account, you will use it to sign in. Nobody will see it.' The 'Name' field description says 'This is the name other users will see. You can always change it.' At the bottom of the form, there is a red button labeled 'Sign up' with a right-pointing arrow icon. The background of the form is a light gray, and the overall background of the image is a scenic landscape with mountains and a sunset sky.

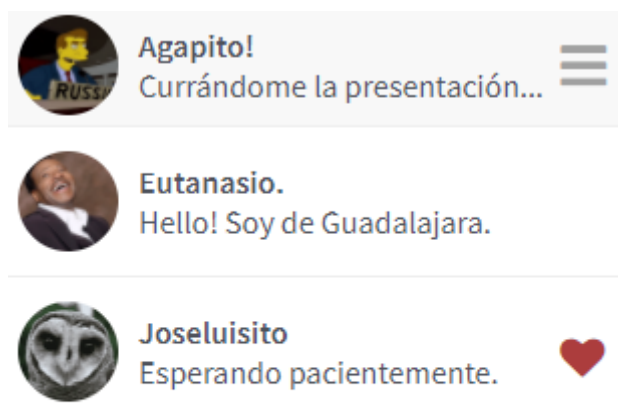
Iniciar sesión.



A login and sign up form with a white background and a red border. At the top, there are two buttons: 'Log in' in red text on a white background, and 'Sign up' in white text on a red background. Below these are two input fields: 'Username' and 'Password', both with red labels and white input boxes. At the bottom is a large red button with the text 'Log In' and a right-pointing arrow.

Para iniciar sesión es tan sencillo como insertar el nombre de usuario y la contraseña.

Contactos cercanos.



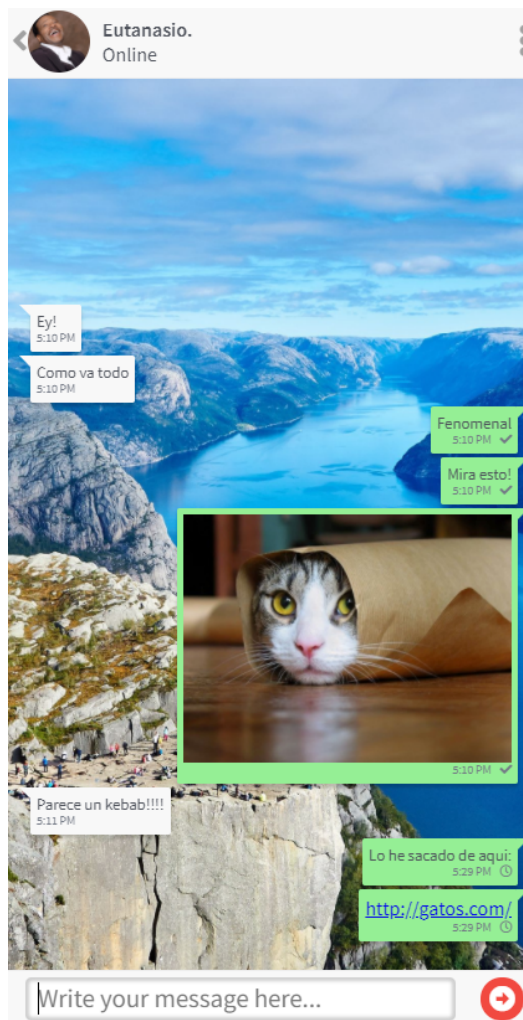
La lista de contactos te muestra todos los usuarios que estén en el rango de distancia elegido.

Mensajería instantánea.

La zona de chat permite a los dos usuarios tener una conversación a través de texto plano, imágenes y gifs, e hipervínculos.

Los mensajes también muestran a los usuarios información extra, como si han sido leídos por el receptor o la hora de envío.

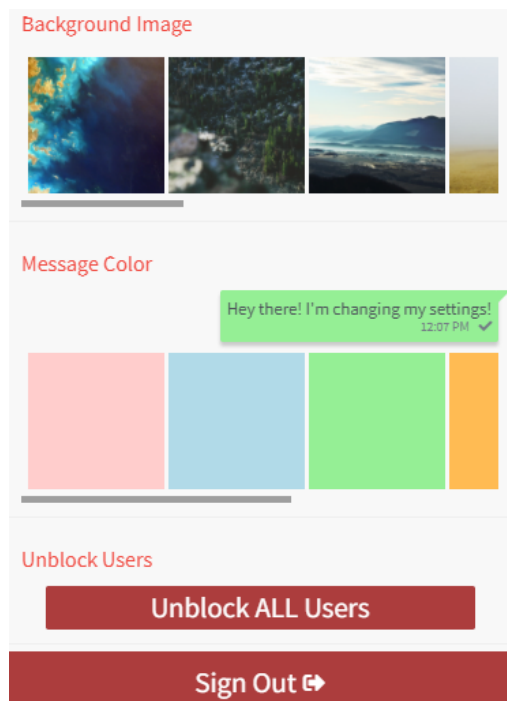
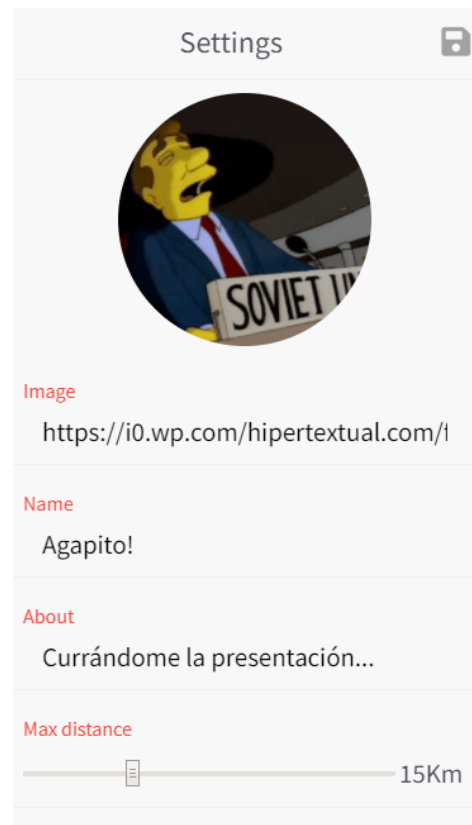
También permite bloquear a un usuario o agregarlo a favoritos.



Menú de personalización.

El menú de personalización te proporciona muchas opciones para hacer que tu experiencia sea única.

- Imágen: la imagen que otros usuarios verán.
- Nombre: el nombre que otros usuarios verán.
- Sobre tí: un corto texto que diga algo sobre tí.
- Distancia máxima: la distancia máxima con los usuarios que te aparecerán en el menú de contactos.
- Imágen de fondo: la imagen de fondo de la aplicación. Las imágenes a elegir son sacadas de una API de fotos aleatorias.
- Color de mensajes: el color que quieres que tus mensajes tengan.
- Desbloquear usuarios: desbloquea a todos los usuarios bloqueados.



12 – Análisis de las competencias y clientes

Para analizar las competencias del mercado voy a separar las características del proyecto en positivas y negativas.

Positivo	Negativo
Facilidad de uso	Bajo presupuesto
Utilidad real	Muchísimos competidores
Gratuito	
Personalización	
Multiplataforma	
Anónimo	

Los clientes objetivo de la aplicación serían:

- Estudiantes aburridos en clase.
- Soltero/as en busca de pareja.
- Gente nueva en una ciudad en busca de amistad.
- Trabajadores de un mismo centro de trabajo que quieren mejorar su relación.

13 – Conclusiones

Sobre los objetivos y las motivaciones

El objetivo principal del proyecto era aprender por mí mismo un conjunto de tecnologías desconocidas, y creo que lo he cumplido.

Está claro que no he usado todas las características que ofrecen estas tecnologías, pero lo que he usado es un cimiento estable para poder utilizarlas día a día y seguir aprendiendo.

Sobre las funcionalidades de la aplicación

Creo que todas las funcionalidades de la aplicación cumplen su objetivo bien y técnicamente están implementadas correctamente y pueden ser escaladas y mejoradas según las necesidades futuras.

Sobre las mejoras de la aplicación

Aunque no quiero que la aplicación se convierta en una amalgama de funcionalidades sin un propósito unificado, sí que creo que hay un par de funcionalidades que podrían mejorar la experiencia del usuario:

- Subir imágenes del dispositivo local.
- Permitir el transpaso de archivos.