# Desarrollo de una aplicación web de intercambios de idiomas

*KeepTalking*

**Autora: Isabella Carranzani Borot**

**Máster en Desarrollo de Sitios y Aplicaciones Web**

Informática, Multimedia y Telecomunicación

**Nombre consultor de TF**
Ignasi Lorente Puchades

**Profesor responsable de la asignatura**
César Pablo Córcoles Briongos

**Enero de 2023**

Universitat Oberta de Catalunya

# Créditos

# Ficha del trabajo final

| | |
|---|---|
| **Título del trabajo:** | *Desarrollo de una aplicación web de intercambios de idiomas - KeepTalking* |
| **Nombre del autor:** | *Isabella Carranzani Borot* |
| **Nombre del consultor/a:** | *Ignasi Lorente Puchades* |
| **Nombre del PRA:** | *César Pablo Córcoles Briongos* |
| **Fecha de entrega (mm/aaaa):** | *01/2023* |
| **Titulación o programa:** | Máster en Desarrollo de sitios y aplicaciones web |
| **Área del Trabajo Final:** | *Informática, Multimedia y Telecomunicación* |
| **Idioma del trabajo:** | *Castellano* |
| **Palabras clave** | *Migración, Aplicación web, Laravel, React, nuevas tecnologías, intercambio de idiomas* |

**Resumen del Trabajo**

El objetivo de este trabajo es desarrollar una aplicación web, KeepTalking, plataforma para el intercambio de idiomas, donde cualquiera puede enseñar su idioma o simplemente entablar una conversación con otra persona en un idioma extranjero. Para este proyecto se quiere crear un entorno de desarrollo moderno, estudiando y siguiendo en todo momento las últimas tecnologías que puedan ser aplicables en el sitio web y mantenidas en un futuro.

Con tal de llevar a cabo dicho desarrollo se hará un breve estudio de competencia, para ver con qué tecnologías trabajan actualmente los competidores y como la aplicación puede ser desarrollada para destacar en el mercado.

Como objetivo final, al acabar el proyecto, este será presentado como idea para un futuro desarrollo con más funcionalidades.

**Abstract**

The main goal of this work is to develop a web application, KeepTalking, a platform for language exchange, where anyone can teach their language or simply start a conversation with another person in a foreign language. For this project we want to create a modern development environment, studying and following at all times the latest technologies that can be applied to the website and maintained in the future.

In order to carry out such development, a brief competition study will be made, to see which technologies the competitors are currently working with and how the application can be developed to stand out in the market.

As a final goal, at the end of the project, it will be presented as an idea for a future development with more functionalities.

# Dedicatoria

Primeramente, quiero agradecer al consultor del TFM que resolvió mis dudas y me dio muy buenas recomendaciones. A mi familia que ha soportado mis locuras mientras escribía este documento. Pero, sobre todo, a mi pareja que ha estado a mi lado en todo momento, ha aguantado mis cambios de humor e, incluso, ha trasnochado conmigo mientras yo estaba desarrollando la aplicación.

Gracias a todos, sin su apoyo, este proyecto no hubiera sido posible.

# Índice

# Lista de ilustraciones

# 1. Introducción

## 1.1. Contexto y justificación del Trabajo

En esta memoria se quiere plasmar, a modo de guía, el plan de desarrollo de una aplicación web que solucione el problema en el que se encuentran los usuarios cuando quieren mantener el conocimiento de un idioma anteriormente aprendido o cuando desean aprender uno desde cero.

Las personas normalmente estudian un nuevo idioma en escuelas oficiales o con libros de texto. Sin embargo, como todos aprecian, las tecnologías están en constante cambio y hoy en día, hay una gran variedad de aplicaciones para aprender un nuevo idioma.

Aunque en el mercado encontramos múltiples webs relacionadas con el intercambio de idiomas, estas no siempre se encuentran actualizadas y a veces traen soluciones o funcionalidades que no son del todo útiles o no funcionan correctamente.
Al tener una aplicación web con ideas claves ante la competencia, como es KeepTalking, se ha decidido desarrollar la idea y documentar tal trabajo para que en un futuro los usuarios que la utilicen se sientan motivados a aprender.

Debido a ello, en este plan, se encontrarán los objetivos, metodologías y herramientas que se utilizarán para el correcto desarrollo de la aplicación web KeepTalking.

## 1.2. Objetivos del Trabajo

### 1.2.1. Principales

Los objetivos principales del proyecto son los siguientes:

- Desarrollar las principales funcionalidades de una aplicación web que sea escalable, robusta y flexible.
- Documentar cada una de las fases del desarrollo hasta obtener los resultados deseados.
- Profundizar en el conocimiento de las tecnologías empleadas en este proyecto, en este caso MySQL para la base de datos, Laravel para el back-end y React en el front-end.
- Aportar conocimientos a lo largo del proceso de desarrollo y poder plasmar todo lo aprendido en este máster.

### 1.2.2. Secundarios

Los objetivos secundarios del proyecto son los siguientes:

- Crear una aplicación web que sea *responsive*, es decir, que se adapte a cualquier dispositivo.
- Crear un plan de desarrollo claro y conciso que sea atractivo para el público objetivo.
- Seguir la planificación de manera adecuada para gestionar correctamente el tiempo de desarrollo y el tiempo de documentación.
- Conseguir que la aplicación sea cien por ciento funcional y que no queden tareas pendientes al acabar el trabajo final.

### 1.3. Impacto en sostenibilidad, ético-social y de diversidad

El desarrollo de este TF no tiene impacto negativo en la sostenibilidad, ya que es desarrollado de manera casera solo con un ordenador. Tras su finalización tiene un pequeño impacto positivo en el aspecto de sostenibilidad medioambiental, porque al tratarse de una aplicación web para el intercambio de idiomas, permite a los usuarios no tener que desplazarse de casa para aprender, y eso ayuda a no consumir tantos combustibles que dañan nuestro ecosistema y nuestro planeta.

A su vez, también tiene un impacto muy positivo en diversidad porque al ser una plataforma de idiomas, cualquier persona puede utilizarla para aprender otro idioma y sentirse incluida se encuentre donde se encuentre. No está de más aclarar que es un sitio web al que puede acceder cualquier persona, sin importar su género.

En ningún caso, ni durante su desarrollo o exposición futura, este trabajo comporta un riesgo a los aspectos éticos-sociales, al contrario, sirve como puente entre los usuarios que estén dispuestos a transmitir sus conocimientos en idiomas extranjeros a otros usuarios que quieran aprenderlos.

### 1.4. Enfoque y método seguido

Para realizar este trabajo final se va a partir de una metodología de trabajo ágil Kanban, se dividirá el proyecto en tareas, donde se conseguirá ir construyendo

el plan de desarrollo de manera incremental. Primero se desarrollarán las características más importantes y al final se integrarán más funcionalidades.

Esta metodología permite que se construya el proyecto de manera ordenada, ya que cada objetivo marcado en las tareas permite tener una mejor planificación del tiempo, por lo que ayudará a centrarse en los objetivos y aumentará la productividad.

Al tratarse de un trabajo final con un proceso de entregas de PECs, cada uno de los hitos Kanban coincidirá con estas entregas, para poder ir manteniendo una evolución y evaluación constante para saber que se está siguiendo el plan establecido inicialmente.

Para facilitar la organización del proceso se ha hecho uso de un tablero *Kanban* de Jira, una herramienta que ayuda a gestionar proyectos y hacer un seguimiento de todos sus componentes. Al tener una interfaz intuitiva en forma de tablero, las tareas se pueden tener controladas en todo momento.

Con el tablero de Jira conseguiremos seguir la metodología ágil con el siguiente flujo de trabajo, distribuido en cinco columnas:

- **Pendientes (*Backlog*)**: Esta columna contiene todas las tareas por hacer en forma de tarjetas. Cuando la tarea pase a desarrollo se moverá a la siguiente columna.

- **En curso (*In progress*)**: Esta columna contiene todas las tareas que se están llevando a cabo en ese momento.

- **Bloqueado (*Blocked*)**: Esta columna contiene las tareas que estén bloqueadas y no se puedan continuar debido a una duda o a algún otro fenómeno. En este caso se deberá hablar con el consultor e intentar sacar la tarea de este estado lo antes posible.

- **En revisión (*Review*):** Esta columna contiene las tareas que ya están finalizadas, pero se tienen que revisar antes de darlas por terminadas.

- **Finalizado (*Done*):** Esta columna contiene todas las tareas que ya hayan sido validadas y corregidas [1]. Las condiciones que se tienen que dar para que una tarea se dé por terminada son:
    - El diseño es fiel al prototipo e incluye mejoras
    - Cumple con el caso de uso propuesto
    - Todos los componentes funcionan como es esperado
    - Las llamadas al backend devuelven los datos esperados
    - La documentación es precisa.

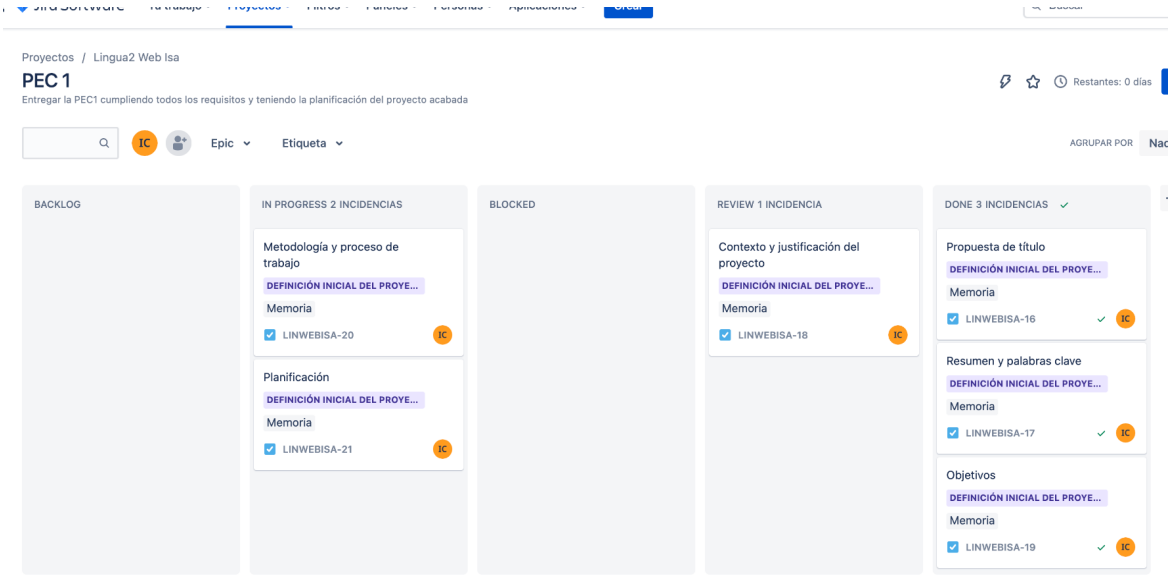



**Ilustración 1 - Tablero kanban de Jira para el TFM**

## 1.5. Planificación del trabajo

### 1.5.1. Definición de las tareas

La planificación del trabajo se ha estructurado siguiendo la metodología descrita anteriormente. Se han repartido los hitos sobre la base de las diferentes PEC, repartidas a lo largo de las dieciséis semanas de duración del TFM.

Con esta solución se pretenden alcanzar los objetivos satisfactoriamente y que la aplicación quede terminada de la mejor manera posible. También, con la ayuda de la herramienta de gestión de tareas Jira, se van a representar los hitos y sus respectivas tareas en un cronograma, al estilo de un diagrama de Gantt.

Las tareas que se van a realizar en este proyecto dependen de las funcionalidades principales que se van a crear para la aplicación, más adelante estas serán presentadas gráfica y detalladamente:

- *Landing page* o página principal
- Cuenta usuario / login
- Tablón de usuarios
- Sistema de chat (mensajería)
- Sistema de calendario y control de reuniones

### 1.5.2. Etapas del proyecto según las PEC

Junto con la definición de objetivos y el tiempo marcado en las PEC de este proyecto, se establecen las siguientes actividades:

- **Hito 1** - Definición inicial del proyecto
    - Propuesta de título
    - Resumen y palabras clave
    - Objetivos
    - Contexto y justificación
    - Metodología y proceso
    - Planificación
- **Hito 2** - Desarrollo landing page, iniciar sesión, registro y editar perfil
    - Requisitos funcionales y no funcionales
    - Diseño en Figma de dos formularios para la autenticación del usuario y la edición. Además de diseñar la landing page y la página de información de la aplicación.
    - Instalación software necesario
        - Librería de componentes Mantine para el desarrollo frontend
        - Laravel para el desarrollo backend
    - Programación backend
        - Creación de los endpoints:
            - `'/register'`: endpoint de registro de usuario

            - `'/login'`: endpoint de inicio de sesión
            - `'/logout'`: endpoint para cerrar sesión
            - `'/countries'`: endpoint para recuperar todos los países
            - `'/languages'`: endpoint para recuperar todos los idiomas
            - `'/user'`: endpoint para recuperar el usuario
            - `'/update'`: endpoint para actualizar los datos del usuario
            - Creación de modelos, controladores y migraciones
        - Modelo, controlador y tabla de usuarios
        - Modelo, controlador y tabla de idiomas
        - Modelo, controlador y tabla *pivot* entre idiomas y usuarios
        - Modelo, controlador y tabla de países
    - Programación frontend
        - Desarrollo de las vistas: *Login*, *Register*, *My Profile, Get Started y Landing Page*
        - Desarrollo de los componentes: Header y Footer
        - Gestión de rutas con ReactRouter
        - Creación de los modelos y de las queries a los endpoints
    - Pruebas para saber si todo funciona correctamente
- **Hito 3** - Tablón de usuarios
    - Diseño en Figma de la página principal
    - Programación backend
        - Creación del endpoint:

        - `'/matching_users'`: endpoint para recuperar los usuarios que coinciden con tus idiomas
    - Programación frontend
        - Desarrollo de las vistas: *Main Page*
        - Desarrollo de los componentes: UserCard y SearchInput
        - Creación de los modelos y de la query al endpoint
- **Hito 3 -** Desarrollo mensajería para entablar conversaciones.
    - Requisitos funcionales y no funcionales
    - Diseño en Figma del listado de chats y del chat.
    - Instalación software necesario
        - Instalación de Pusher, sistema de websockets online
    - Programación backend
        - Creación de los endpoints:
            - `'/chats'`: endpoint para obtener los chats
            - `'/delete_all'`: endpoint para eliminar todos los chats
            - `'/delete_chat'`: endpoint para eliminar un chat
            - `'/messages'`: endpoint para enviar un mensaje
            - `'/get_messages'`: endpoint para obtener los mensajes
        - Creación del evento mensaje
        - Creación de los canales
        - Creación de modelos, controladores y migraciones
            - Modelo, controlador y tabla de mensajes
            - Modelo, controlador y tabla de chats
    - Programación frontend

        - Desarrollo de las vistas: *Chats y Chat Panel*
        - Instalación y configuración de Pusher en React
        - Inicialización de los canales en pusher para cada chat
        - Creación de los modelos y de las *queries* a los endpoints
    - Pruebas para saber si todo funciona correctamente
- **Hito 4** - Desarrollo del calendario, para poder agendar reuniones.
    - Requisitos funcionales y no funcionales
    - Diseño en Figma de la vista del calendario
    - Instalación software necesario
        - Componente FullCalendar para React
    - Programación backend
        - Creación de los endpoints:
            - `'/meets'`: endpoint para obtener las reuniones
            - `'/meet'`: endpoint para crear una reunión
            - `'/delete_meet'`: endpoint para eliminar una reunión
        - Creación de modelos, controladores y migraciones
            - Modelo, controlador y tabla de reuniones
    - Programación frontend
        - Desarrollo de la vista: *Calendar*
        - Instalación y configuración del componente FullCalendar
        - Desarrollo del componente CreateMeetModal, para añadir una reunión desde diferentes vistas.
        - Desarrollo del componente EditMeetModal, para editar una reunión previamente agendada.
        - Creación de los modelos y de las *queries* a los endpoints

- Pruebas para saber si todo funciona correctamente

- **Memoria del TFM**
  - Elaboración estructura
  - Documentación del desarrollo
  - Revisiones y correcciones
- **Entrega final**
  - Últimas modificaciones
  - Resultados finales y Anexo
  - Vídeo demostración aplicación
  - Presentación del proyecto
- **Defensa visual**
  - Apoyo visual para la presentación
  - Defensa del trabajo

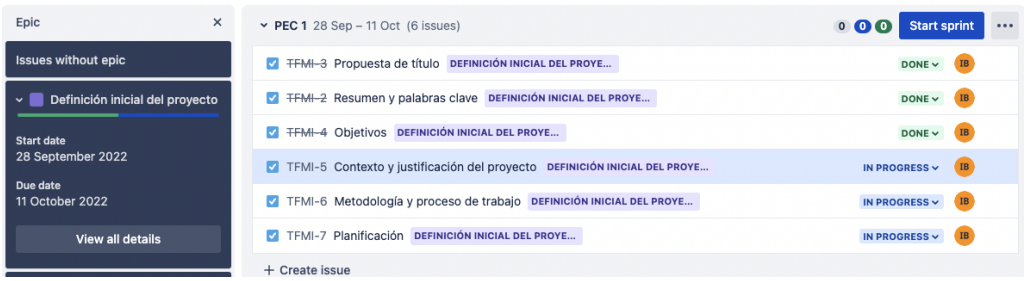



**Ilustración 2 - Hito 1, Pec1 - Definición inicial del proyecto**

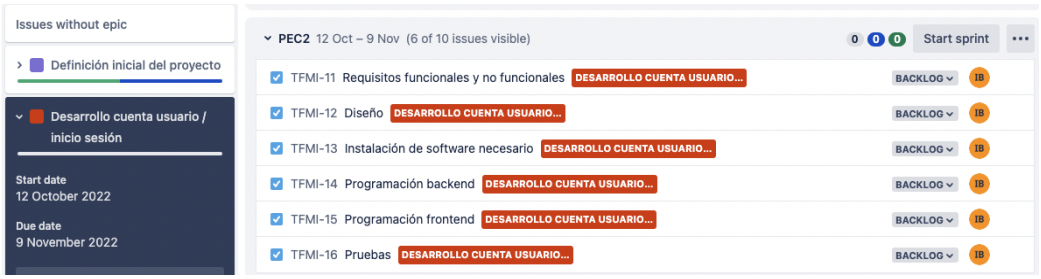



**Ilustración 3 - Hito 2, Pec2 - Desarrollo cuenta usuario / Inicio sesión**

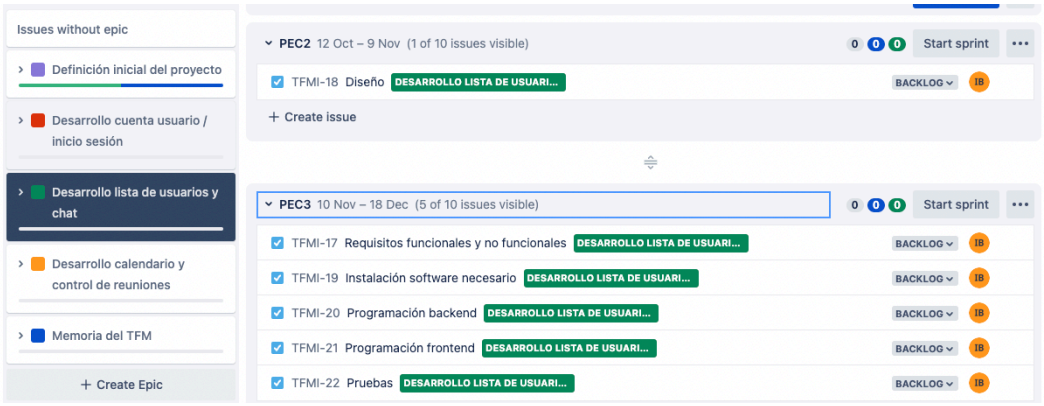



**Ilustración 4 - Hito 3, Pec2 y Pec3 - Desarrollo lista de usuarios y mensajería**

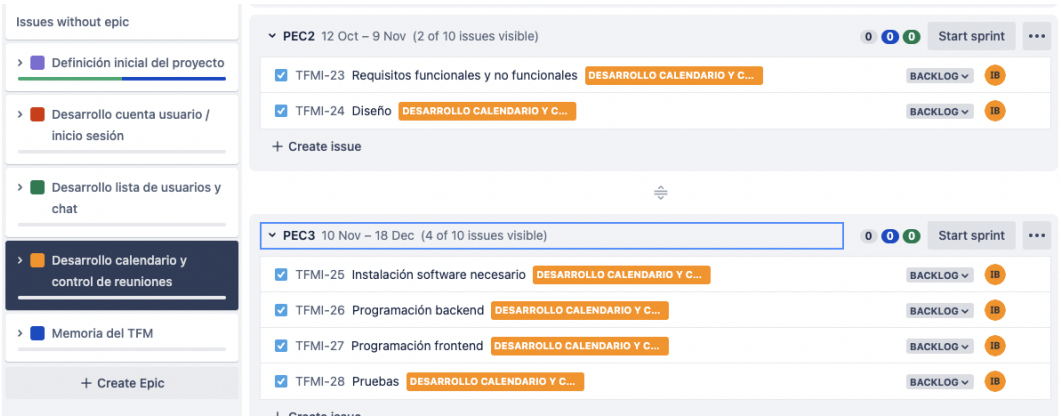



**Ilustración 5 - Hito 4, Pec2 y Pec3 - Desarrollo calendario y control de reuniones**

### 1.5.3. Diagrama de Gantt

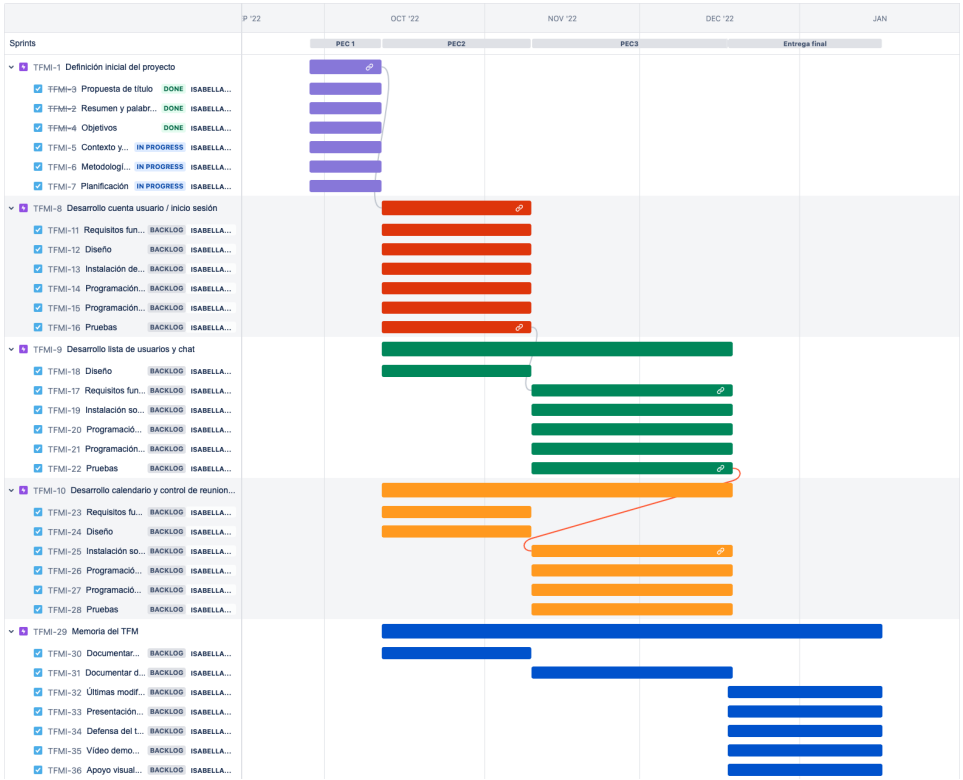



**Ilustración 6 - Diagrama de Gantt**

# 2. Presentación KeepTalking

## 2.1. ¿Qué es KeepTalking?

Es una aplicación web que permite a los usuarios registrarse y aprender idiomas. Se ofrecen intercambios de idiomas donde los usuarios se ponen en contacto a través de la aplicación con otras personas que quieren aprender la lengua materna de la contraparte para tener un encuentro informal, online o presencial y mejorar sus habilidades de comunicación.

## 2.2. Punto de partida de este proyecto

KeepTalking es una idea de negocio en construcción. En primera instancia se ofrecería como una herramienta gratuita para el público en general, a medida que gane popularidad y aumente el número de usuarios, se valorarán nuevas funcionalidades y una cuota de suscripción.

## 2.3. Funciones y características

La aplicación está pensada para que la use cualquier tipo de usuario. Se quiere llegar tanto a personas acostumbradas a las tecnologías, como a los que no han utilizado nunca un sitio web de idiomas y se está iniciando en el sector solo porque les interesa la temática que se va a exponer en este proyecto.

Las principales funciones que debe cubrir la aplicación son:

- Gestionar un sistema de usuarios con su respectivo “Inicio de sesión”, “Registro de usuario” y “Editar perfil”

- Mostrar un tablón con un listado de usuarios filtrable
- Mostrar la información acerca de cada usuario de la lista
- Servicio de mensajería instantánea entre usuarios
- Sistema de calendario donde gestionar reuniones

## 2.4. Usuarios destino

La aplicación está diseñada para que pueda usarla el mayor número de usuarios posible y en el mayor número de dispositivos distintos.

Por ellos, se ha realizado un diseño muy fácil e intuitivo basado en rellenar formularios y establecer conversaciones entre usuarios. Además, la aplicación cuenta con una interfaz completamente *responsive.* Personas de cualquier edad que quieran aprender o mejorar un idioma extranjero y que estén dispuestas a tener un encuentro informal con otros podrán hacer uso sin dificultades de la solución.

## 2.5. Propuesta de valor

### 2.5.1. Personalización

KeepTalking ofrecerá un servicio personalizado para cada usuario. Al registrarse en la plataforma, podrán crear un perfil personal único que incluya una descripción de sí mismos y que permita encontrar otros perfiles que coincidan con sus atributos y que se adapten a su demanda. Por ejemplo, un estudiante que habla inglés, pero quiere aprender alemán, podrá encontrar perfiles que hablen alemán y quieran aprender inglés, y que además se adapten a las características y gustos de cada uno. Es decir, si al alemán le gusta hablar de

música, podrá encontrar a angloparlantes a los que les guste la música y así aumentarán las posibilidades de que coincidan.

### 2.5.2. Encontrar un lugar de encuentro a través de la plataforma

Como se ha explicado anteriormente, KeepTalking no solo abarca el proceso de búsqueda de usuarios y la posterior vinculación entre ellos. De hecho, su alcance va más allá. Uno de los problemas que plantean las reuniones presenciales entre "desconocidos" es encontrar un lugar que se adapte a los participantes del encuentro y que sea idóneo para la actividad que se va a desarrollar. Además, otras razones como la privacidad de los usuarios o la necesidad de disponer de una serie de herramientas para el correcto desarrollo de la reunión (por ejemplo, es posible que se requiera una impresora, WIFI, recursos didácticos, silencio o ambiente de trabajo, etc.), pueden ser factores muy importantes a la hora de encontrar un sitio que se ajuste a las necesidades del encuentro. En este sentido, y con el objetivo de garantizar que el producto cubra todos estos deseos, la plataforma ofrece también la posibilidad de agendar reuniones virtuales. Por ejemplo, si el alemán y el inglés pretenden llevar a cabo una sesión de conversación, podrán reunirse el día y a la hora que deseen.

### 2.5.3. Sistema de mensajería

Para realizar la comunicación con los otros usuarios y poder entablar conversaciones didácticas o de organización de encuentros, KeepTalking les proporciona a los alumnos un sistema de mensajería instantánea. Con esto, podrán mantener el contacto durante todo el día y no tendrán que enviar

teléfonos, correos ni ningún dato no vinculado con la aplicación, ya que el chat estará vinculado a su perfil.

### 2.5.4. Diseño del producto

Por último, teniendo en cuenta que en el mercado actual ya existen aplicaciones bien establecidas que cubren este segmento, es necesario crear una interfaz perfecta, sencilla y que funcione sin fallos, para que el usuario pueda tener una experiencia completa y agradable a través de la plataforma.

# 3. Análisis del mercado y viabilidad

Es esencial entender el mercado en el que operará KeepTalking y saber si el mercado es amplio y hay espacio suficiente para que se posicione en un buen lugar. Conocer a los consumidores es importante, pero conocer a los competidores es clave para el rendimiento de la empresa y ser capaz de seguir creciendo a largo plazo.

En este apartado se realizará una descripción del estado actual de la industria y un análisis más profundo, tratando de cubrir los aspectos anteriores, utilizando la siguiente estructura: se realizará una perspectiva de la industria, luego se realizará un análisis de la competencia.

## 3.1. Perspectivas de la industria

La necesidad de aprender otros idiomas es cada vez más plausible. Nos relacionamos constantemente con personas de diferentes nacionalidades, ya que las comunidades se mezclan con personas de diferentes orígenes. Un gran porcentaje de personas mayores todavía no se han puesto al día en esta materia, pero las generaciones más jóvenes están cada vez más predispuestas a aprender otros idiomas. Además, la globalización también está brindando nuevas oportunidades a jóvenes que sepan hablar idiomas y que apuestan por viajar al extranjero a continuar sus carreras laborales.

El mercado en el que operará KeepTalking es muy amplio e incluye actores muy diversos; desde academias hasta libros de texto, pasando por cursos de

inmersión o aprendizaje online, todos ellos compiten por atender una misma necesidad, el aprendizaje de idiomas. En los últimos años, con la implantación de las nuevas tecnologías y el consiguiente giro del mercado hacia un enfoque más digitalizado, las empresas más tradicionales están viendo cómo pierden cuota de mercado y el consiguiente florecimiento del aprendizaje online, que se espera que crezca a un "CAGR3 del 18,7% de 2020 a 2027 hasta alcanzar los 21.200 millones de dólares en 2027" (Global Opportunity Analysis and Industry Forecast (2020-2027) | Meticulous Market Research Pvt. Ltd., 2020). También hay otros factores que han hecho que el mercado sea potencialmente más atractivo, como la inclinación por los empleados bilingües o multilingües por parte de los reclutadores de las grandes empresas, la integración de otros idiomas en nuestro sistema educativo o el continuo contacto con extranjeros cuando viajamos o incluso en nuestras actividades cotidianas. Además, la crisis de Covid-19 también ha potenciado un entorno de aprendizaje online.

En cuanto a las aplicaciones de aprendizaje de idiomas, la profesora de la UOC Laia Canals, experta en la materia, señaló en 2020 que se estimaba un total de 2.000 en el mercado mundial, y que el número de usuarios activos se incrementó en un 125% al inicio de la cuarentena en 2020 [2].

### 3.2. Análisis de la competencia

Considerando todo el mercado de aprendizaje de idiomas a través de apps o páginas web, Tandem [3] es el principal líder del mercado. Se basa en chatear con otros usuarios online, con una amplia oferta de diferentes idiomas y si se desea, se pueden contratar tutores personales. Otras aplicaciones con el mismo

perfil online son Babbel [4], Hellotalk [5] o Busuu [6]. Como se ha mencionado anteriormente, la mayoría de ellas se centran en el aprendizaje online.

Al centrarse en las plataformas que ofrecen no solo reuniones en línea entre pares, sino también la posibilidad de interacciones en persona, se observa que la concentración del mercado es claramente menor. Una solución con estas características es Polyglot [7]. Esta plataforma de 1 millón de usuarios consiste en una comunidad en la que cualquier persona puede tener una experiencia de intercambio de idiomas. El principal problema es que no hay una aplicación móvil disponible, solo una página web. La misma situación ocurre con Meetspeaker, que tiene, en comparación con Polyglot, una interfaz más sencilla. Las dos tienen aplicaciones web completamente funcionales y aun así usan tecnologías anticuadas, como por ejemplo Polyglot que su código frontend se basa en una plantilla de Bootstrap y jQuery. Así, KeepTalking es la única opción que podría ofrecer tanto una aplicación web con un diseño completamente *responsive,* actualizada y utilizando las últimas tecnologías para adentrarse en este nicho.

En el siguiente gráfico se muestra una comparación entre los principales competidores y KeepTalking trazada en un mapa de posicionamiento. Para elaborar este mapa se han elegido dos variables: “Actividades en persona u online”, definida como el grado de actividad presencial y los recursos disponibles para este tipo de reuniones; y “Digitalización y virtualidad”, que califica la optimización de la aplicación y/o el sitio web, sus tecnologías, y lo bien que funcionan según los comentarios de los usuarios.

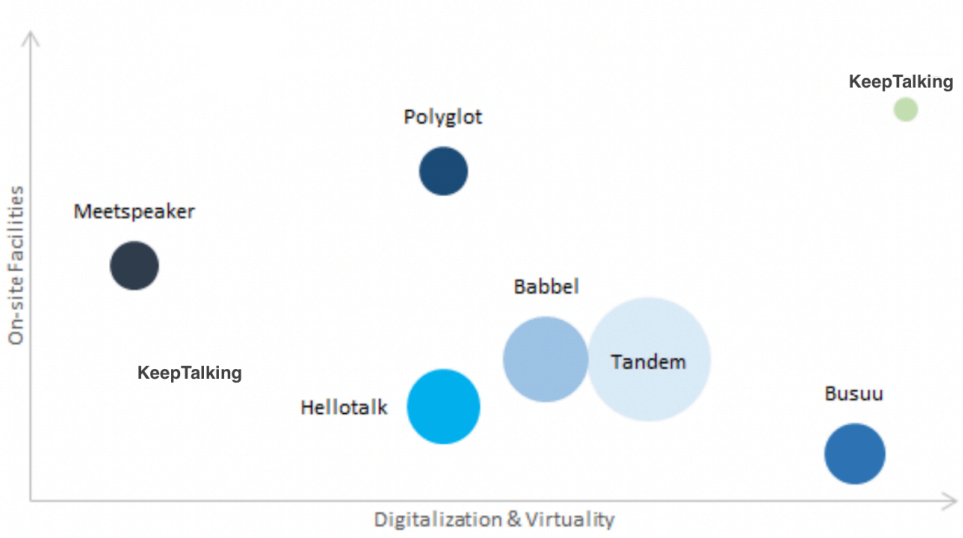



**Ilustración 7 - Mapa de posicionamiento de KeepTalking y competencia**

Por lo tanto, es importante destacar la importancia que KeepTalking da a las interacciones entre los usuarios y que, tras completar el desarrollo de toda la aplicación y futuras mejoras, KeepTalking quedaría posicionada como líder en el nicho de los intercambios de idiomas.

# 4. Catálogo de requisitos

A continuación, se describen los diferentes requisitos de los que consta la aplicación desarrollada en este proyecto.

## 4.1. Requisitos no funcionales

- ***Responsive*:** El sistema debe tener una interfaz preparada para que se pueda ejecutar en cualquier tipo de dispositivo.
- **Rendimiento:** La aplicación debe funcionar fluidamente, independientemente del número de usuarios y de la carga de datos.
- **Tiempo de respuesta:** El sistema debe tener unos tiempos de respuesta cortos para que la navegación sea fluida y el usuario tenga buena experiencia.
- **Mantenibilidad**: La aplicación debe ser fácilmente mantenible.
- **Github**: Portal para alojar y controlar las versiones del proyecto.
- **Backend**: Estará desarrollado en Laravel.
- **Frontend:** Estará desarrollado en ReactJS.
- **Base de datos**: Se creará una base de datos con MySQL.
- **Docker**: La aplicación debe estar desplegada en un entorno local con Docker.
- **NodeJS:** Se empleará para instalar las dependencias.

## 4.2. Requisitos funcionales

Cada requisito funcional será identificado por un código para posteriormente ser relacionado con los casos de uso correspondientes.

- **Registro de usuario (RQ01):** La aplicación debe ser capaz de registrar un usuario nuevo y almacenarlo en la base de datos.
- **Inicio de sesión (RQ02):** El sistema debe ser capaz de crear una sesión para cada usuario obteniendo su correo electrónico y su contraseña, comprobando que existe un usuario en base de datos con esas credenciales.
- **Perfil usuario - Edición (RQ03):** El sistema debe ser capaz de ofrecer la posibilidad de cambiar los datos personales con los que se registró en el sistema.
- **Tablón de usuarios con buscador (RQ04):** La aplicación debe mostrar un listado de todos los usuarios almacenados en base de datos que cumplan con las condiciones del usuario registrado.
- **Pantalla de mensajería - Chat (RQ05):** La aplicación debe permitir la comunicación entre usuarios de manera instantánea.
- **Calendario para organizar reuniones (RQ06):** El sistema debe tener la posibilidad de gestionar reuniones entre usuarios.

# 5. Análisis del sistema

## 5.1. Casos de uso

Cada caso de uso será identificado por un código para relacionarlo con los requisitos funcionales y con el prototipo de la aplicación.

- **Caso de uso: (CU01)** Registro **-** Este caso de uso aborda el requerimiento funcional **RQ01**, de registro de usuario.
- **Descripción:** Registro de un nuevo usuario.
- **Actores:** Usuario.
- **Resumen**: El usuario introduce sus datos en la aplicación y genera unas credenciales con las que poder iniciar sesión.
- **Precondiciones**: El usuario no puede tener una sesión iniciada y no puede introducir una dirección de correo que ya exista en la base de datos.
- **Postcondiciones**: Se almacenan los datos del usuario en la base de datos.
- **Escenario principal:**
    - El usuario hace clic en el botón "*Register*".
    - Aparece la pantalla de registro en la que hay un formulario.
    - El sistema valida que todos los campos estén correctamente rellenos.
    - Se almacenan los datos del usuario y base de datos.
    - Se redirige a la pantalla de inicio de sesión.

- **Escenarios alternativos:**
    - El usuario puede cerrar la aplicación en cualquier momento.
    - Algún campo del formulario no pasa la validación y se muestra el mensaje correspondiente.
    - El correo ya existe y aparece el mensaje correspondiente.

- **Caso de uso**: **(CU02)** Iniciar sesión **-** Este caso de uso aborda el requerimiento **RQ02,** de inicio de sesión.
- **Descripción**: El usuario inicia sesión en la aplicación.
- **Actores**: Usuario.
- **Resumen**: El usuario inicia sesión en la aplicación usando las credenciales que ha obtenido previamente en el registro.
- **Precondiciones**: El usuario debe estar registrado previamente.
- **Postcondiciones**: El usuario inicia sesión y puede acceder a la aplicación.
- **Escenario principal:**
    - El usuario pulsa "*Login*".
    - Aparece la pantalla de inicio de sesión.
    - El usuario introduce sus credenciales.
    - El sistema valida las credenciales.
    - El sistema crea la sesión y redirige a la pantalla inicial.
- **Escenarios alternativos:**
    - El usuario puede cerrar la aplicación en cualquier momento.
    - Algún campo del formulario no pasa las validaciones y se muestra el mensaje adecuado.

    - Las credenciales introducidas no son correctas y se muestra un mensaje.

- **Caso de uso**: **(CU03)** Editar perfil **-** Este caso de uso aborda el requerimiento **RQ03**, perfil de usuario y edición.

- **Descripción**: Editar los datos con los que el usuario se ha registrado en el sistema.
- **Actores**: Usuario.
- **Resumen**: El usuario accede a su perfil y edita los datos personales.
- **Precondiciones**: El usuario debe estar registrado previamente y con la sesión iniciada.
- **Postcondiciones**: Los nuevos datos del usuario quedan registrados en la base de datos.
- **Escenario principal:**
    - El usuario ha iniciado sesión y hace clic en "*My profile*".
    - Aparece el formulario con los datos introducidos en el registro.
    - El usuario modifica los datos que quiere editar y los guarda.
    - Se almacenan los nuevos datos.
- **Escenarios alternativos:**
    - El usuario puede cerrar la aplicación en cualquier momento.
    - Algún campo del formulario no pasa la validación y se muestra un mensaje de error.

- **Caso de uso**: **(CU04)** Entablar conversación **-** Este caso de uso aborda los requerimientos **RQ04** y **RQ05,** ya que para entablar una conversación el usuario debe revisar el tablón de usuarios e iniciar el chat.

- **Descripción**: Comenzar una conversación con otro usuario por mensajería instantánea.
- **Actores**: Usuario.
- **Resumen**: El usuario inicia una conversación con otro usuario.
- **Precondiciones**: El usuario debe estar registrado previamente y con la sesión iniciada.
- **Postcondiciones**: La conversación fluye fácilmente entre los dos usuarios.
- **Escenario principal:**
    - El usuario ha iniciado sesión y revisa el listado de usuarios.
    - Da clic en la opción “Let's chat” de otro usuario.
    - Aparece el chat en pantalla y establecen una conversación.
- **Escenarios alternativos:**
    - El usuario puede cerrar la aplicación en cualquier momento.
    - El otro usuario no contesta a sus mensajes.

- **Caso de uso**: **(CU05)** Agendar reunión **-** Este caso de uso aborda los requerimientos **RQ04** y **RQ06,** ya que para agendar una reunión el usuario debe revisar el tablón y elegir la opción de crear la reunión.

- **Descripción**: Agenda una reunión online o presencial con otro usuario.
- **Actores**: Usuario.

- **Resumen**: El usuario programa una reunión con otro usuario para seguir la conversación del chat o profundizar en los conocimientos.
- **Precondiciones**: El usuario debe estar registrado previamente y con la sesión iniciada.
- **Postcondiciones**: La reunión queda registrada adecuadamente en el calendario de ambos.
- **Escenario principal:**
    - El usuario ha iniciado sesión.
    - Da clic en la opción “Calendar” y selecciona un día o en el *“Dashboard”* al botón “Create meet”.
    - Se abre la modal de crear reunión con un formulario.
- **Escenarios alternativos:**
    - El usuario puede cerrar la aplicación en cualquier momento.

## 5.2. Diagrama de casos de uso


**Ilustración 8 - Diagrama de casos de uso**

## 5.3. Modelo conceptual de datos

A continuación, se muestra el modelo conceptual de datos del sistema. Es un modelo relacional en el que se encuentran las relaciones entre las distintas entidades.

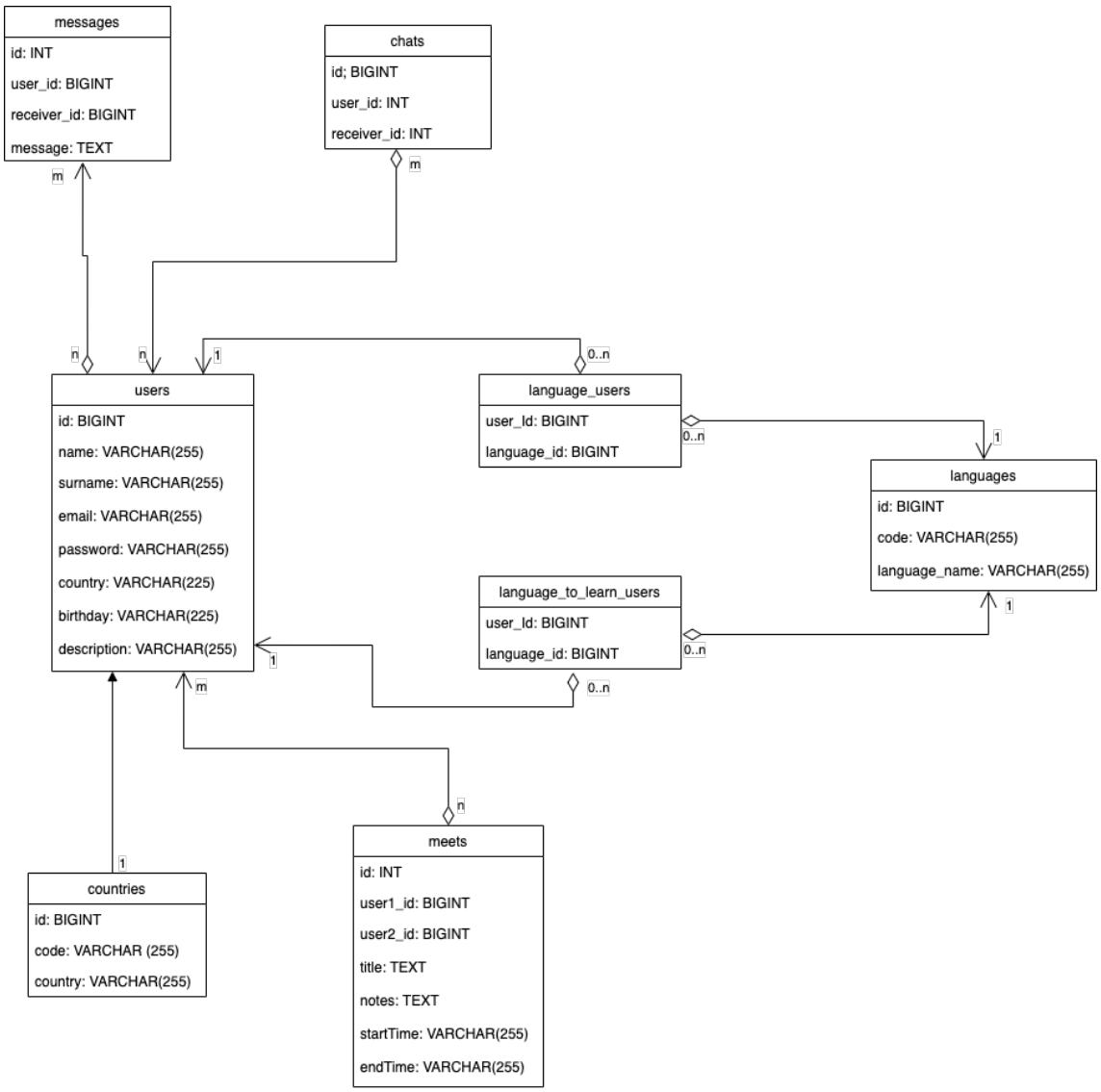



**Ilustración 9 - Modelo conceptual de datos**

Detalladamente se explica en que consiste cada entidad y sus atributos:

- **users**: Gestiona los datos de los usuarios registrados en la aplicación y sus credenciales para iniciar la sesión. Los atributos que contiene son:
    - **ID**: Identificador único
    - **NAME**: Nombre del usuario
    - **SURNAME**: Apellidos del usuario
    - **EMAIL**: Correo electrónico del usuario
    - **PASSWORD**: Contraseña para iniciar sesión, se almacenada en base de datos codificada
    - **COUNTRY**: País donde reside el usuario
    - **BIRTHDAY**: Fecha de nacimiento del usuario
    - **DESCRIPTION:** Descripción del usuario
- **languages:** Contiene todos los idiomas que el usuario puede seleccionar. Los atributos son:
    - **ID**: Identificador único
    - **CODE**: Código de dos letras para identificar cada idioma
    - **LANGUAGE_NAME**: Nombre del idioma
- **language_users**: Tabla *pivot* entre los idiomas y los usuarios. Relaciona los usuarios con los idiomas que habla. Atributos:
    - **USER_ID:** Identificador del usuario. Clave foránea
    - **LANGUAGE_ID**: Identificador del idioma. Clave foránea
- **language_to_learn_users**: Tabla *pivot* entre los idiomas y los usuarios. Relaciona los usuarios con los idiomas que quiere aprender. Atributos:
    - **USER_ID:** Identificador del usuario. Clave foránea
    - **LANGUAGE_ID**: Identificador del idioma. Clave foránea

- **countries**: Contiene todos los países que el usuario puede seleccionar. Los atributos son:
    - **ID**: Identificador único
    - **CODE**: Código de dos letras para identificar cada país.
    - **COUNTRY**: Nombre del país
- **meets**: Contiene todas las reuniones agendadas entre los usuarios. Atributos:
    - **ID**: Identificador único de cada reunión
    - **USER1_ID**: Identificador único de uno de los dos usuarios
    - **USER2_ID**: Identificador único de uno de los dos usuarios
    - **TITLE**: Título del evento
    - **NOTES**: Notas importantes del evento
    - **STARTTIME**: Día y hora a la que empieza la reunión
    - **ENDTIME**: Día y hora a la que acaba la reunión
- **messages**: Contiene todos los mensajes que se envían los usuarios. Los atributos son:
    - **ID**: Identificador único de cada mensaje
    - **USER_ID:** Usuario que envía el mensaje
    - **RECEIVER_ID:** Usuario que recibe el mensaje
    - **MESSAGE:** Texto del mensaje que ha sido enviado
- **chats**: Contiene todos los chats que han sido iniciados entre los usuarios. Los atributos son:
    - **ID**: Identificador único de cada chat
    - **USER_ID**: Usuario que participa en el chat
    - **RECEIVER_ID**: Segundo usuario que participa en el chat

# 6. Prototipos

En esta sección se muestra el diseño de la apariencia que se quiere conseguir en el producto final de la aplicación.

## 6.1. Logo

El uso de color como técnica de estudio. Rojo: Es un color que el cerebro memoriza con facilidad, ya que lo percibe como una llamada de alerta.


**Ilustración 10 - Logotipo de la aplicación**

## 6.2. Diseño

### 6.2.1. Landing page y pequeña introducción

Con estas páginas se quiere conseguir que el usuario se sienta atraído y quiera utilizar la aplicación.

Se ha diseñado una página principal sencilla, con un eslogan que llama la atención y una introducción con los tres puntos más importantes de la aplicación.

En la parte superior de ambas páginas se encuentran los dos botones que permiten al usuario registrarse (*Sign up*) o iniciar sesión (*Sign in*).

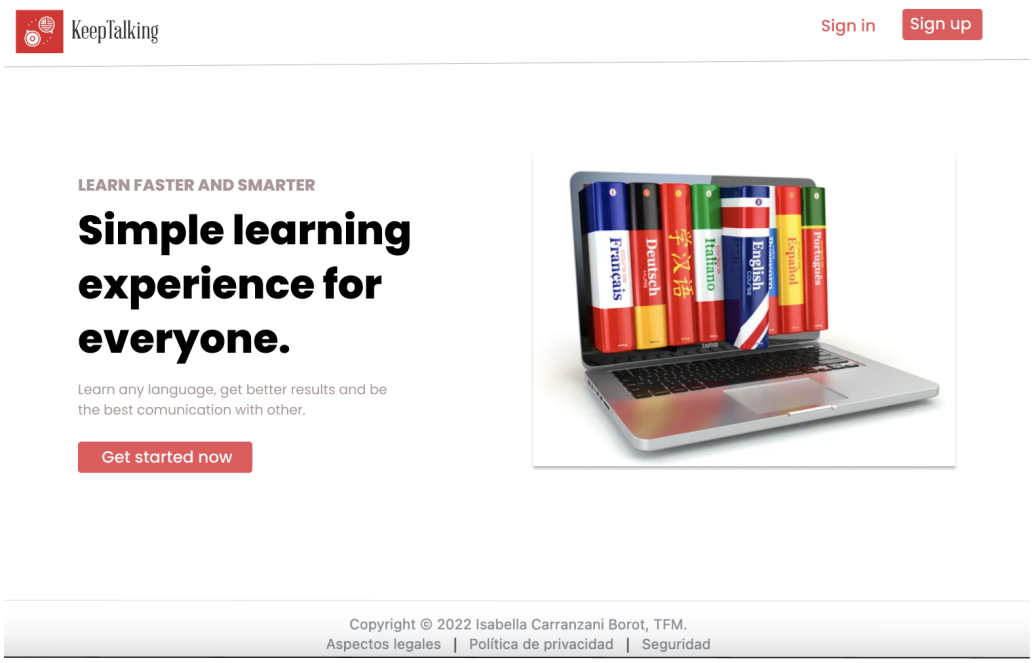



**Ilustración 11 - Prototipo en Figma de la Landing Page**

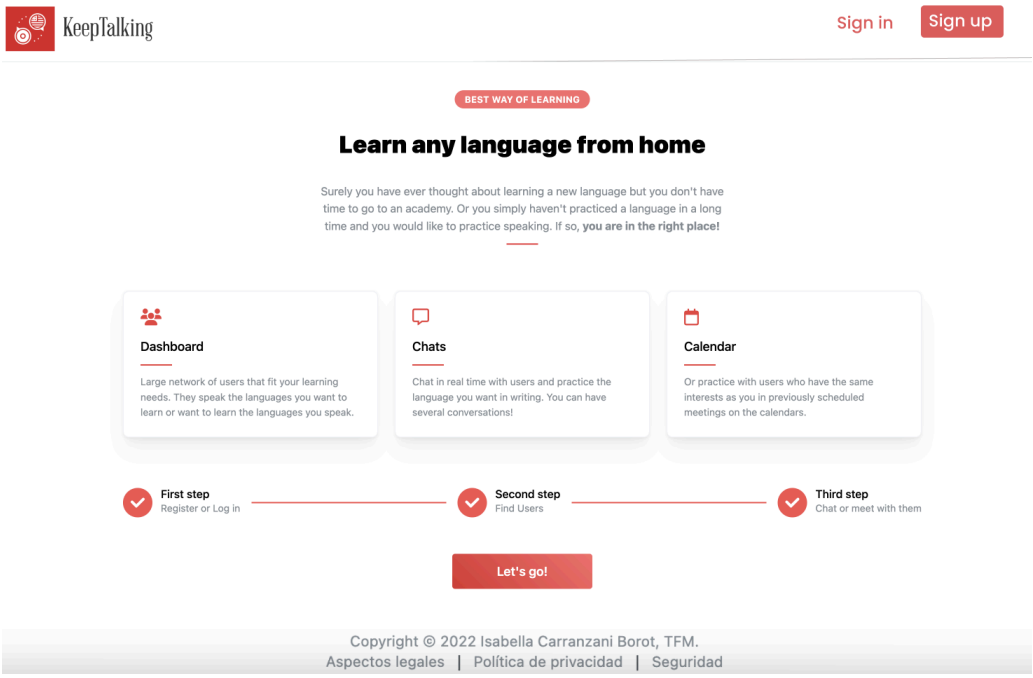



**Ilustración 12 - Prototipo en Figma de la introducción**

### 6.2.2. Registro de usuario

Con este diseño se da solución al caso de uso CU01 y se cumple el requisito funcional RQ01.

El diseño del registro consta de un formulario, un botón para validar y enviar los datos y un enlace que permite al usuario cambiar a la página de inicio de sesión en caso de que ya se haya registrado en la plataforma.

Los valores requeridos para el registro se introducen en diez *inputs* y son los siguientes:

Tipo texto

- Dirección de correo electrónico (*email)*
- Nombre del usuario
- Apellido del usuario
- Contraseña
- Confirmación de la contraseña
- Descripción del usuario

Tipo selector y selector múltiple

- País
- Idiomas que habla el usuario (máximo cuatro)
- Idiomas que quiere aprender el usuario (máximo cuatro)

Tipo fecha

- Fecha de nacimiento

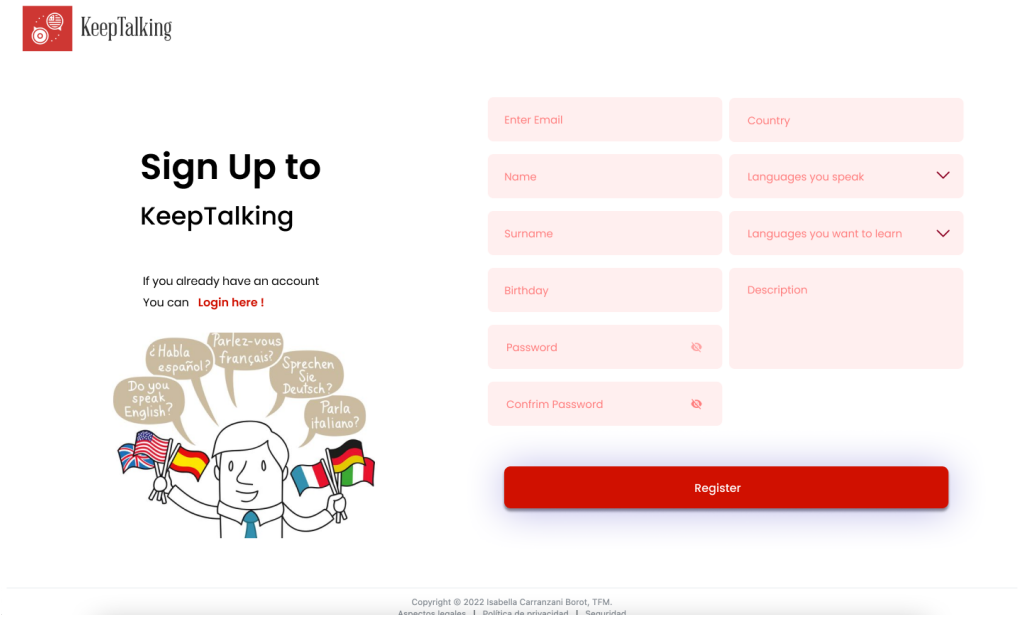



**Ilustración 13 - Prototipo en Figma del Registro**

### 6.2.3. Inicio de sesión

Con este diseño se da solución al caso de uso CU02 y se cumplen los requisitos funcionales RQ01 y RQ02, ya que para poder iniciar sesión el usuario debe haberse registrado previamente.

El diseño de inicio de sesión consta de un formulario, un botón para validar y enviar los datos y un enlace que permite al usuario cambiar a la página de registro en caso de que no se haya registrado en la plataforma.

Los valores requeridos para el inicio de sesión se introducen en dos *inputs* y son los siguientes:

Tipo texto

- Dirección de correo electrónico (*email)*
- Contraseña

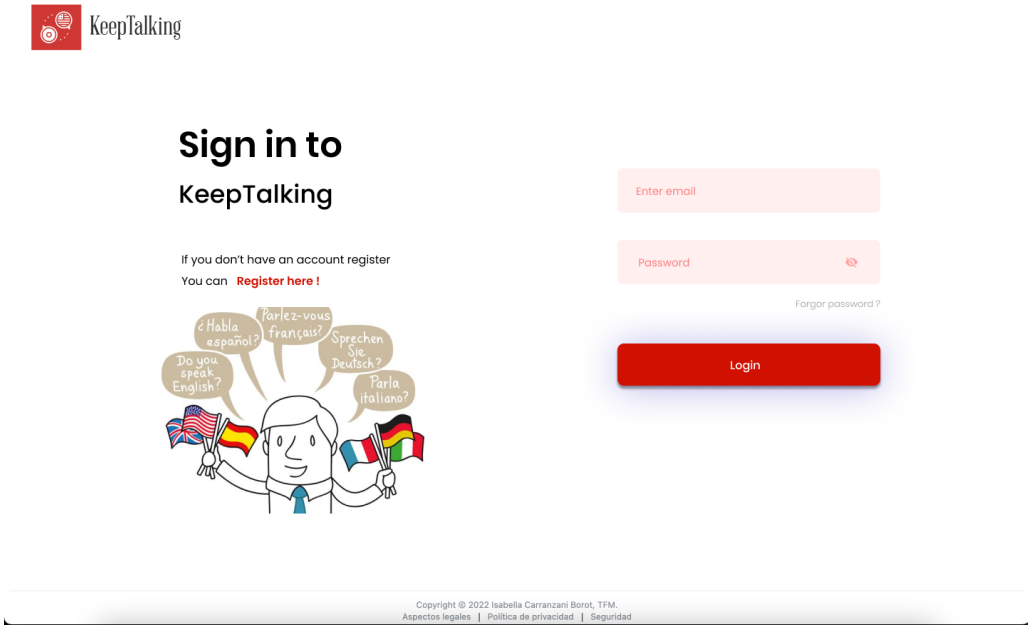



**Ilustración 14 - Prototipo en Figma del Login**

### 6.2.4. Perfil de usuario

Con este diseño se da solución al caso de uso CU03 y se cumple el requisito funcional RQ03 y RQ02, porque el usuario puede editar su perfil si ha iniciado sesión.

El diseño del perfil de usuario es muy parecido al diseño de registro, consta de un formulario, un botón para validar y actualizar los datos. Además, en el encabezado ya se encuentran todas las páginas disponibles para los usuarios que se han conectado.

Los valores que se pueden actualizar, exceptuando el correo electrónico, se introducen en nueve *inputs* y son los siguientes:

Tipo texto

- Nombre del usuario

- Apellido del usuario
- Contraseña
- Confirmación de la contraseña
- Descripción del usuario

Tipo selector y selector múltiple

- País
- Idiomas que habla el usuario (máximo cuatro)
- Idiomas que quiere aprender el usuario (máximo cuatro)

Tipo fecha

- Fecha de nacimiento

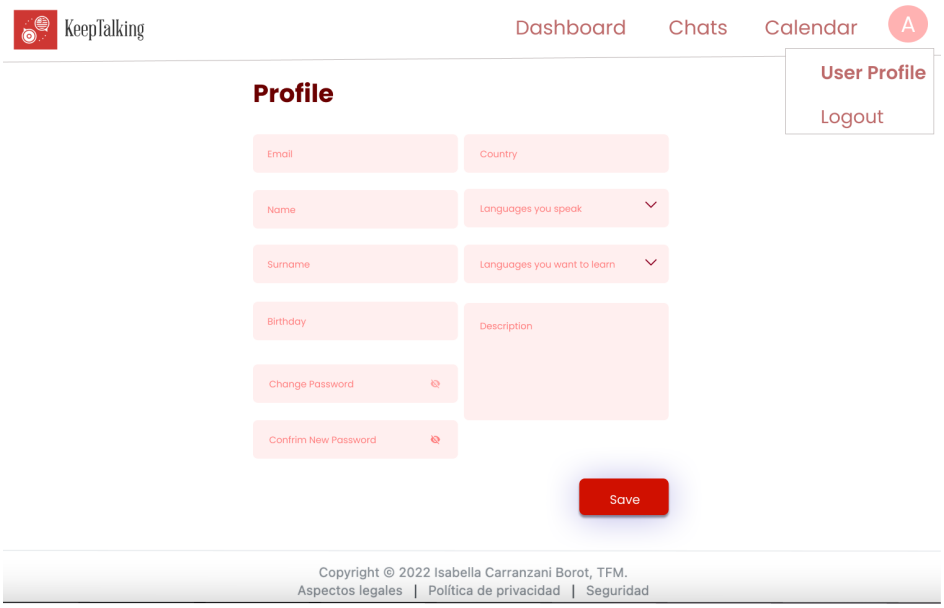



**Ilustración 15 - Prototipo en Figma del perfil de usuario**

### 6.2.5. Página principal / Dashboard

Con este diseño se cumple el requisito funcional RQ04, donde se encuentra un listado de las personas que coincide con la configuración de idiomas que tiene el usuario conectado, es decir, que hablan algún idioma de los que quiere estudiar y que quieran aprender algún idioma que habla el usuario.

El diseño de la página principal consta de un filtro de búsqueda de usuario por nombre. El listado contiene cajas con la información de cada uno, cada caja tiene los idiomas que habla y quiere aprender, además del nombre y la descripción. En la parte inferior hay dos botones que permiten iniciar una conversación o fijar una reunión en el calendario.

En el encabezado, como en el diseño anterior, se encuentran los enlaces a las otras páginas de la web.

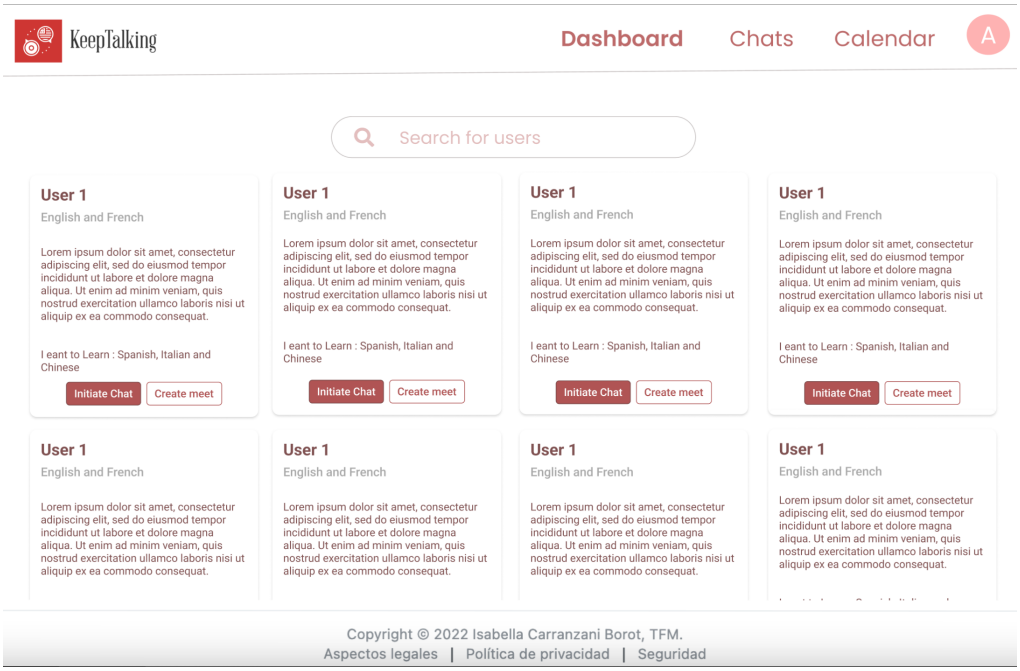



**Ilustración 16 - Prototipo en Figma de la página principal**

### 6.2.6. Chats

Con este diseño se da solución al caso de uso CU04 y se cumple el requisito funcional RQ05.

El diseño de los chats tiene dos páginas:

**1 -** Listado de chats activos. Esta pantalla contiene un listado con las conversaciones que el usuario conectado tiene iniciadas. Cada fila de la lista tiene el nombre del usuario con el que se está conversando y la opción de eliminar el chat. Además, existe la posibilidad de eliminar todas las conversaciones con el botón en la parte superior de la lista.

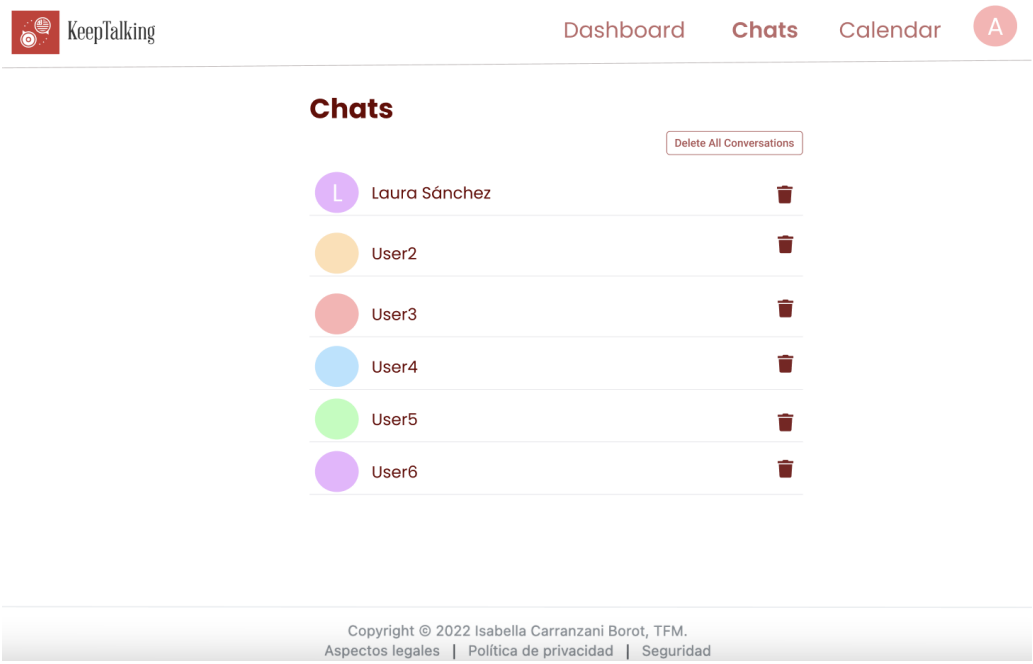



**Ilustración 17 - Prototipo en Figma del listado de conversaciones (chats)**

**2 -** Pantalla de la conversación. Este diseño es una pantalla de chat en tiempo real. Contiene el nombre de usuario de la persona con la que se entabla la conversación y los mensajes que se están enviando.

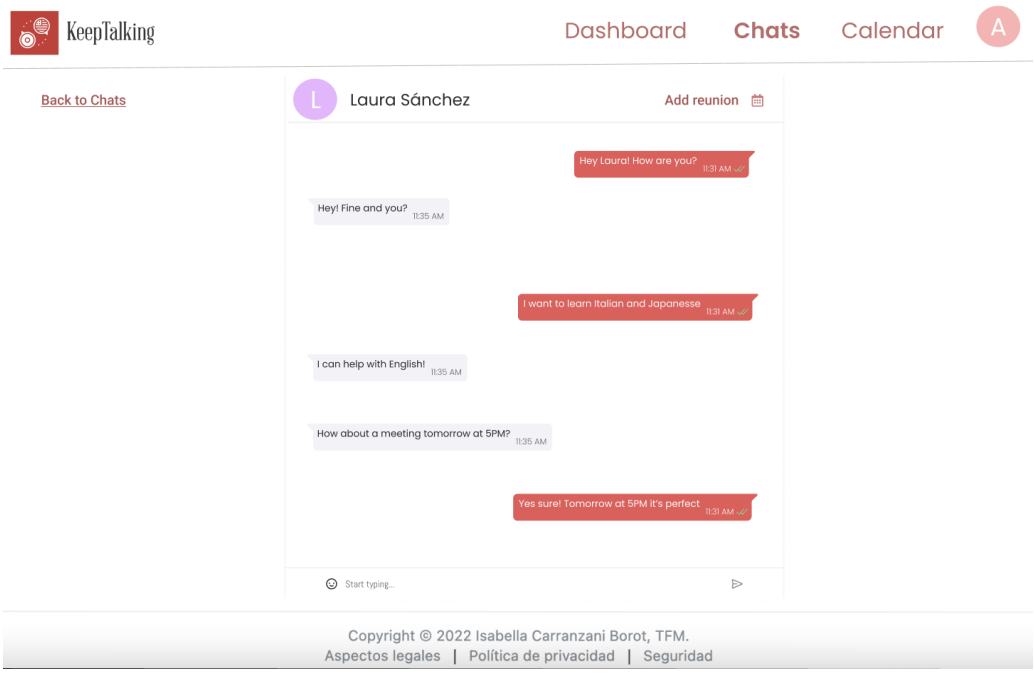



**Ilustración 18 - Prototipo en Figma de una conversación (chat)**

### 6.2.7. Calendario

Con este diseño se da solución al caso de uso CU05 y se cumple el requisito funcional RQ06.

El diseño consta de un gran calendario con las reuniones ya agendadas, un botón para añadir una nueva reunión y otro botón para eliminar la que se desee.

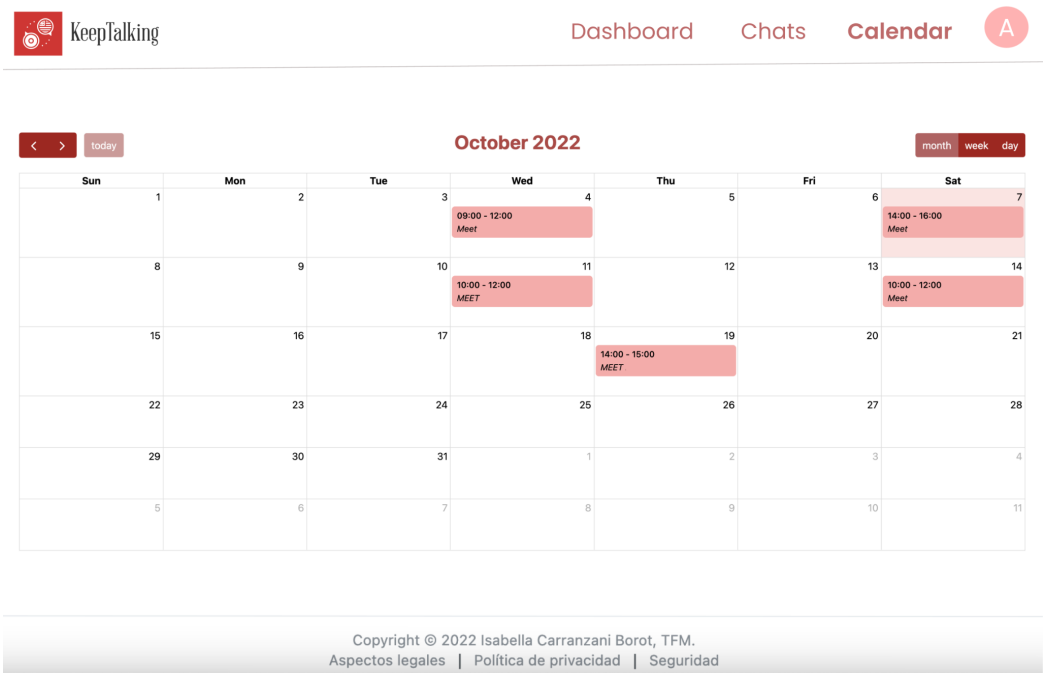



**Ilustración 19 - Prototipo en Figma del calendario y sus opciones**

# 7. Usabilidad/UX

A la hora de realizar el diseño de la aplicación, se han tenido en cuenta diferentes patrones que ayudan a que la aplicación tenga una interfaz intuitiva y amigable para el usuario.

Al estar dirigida para un gran número de usuarios de todas las edades, la interfaz ha de contar con un diseño sencillo. Así como la estructura de la aplicación ha de ser lo más simple posible para que el usuario pueda acceder a cualquier pantalla de la aplicación y ejecutar la acción que desea.

Para ello, se han usado diferentes patrones de diseño que facilitan el uso y proporcionan una buena experiencia al usuario.[8]

- **Morphing Controls**: Con cambios en los diseños de los botones, se le muestra al usuario si está disponible para pulsarse o no. Como, por ejemplo, en los botones del menú en el encabezado, si está disponible para pulsar cambian de color y si está seleccionado tiene un color diferente.

- **Table Filter**: Se ofrece al usuario un mecanismo de filtrado de datos. Por ejemplo, como se ve en el diseño en la página principal, se añaden filtros para poder encontrar los usuarios lo más rápido posible.

- **HomeLink**: Se muestra, en todo momento, en el lugar fijo del encabezado a la izquierda, un enlace que lleva al usuario a la página principal, en este caso es el Logo de la aplicación.

- **Input feedback:** En la página de login y registro se muestra información al usuario sobre si está rellenando los datos correctamente.

- **Account registration**: Se intenta pedir el menor número de datos a los usuarios para que registrarse o iniciar sesión sea lo más amigable y rápido posible.

- **Call to action:** Los botones de inicio de sesión, registro y de la *landing page* son bastante visibles, con un color destacado y están muy cerca del contenido relevante.

- **Cards**: En el tablón de usuarios se usan *cards* para distribuir los datos. Es un patrón de interfaz modular que es muy útil para una amplia variedad de pantallas y resoluciones.

También, se han tenido en cuenta los patrones para adaptar el diseño a diferentes tamaños de pantallas. Para ello se ha seguido una estrategia de diseño *DesktopFirst*, por lo que se han diseñado primero los prototipos para pantalla de gran formato y posteriormente se ha hecho responsive.

# 8. Tecnologías

En esta sección se especifican las diferentes herramientas de software utilizadas para el desarrollo de este TF.

## 8.1. Entorno de desarrollo

### 8.1.1. Creación de las aplicaciones *frontend* y *backend*

Para hacer el proyecto se ha decidido crear las aplicaciones *frontend* y *backend* en React y Laravel, dentro de un mismo directorio y repositorio GitHub, para tener el control de versiones actualizado en todo momento y poder ir construyendo las dos aplicaciones en paralelo.

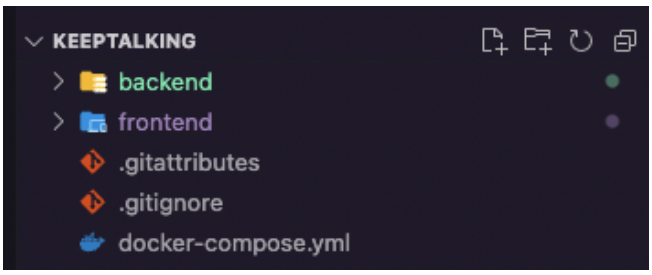



**Ilustración 20 - Estructura del proyecto**

**Los beneficios de desarrollar el frontend en ReactJs y TypeScript:**

- Es intuitivo para trabajar, además, permite un desarrollo de aplicaciones rápido y de calidad.
- Otorga a los desarrolladores la posibilidad de reutilizar los componentes creados. Esto garantiza un muy buen rendimiento de desarrollo.
- React usa una combinación de JavaScript y HTML, lo que simplifica la adaptación del desarrollador y la escritura de código.

- Facilita el mantenimiento de la aplicación y la lectura de los componentes, ya que las vistas se pueden desglosar en varios elementos, creando archivos con pocas líneas de código.
- TypeScript y los “types” promueven la calidad del código, facilita la lectura y comprensión. Además, agiliza el *“refactoring”* del código porque es el compilador el que encuentra los errores.

**Los beneficios de desarrollar el backend en Laravel:**

- Laravel otorga gran seguridad a las aplicaciones web, porque no permite que ninguna amenaza de seguridad ingrese a las mismas.
- Facilita la creación de aplicaciones que proporcionan acceso solamente a los usuarios autorizados.
- Tiene las migraciones de base de datos. Permiten deshacer o hacer cambios de manera intuitiva.
- Es una plataforma de código abierto con mucha documentación y una comunidad de soporte actualizada.

### 8.1.2. Docker

Se ha escogido Docker como contenedor porque acelera el desarrollo facilitando un entorno donde probar el código.

**Beneficios de utilizar Docker:**

- Permite empaquetar la aplicación de una manera estándar y garantiza que el software funcionará siempre igual, independientemente del entorno donde se esté ejecutando.

- Permite ejecutar contenedores en cualquier infraestructura (Linux, Windows, MacOs).
- Los contenedores aíslan aplicaciones y de la infraestructura subyacente añadiendo una capa de protección a la aplicación.
- Es muy ligero, ya que los contenedores se ejecutan en una máquina compartiendo el kernel del sistema operativo. Eso proporciona un arranque casi instantáneo.

**Instalación y configuración de Docker en el proyecto**

En el caso de este proyecto se instaló Docker en el escritorio del ordenador con el instalador *Docker.dmg* [9].

Una vez instalado en el escritorio, se creó en la raíz de cada aplicación, *backend* y *frontend*, un *DockerFile* para la creación de una nueva imagen. Y en la raíz del proyecto se debe agregar el archivo *docker-compose.yml,* este define todos los servicios que se van a implementar y que dependen de las imágenes anteriormente generadas.

- **DockerFile frontend:**

En este archivo se crea la imagen de la aplicación frontend. Se especifica el servidor dónde estará alojada, el directorio dónde se encuentra la aplicación, el puerto y los comandos necesarios para ejecutarla.

```
FROM node:alpine AS builder

WORKDIR /app/frontend

COPY . .

RUN npm install

EXPOSE 3000
CMD npm start
```

**Ilustración 21 - Captura del DockerFile frontend**

- **Dockerfile backend:**

Este DockerFile tiene el mismo objetivo que el otro, pero para la aplicación backend. En este caso, se instala composer en Docker y se ejecuta *composer install* para resolver e instalar las dependencias.

```
FROM php:8.1-fpm-alpine
RUN docker-php-ext-install pdo pdo_mysql sockets
RUN curl -sS https://getcomposer.org/installer | php -- \
    --install-dir=/usr/local/bin --filename=composer
COPY --from=composer:latest /usr/bin/composer /usr/bin/composer
WORKDIR /app/backend
COPY . .
RUN composer install
```

**Ilustración 22 - Captura del DockerFile backend**

- **Docker-compose.yml:**

En este archivo se añaden los servicios que se utilizarán para este proyecto.

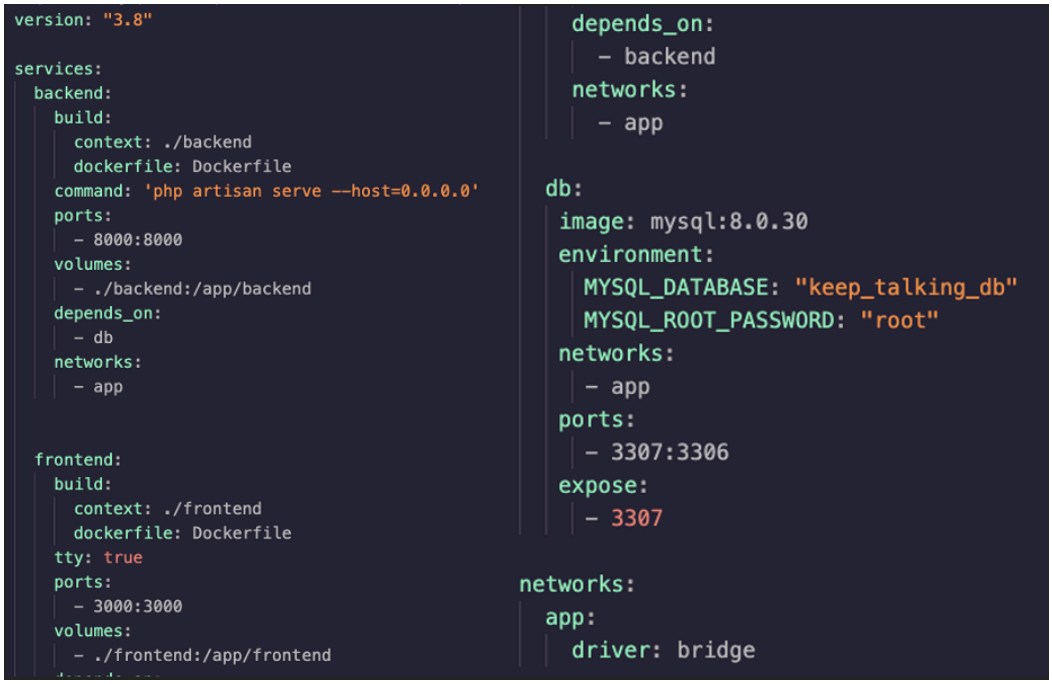
En este caso se crean los siguientes servicios:

- **Servicio backend**, donde se especifica el lugar en el que se encuentra en DockerFile, en que puerto estará desplegado en local, en este caso 8000. Además, se especifica que el backend depende

de la base de datos, por lo tanto, si el servicio de esta no está funcionando, el backend tampoco funcionará.

- **Servicio frontend**, es exactamente igual que el servicio backend, la diferencia es que en este caso el puerto es el 3000.

- **Servicio db,** este es el servicio de la base de datos, se debe especificar cuál será la base de datos, para esta aplicación es MySQL, el nombre, la contraseña del usuario root y el puerto donde estará expuesta, en este caso 3007.



```yaml
version: "3.8"

services:
  backend:
    build:
      context: ./backend
      dockerfile: Dockerfile
    command: 'php artisan serve --host=0.0.0.0'
    ports:
      - 8000:8000
    volumes:
      - ./backend:/app/backend
    depends_on:
      - db
    networks:
      - app


  frontend:
    build:
      context: ./frontend
      dockerfile: Dockerfile
    tty: true
    ports:
      - 3000:3000
    volumes:
      - ./frontend:/app/frontend
    depends_on:
      - backend
    networks:
      - app

  db:
    image: mysql:8.0.30
    environment:
      MYSQL_DATABASE: "keep_talking_db"
      MYSQL_ROOT_PASSWORD: "root"
    networks:
      - app
    ports:
      - 3307:3306
    expose:
      - 3307

networks:
  app:
    driver: bridge
```

**Ilustración 23 - Captura del docker-compose.yml**

Una vez creados los tres archivos de configuración de Docker se ejecuta *docker compose* y se crean los contenedores configurados anteriormente, ya listos para desplegar la aplicación.

- La aplicación frontend se encuentra en la ruta **localhost:3000**

- La aplicación backend se encuentra en la ruta **localhost:8000**
- La base de datos está expuesta en el puerto **3007**

Para gestionar la base de datos se usa la plataforma pública Workbench, ya que es libre y permite conectar a la base de datos del servidor.

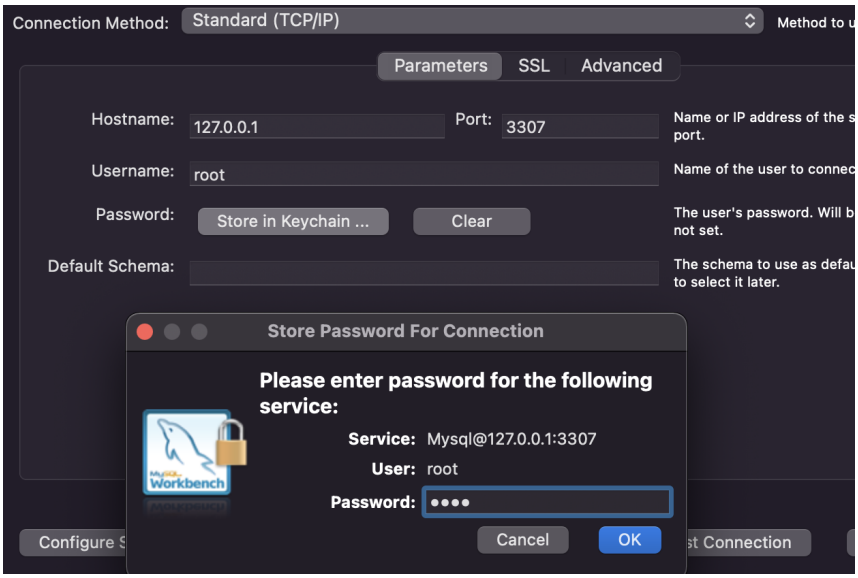



**Ilustración 24 - Captura de la conexión de Workbench con la base de datos.**

### 8.1.3. Librerías y herramientas externas

Para el desarrollo frontend se ha utilizado la librería de componentes React, **Mantine** [10]. Es una librería relativamente nueva, puesto que se lanzó en enero de 2021, sin embargo, se han publicado más de 100 versiones y tiene una gran comunidad de soporte. Esta librería está centrada en proporcionar una gran experiencia al usuario y al desarrollador, por ello se ha elegido para aprovechar todo el potencial que puede proporcionar.

```
~/TFM/KeepTalking/frontend > npm install @mantine/core @mantine/hooks @mantine/form @emotion/react
```

**Ilustración 25 - Captura de la instalación de la librería Mantine**

Asimismo, se instala el paquete ***react-router-dom*** para gestionar la actualización de las URL y la navegación dentro de la web [11], permitiendo un renderizado de la UI más rápido.

Para los iconos de la aplicación se ha instalado el paquete para React de la librería de iconos **FontAwesome** [12]. Es la librería más conocida por los desarrolladores y tiene millones de iconos muy útiles para las aplicaciones web.

Por último, para facilitar el desarrollo y el entendimiento del código, se ha instalado y configurado ***eslint*** y ***prettier***. Su función es analizar el código de la aplicación, detectar problemas y si es posible, poder resolverlos él mismo. Por otro lado, **Prettier** es un formateador de código que corrige el formato del código al deseado.

```
"devDependencies": {
  "eslint": "^8.8.0",
  "eslint-config-prettier": "^8.3.0",
  "eslint-import-resolver-typescript": "^2.5.0",
  "eslint-plugin-import": "^2.25.4",
  "eslint-plugin-import-helpers": "^1.2.1",
  "eslint-plugin-jsx-a11y": "^6.5.1",
  "eslint-plugin-prettier": "^4.0.0",
  "eslint-plugin-react": "^7.28.0",
  "eslint-plugin-react-hooks": "^4.3.0",
  "eslint-plugin-unused-imports": "^2.0.0"
},
```

**Ilustración 26 - Captura de pantalla de como quedan las nuevas dependencias**

Para el backend se instaló ***jwt-auth***, un estándar para la creación de tokens de acceso que permiten la propagación de identidad y privilegios, necesario para el desarrollo del *Login* y el *Registro* de la aplicación, una vez que el usuario inicia sesión, la respuesta devuelve un token [13].

Para los websockets que se utilizan para el desarrollo del chat, se instaló **Pusher** [14]. Es un servicio online que encapsula la implementación de Websockets permitiendo desarrollar una aplicación sin necesidad de tener que ejecutar un servidor de Websockets propio. Implementa una capa de eventos abstracta que puede ser enganchada con cualquier cliente o servidor.

# 9. Proceso de desarrollo

Las primeras vistas que se desarrollaron han sido las únicas que son completamente independientes del backend. **Landing page** y **página de información** sobre la aplicación. A partir de estas dos páginas se han ido creando las posteriores vistas que conforman la aplicación KeepTalking.

## 9.1. Primera entrega

Para la primera entrega se han desarrollado las páginas de login, de registro y edición de perfil. El proceso ha sido el siguiente:

**Backend**

- Creación de los controladores y modelos de *User* y *Authentication*.
- Creación de las tablas de base de datos de *User*, *Countries*, *Languages* y las de relaciones entre usuarios e idiomas con una tabla pivot.
- Creación de los endpoints:
    - `'api/login'`
    - `'api/register'`
    - `'api/update'`
    - `'api/countries'`
    - `'api/languages'`

**Frontend**

- Creación de las vistas *Login*, *Register* y *User Profile*.

- Creación de los componentes *LoginForm*, *RegisterForm y ProfileForm*, donde se encuentran los formularios de cada vista y las llamadas a los *endpoints.*
- Por último, creación de los componentes *Header* y *Footer* para unificar el diseño de las vistas.

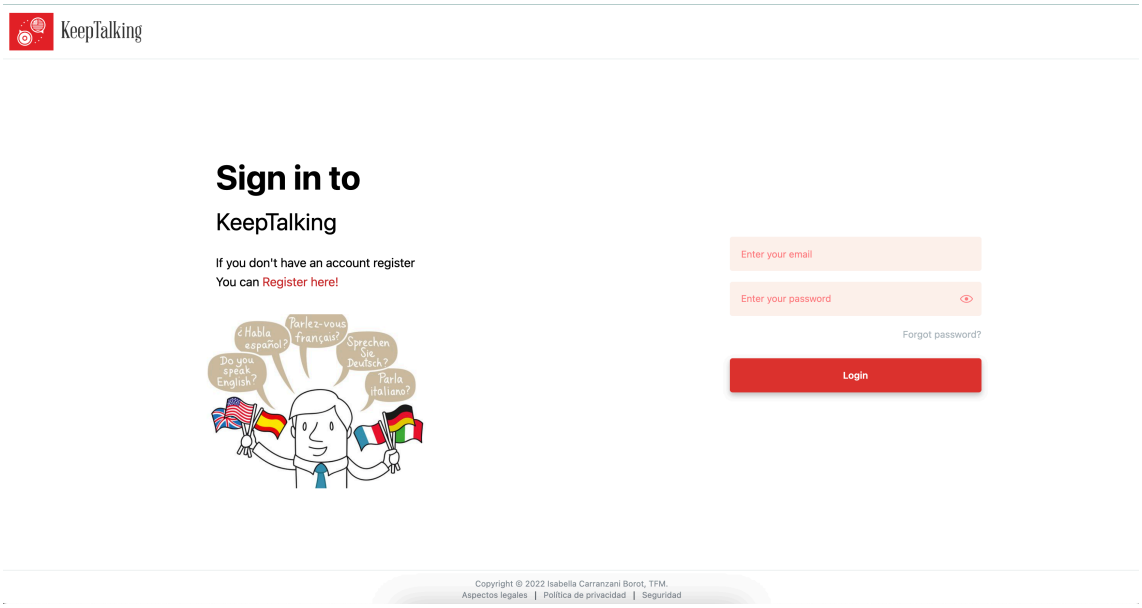



**Ilustración 27 - Login**

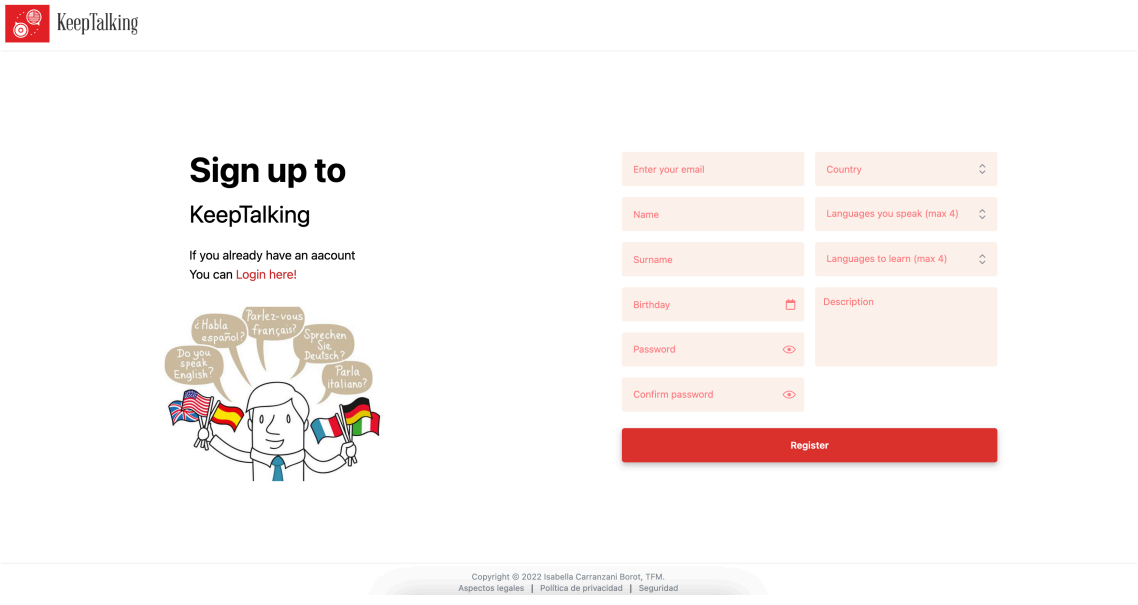


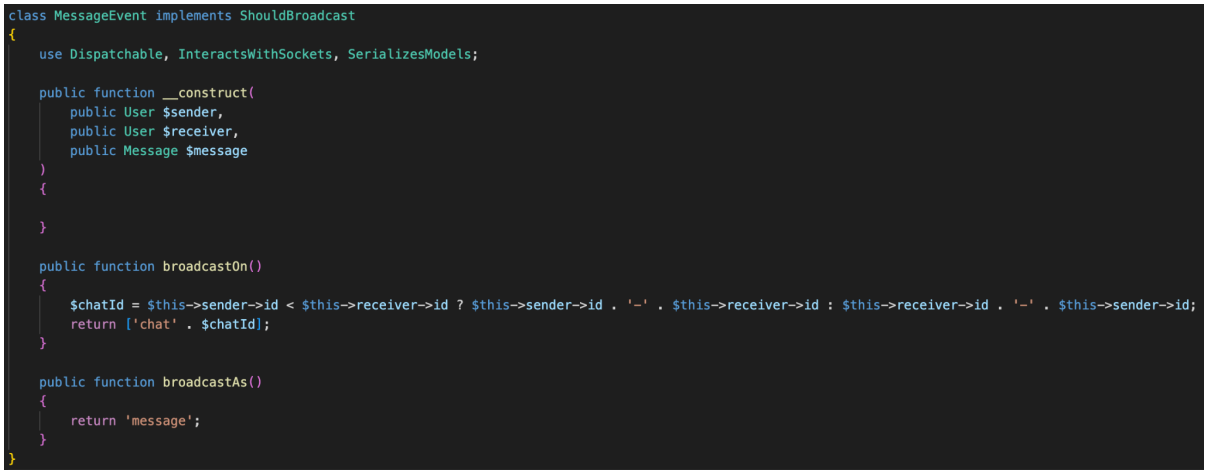

**Ilustración 28 - Registro**

## 9.2. Segunda entrega

En esta segunda entrega se ha desarrollado la página principal *(dashboard)*, las dos vistas de **chat** y la **lógica del chat**. El proceso ha sido el siguiente:

### Backend

- Instalación y configuración de la herramienta de Websocket en línea Pusher.
- Creación de los controladores y modelos de *Message* y *Chat*.
- Creación de las tablas de base de datos de *Messages*, *Chats* y la relación con los usuarios.
- Creación del evento *Message*, para emitir el mensaje en el chat adecuado según los usuarios que están interactuando.



```php
class MessageEvent implements ShouldBroadcast
{
    use Dispatchable, InteractsWithSockets, SerializesModels;

    public function __construct(
        public User $sender,
        public User $receiver,
        public Message $message
    )
    {

    }

    public function broadcastOn()
    {
        $chatId = $this->sender->id < $this->receiver->id ? $this->sender->id . '-' . $this->receiver->id : $this->receiver->id . '-' . $this->sender->id;
        return ['chat' . $chatId];
    }

    public function broadcastAs()
    {
        return 'message';
    }
}
```

**Ilustración 29 - Evento Message y transmisión con broadcast**

- Creación de los endpoints:
  - `'api/chats'`
  - `'api/delete_all'`
  - `'api/delete_chat'`

- `'api/get_messages'`
- `'api/messages'`

**Frontend**

- Instalación y configuración de la herramienta Pusher para inicializar los chats en el frontend.
- Creación del componente *SearchInput*, como barra de búsqueda para encontrar los usuarios por nombre en el dashboard.
- Creación de las vistas *MainPage, que contiene la lista de usuarios compatibles*, *Chats y ChatPanel*.
- Modificación del *Header* para unificar el diseño de las vistas.

En la página principal se ha hecho una pequeña modificación del diseño y se han añadido dos nuevos filtros para facilitar al usuario la búsqued. Los nuevos filtros son dos selectores que permiten al usuario buscar por idiomas que quiere aprender o por idiomas que quiere enseñar.

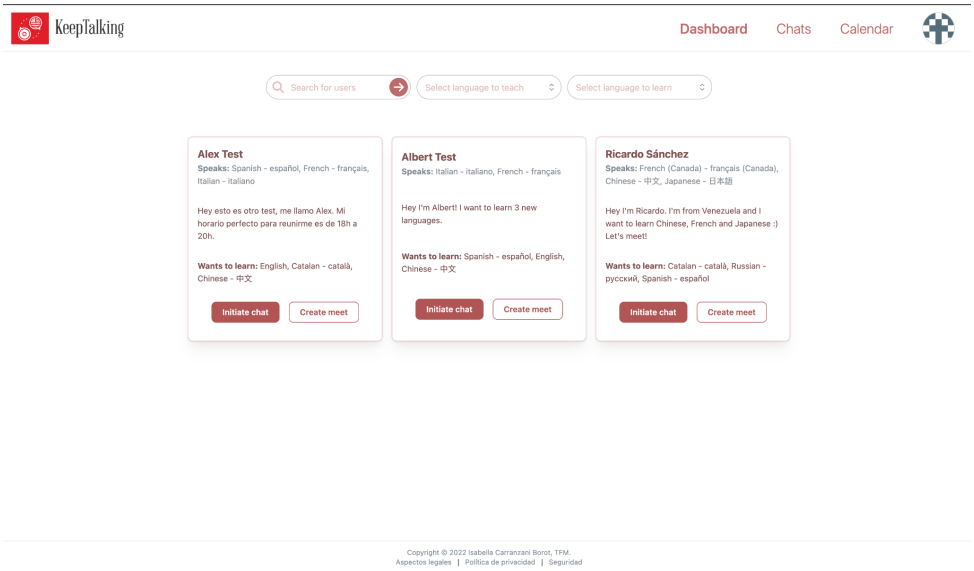



**Ilustración 30 - Página principal**

Además, se han añadido avatares para poder distinguir a los usuarios, para ello se usa la librería **DiceBear Avatar** [15]. Una librería muy útil que permite a partir de una semilla, en este caso, el identificador de usuario, obtener la URI de una imagen para añadir como avatar.

```
<Avatar
  src={createAvatar(style, {
    seed: id,
    dataUri: true,
  })}
```

**Ilustración 31 - Creación de avatar con DiceBear**

Por último, para el chat, como se ha comentado previamente, se utiliza el servidor de **WebSockets Pusher**. Un servicio externo, muy útil y de acceso gratuito que facilita el desarrollo del chat. En el backend, se añade la configuración de Pusher con la llave de acceso, entre otras opciones, que se obtienen una vez se ha creado una aplicación en la web de Pusher y en el frontend se inicializa y se suscribe al canal de cada usuario.

```
public function pusher(array $config)
{
    $pusher = new Pusher(
        $config['key'],
        $config['secret'],
        $config['app_id'],
        $config['options'] ?? [],
        isset($config['client_options']) && ! empty($config['client_options'])
                ? new GuzzleClient($config['client_options'])
                : null,
    );

    if ($config['log'] ?? false) {
        $pusher->setLogger($this->app->make(LoggerInterface::class));
    }

    return $pusher;
}
```

**Ilustración 32 - Inicialización Pusher en el backend**

Para que los chats sean *end-to-end* y no públicos, cada chat tiene un nombre por pareja de usuarios que se crea con los dos identificadores de los usuarios que han iniciado la conversación.

```
useEffect(() => {
  Pusher.logToConsole = true

  const pusher = new Pusher('ad83db98ed4402c50f5a', {
    cluster: 'eu',
  })

  const channel = pusher.subscribe(`chat${chatId}`)
  channel.bind('message', function (newMessage: any) {
    const msg = { ...newMessage.message, user: newMessage.sender }
    setMessages(current => [...current, msg])
  })
}, [data])
```

**Ilustración 33 - Inicialización y suscripción a Pusher en el frontend**

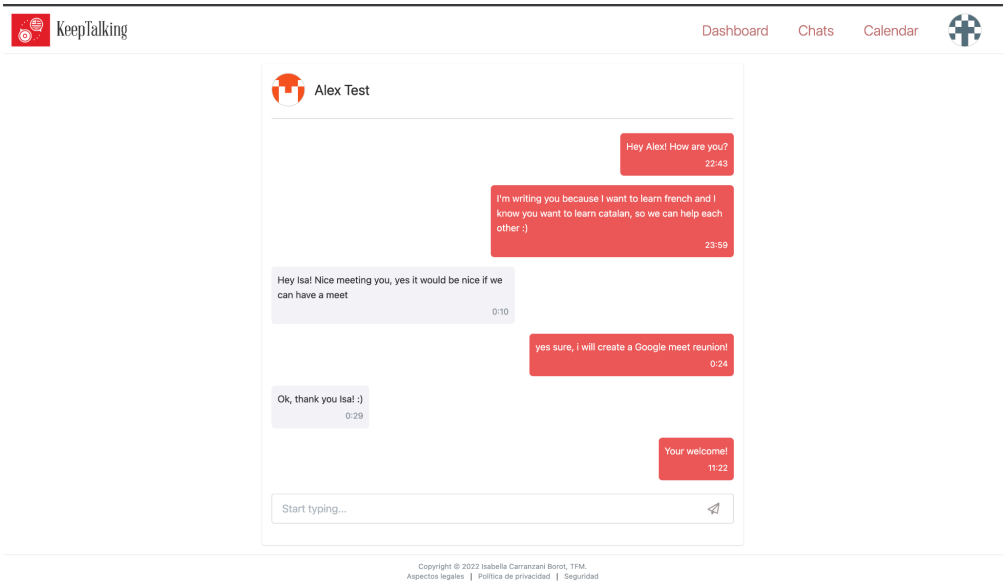



**Ilustración 34 - ChatPanel**

## 9.3. Tercera entrega

En la tercera entrega se ha añadido la modal con el formulario para crear eventos en las vistas *MainPage* y *Chat*. Además, se ha añadido la vista *Calendar,* la cual contiene el calendario del usuario con los eventos programados. El proceso ha sido el siguiente:

**Backend**

- Creación de los controladores y modelos de *Meet*.
- Creación de la tabla de base de datos *Meets* y la relación con los usuarios.

- Creación de los endpoints:
    - `'api/meets'`
    - `'api/meet'`
    - `'api/delete_meet'`
    - `'api/update_meet'`

**Frontend**

Para tener un calendario anual, semanal y diario con eventos se ha utilizado el componente **FullCalendar**.

- Creación del componente *CreateMeetModal* con un pequeño formulario para generar reuniones con los campos: *Usuario, Título, Notas, Fecha y hora* de la reunión.
- Creación de la vista *Calendar,* que contiene el calendario del usuario.
- Instalación y uso del componente FullCalendar en la vista *Calendar.*

En la vista *Calendar* se pueden llevar a cabo tres acciones con respecto a los eventos, añadir, editar y eliminar:

- Para añadir un evento hay que seleccionar el día en el calendario y automáticamente se abrirá la modal con el formulario para crear el evento.
- Para editar un evento hay que seleccionar uno ya existente y también, automáticamente, se abrirá la modal de edición con el formulario con la información del evento seleccionado.
- Por último, para eliminar un evento, es el mismo procedimiento que para editar, pero en la parte inferior izquierda de la modal de edición aparece un botón para anular.

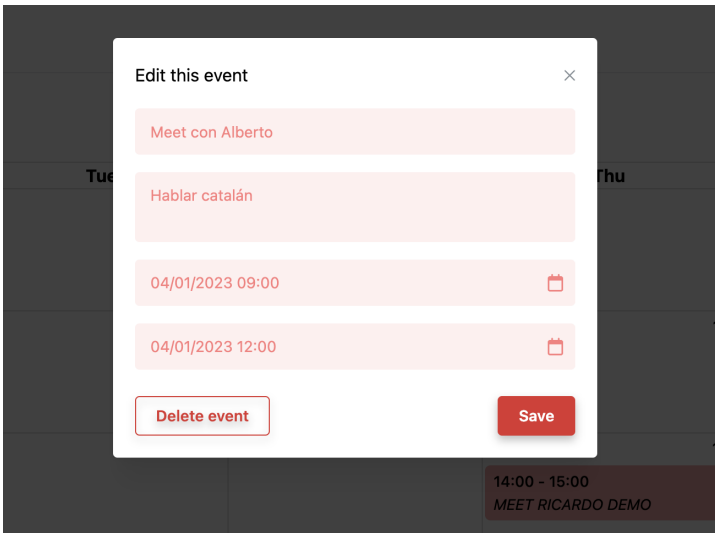



**Ilustración 35 - Modal de edición de evento**

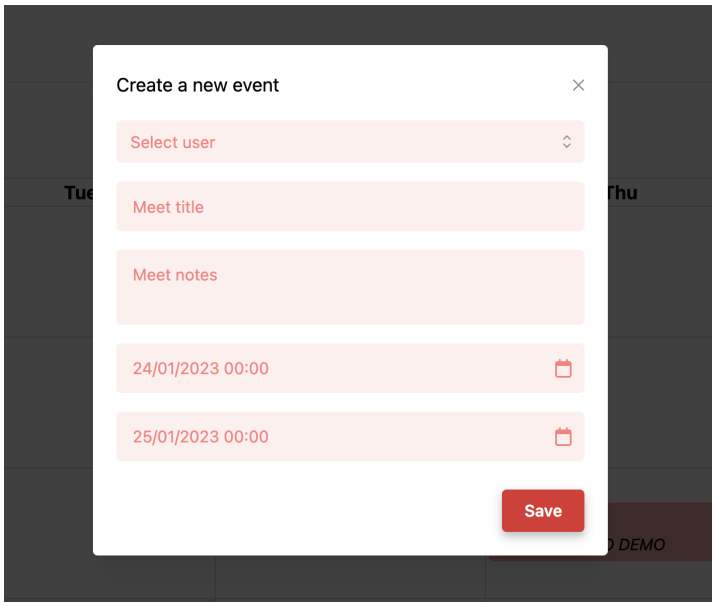



**Ilustración 36 - Modal de creación de eventos**

La librería **FullCalendar** [16] es una librería de JavaScript para generar interfaces de calendario. Es de código abierto y se puede adaptar para React y tiene compatibilidad con TypeScript. Además, también se ha elegido este componente porque ofrece un diseño y funcionalidades simples, que ayudan en la elaboración de la vista *Calendar*.

El único inconveniente que se encuentra a la hora de utilizar esta librería es que el calendario solo se puede estilar con estilos css. Como se puede ver en la imagen, los estilos se deben aplicar a los selectores de la librería y a las variables globales de FullCalendar.

```
.fc .fc-button-primary:not(:disabled).fc-button-active:active, .fc .fc-button-primary:not(:disabled):active, .fc .fc-button-primary:active{
  box-shadow: 0 0 0 0.2rem #ab161626;
  background-color: #ab1616;
  border-color: #ab1616;
}
.fc .fc-button-primary:disabled{
  background-color: #bb5f5f;
  border-color: #bb5f5f;
}
.fc-toolbar-title{
  color: #b64242;
}
.fc-direction-ltr .fc-daygrid-event.fc-event-end, .fc-direction-rtl .fc-daygrid-event.fc-event-start, .fc-h-event .fc-event-main, .fc-v-event{
  background-color: ${theme.colors.red[3]};
  border-color: ${theme.colors.red[3]};
  border-radius: 5px;
  padding: 4px 8px;
  color:  ${theme.black};
}`}
```

**Ilustración 37 - Ejemplo de los estilos de FullCalendar**

# 10. Trabajos futuros

Antes de empezar con el desarrollo del proyecto, se estudiaron las funcionalidades que se deseaban tener en el resultado final de la aplicación, pero su implementación, dependió del tiempo disponible y de las dificultades que se han ido encontrando a lo largo del desarrollo.

Finalmente, se ha conseguido un producto operativo y con las funcionalidades planteadas principalmente, pero, además durante el proceso han surgido bastantes ideas que pueden ser interesantes de cara a futuro, para que sea una aplicación muy completa.

## 10.1. Integración con Google

Este es el primer proyecto futuro que se añadiría a la aplicación, ya que aportaría mucho valor añadido si los usuarios pudieran iniciar sesión con su cuenta de Google y crear reuniones en línea con Google Meet.

## 10.2. Traducciones

Otra de las principales mejoras que se haría en un futuro cercano sería añadir traducciones y que la aplicación esté en varios idiomas para que todos los usuarios puedan hacer uso de ellos sin dificultades.

## 10.3. Sistema de valoración de usuarios

Podría ser interesante que cada usuario tuviera un sistema de valoración en el que cada uno que conecte con él/ella, pudiera dejar una valoración en forma de nota y que además se pudiera añadir un comentario. Esto ayudaría a los demás usuarios porque tendrían una referencia de alguien que ya ha conversado con la persona que está siendo valorada.

### 10.4. Personalización avanzada del perfil de usuario

Sería muy importante de cara al futuro que los usuarios pudieran tener un perfil más personalizado. Actualmente, al registrarse, el usuario puede añadir campos obligatorios para el correcto funcionamiento de la aplicación. Sin embargo, sería de gran ayuda para ellos poder añadir al perfil más detalles personales, como podrían ser un ejemplo:

- **Fotografía de perfil:** Podría servir para que los consumidores de la aplicación ganen confianza a la hora de entablar conversaciones con otras personas, ya que pueden ver un poco más de ellos.
- **Hobbies o intereses personales:** También para mejorar la confianza de las personas o para que las conversaciones sean más fluidas, añadir intereses personales puede ser un elemento primordial para que decidan si empezar a interactuar con esa persona o no.

### 10.5. Compartir archivos

En los chats, actualmente se pueden enviar mensajes de texto de cualquier largo, pero sería una mejora, poder enviar archivos de texto o incluso imágenes que tengan relación con la enseñanza del idioma que están hablando.

El envío de diferentes archivos puede ayudar a que los usuarios tengan más apoyo educativo para el aprendizaje.

### 10.6. Sistema de envío de emails

Desde luego esta es una mejora muy importante para diferentes casos de uso, desde la función de modificar contraseñas hasta notificar el registro o mensajes a los usuarios, así como *newsletters* o notificaciones que les ser de interés a los mismos.

### 10.7. Posibilidad de añadir usuario profesional

Sería muy interesante tener la opción de poder crear usuarios con rol "profesional", para que personas que sean profesores de idiomas puedan dar clases pagadas a través de la aplicación.

Por lo que generar este rol de usuario sería una gran mejora porque proporcionaría calidad a la aplicación y las personas que quisieran aprender podrían hacerlo con profesores oficiales.

# 11. Conclusión

El proyecto se ha basado en el desarrollo de una aplicación totalmente funcional orientada al aprendizaje de idiomas usando tecnologías que se han aprendido al cursar este máster.

Esta decisión se tomó por tres razones, la primera es que llevo más de dos años trabajando como desarrolladora *frontend*, pero hasta la fecha, no había hecho una aplicación web de este calibre desde cero. Desarrollando no solo el *frontend*, sino también el *backend.*

La segunda razón es que el máster de desarrollo de sitios y aplicaciones web se centra en el desarrollo *frontend,* y solo hay una asignatura dedicada al *backend.* Esto para el desarrollo laboral ha ido muy bien, no obstante, aún se podía mejorar y aprender sobre backend y por ello, el trabajo final ha aportado el contexto perfecto para motivar la mejora en este aspecto.

La tercera y última es que, tras barajar varias opciones y temáticas para el trabajo, hoy en día existen muchas academias y aplicaciones de idiomas, pero ninguna que pueda ofrecer charlas informales entre usuarios con intereses de aprendizajes comunes.

Por todo eso, se usaron tecnologías modernas para llevar a cabo y desarrollar esta aplicación. Nunca se había realizado un proyecto que utilizase Laravel para crear la API de una aplicación web. Ni tampoco se había diseñado una base de

datos desde cero. Con lo aprendido en el máster se ha podido investigar y se han incluido en este proyecto funcionalidades muy interesantes, obteniendo un resultado muy satisfactorio.

Como conclusión personal, considero el desarrollo de este trabajo un éxito tanto personal como profesional. En cuanto a los objetivos, se han cumplido todos como se habían planteado en un principio, puesto que he aprendido mucho sobre Laravel, React y MySQL. He aplicado la comunicación con *Websockets*, he utilizado herramientas como *Migrations* y *broadcast.* He creado mi propia API con varios *endpoints* que manipulan los datos de base de datos, he diseñado y he aprendido.

# 12. Bibliografía

A continuación, se detallan los principales recursos web consultados durante la realización de este trabajo:


1. Agile Alliance. *Definition of Done*. Agile Alliance. Visitada el 21 de noviembre de 2022: https://www.agilealliance.org/glossary/definition-of-done
2. Ponce de león, P. (2020, June 18). *La pandemia dispara el uso de aplicaciones de aprendizaje de idiomas*. UOC. Visitada el 16 de octubre de 2022: 277-Aprendizaje-idiomas-pandemia
3. Tandem. *Habla con Tandem*. Tandem Aplicación de Intercambio de Idiomas | Encuentra compañeros de aprendizaje de idiomas. Visitada el 16 de octubre de 2022: https://www.tandem.net/es
4. Babbel. *Language learning that works*. Babbel: Learn Spanish, French and Other Languages Online. Visitada el 16 de octubre de 2022: https://uk.babbel.com/
5. HelloTalk. *Talk to the World*. HelloTalk | Free Language Exchange | Language Partners. Visitada el 16 de octubre de 2022: https://www.hellotalk.com/
6. Busuu. Busuu - Aprender idiomas online: empezar a aprender gratis. Visitada el 16 de octubre de 2022: https://www.busuu.com/es
7. Polyglot Club. *Learn languages an make friends*. Polyglot Club Official Website - Practice languages and find friends. Visitada el 16 de cotubre de 2022: https://polyglotclub.com/

8. Meetspeaker. *Practise and teach languages*. MeetSpeaker: Dive into Languages. Visitada el 16 de octubre de 2022: MeetSpeaker

9. UI patterns. *Morphing Controls design pattern*. UI-Patterns.com. Visitada el 18 de diciembre de 2022: https://ui-patterns.com/patterns/morphing-controls
10. Docker. *Install Docker on Mac*. Install on Mac. Visitada el 11 de noviembre de 2022: https://docs.docker.com/desktop/install/mac-install/
11. Rtishchev, V. *Getting started*. Mantine. Visitada casi todos los días: https://mantine.dev/pages/getting-started/
12. ReactRouter. *Home | React Router*. React Router: Home v6.4.3. Visitada el 9 de noviembre de 2022: https://reactrouter.com/en/main
13. FontAwesome. *Set Up with React*. Font Awesome. Visitada el 9 de noviembre de 2022: https://fontawesome.com/docs/web/use-with/react/
14. JWT AUTH. *Home - jwt-auth*. jwt-auth: Home. Visitada el 10 de novimebre 2022: https://jwt-auth.readthedocs.io/en/develop/
15. MessageBird. *Pusher*. Pusher | Leader In Realtime Technologies. Visitada el 18 de diciembre de 2022: https://pusher.com/
16. bunny.net. *Avatars DiceBear*. DiceBear Avatars. Visitada el 18 de diciembre de 2022: https://avatars.dicebear.com/
17. FullCalendar LLC. 2019. “React Component - Docs.” FullCalendar. Visitada el 23 de diciembre de 2022: https://fullcalendar.io/docs/react.

# 13. Anexos

## 13.1. Anexo 1. Entregables del TFM

A lo largo de este trabajo final se han ido realizando varias entregas tanto de esta memoria como del código fuente de la aplicación.

Para almacenar el código de manera segura durante todo el proceso, se ha usado un repositorio Github, ahí se conservan todos los *commits* y versiones que se han ido aplicando. Teniendo en cuenta que en todo momento el código del repositorio era 100% funcional.

Se da acceso libre al repositorio con todo el código publicado desde el principio del desarrollo.
El enlace es el siguiente: https://github.com/Carranzani98/KeepTalking

## 13.2. Anexo 2. Guía de uso

La aplicación consta de un rol de usuario, el cual puede acceder a tres páginas con tres funcionalidades diferentes: buscar usuarios, chatear con ellos y agendar reuniones.
A continuación, se explican las funciones que pueden realizar los usuarios.

La primera pantalla que encontrarán los usuarios al acceder a la aplicación es la **Pantalla de inicio** desde donde se puede acceder a una página de información

de la web. Para acceder a la página de información se debe seleccionar el botón **“Get started”** (Está marcado con un ovalo rojo en la imagen inferior).

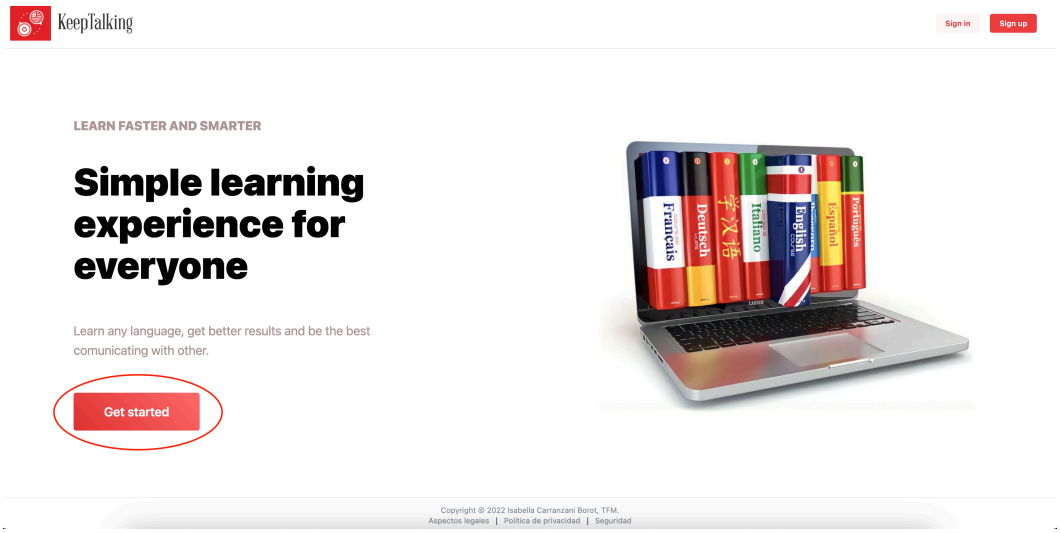



**Ilustración 38 - Guía de usuario, Landing Page**

Una vez en esta página, el usuario puede conocer mejor de que se trata esta aplicación y que funcionalidades tiene. Para acceder a la aplicación, es necesario estar registrado, por lo que, para ir a la página de registro, el usuario tiene tres opciones: En el botón **“Sing up”** que se encuentra en la cabecera marcado en rojo para registrarse o en el botón **“Sing in”** que se encuentra en la cabecera para ir a la página de inicio de sesión o en el botón **“Let’s go”** en la parte inferior de la página accediendo al inicio de sesión.

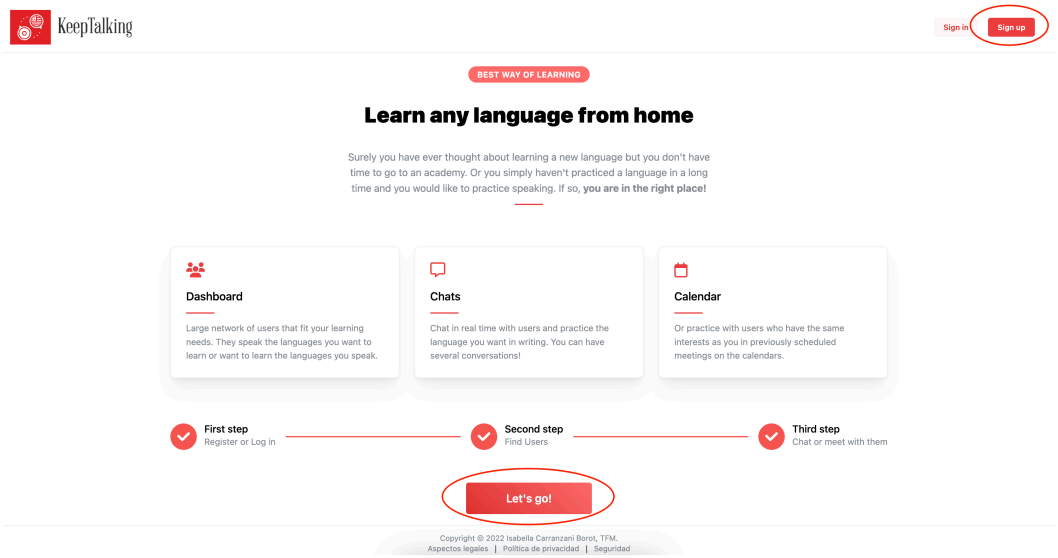



**Ilustración 39 - Guía de usuario, Página de información**

Una vez en la página de registro, el usuario debe rellenar los campos requeridos y dar al botón **"Register"** que se encuentra en la parte inferior, está marcado en rojo, para poder ir a la página de inicio de sesión.

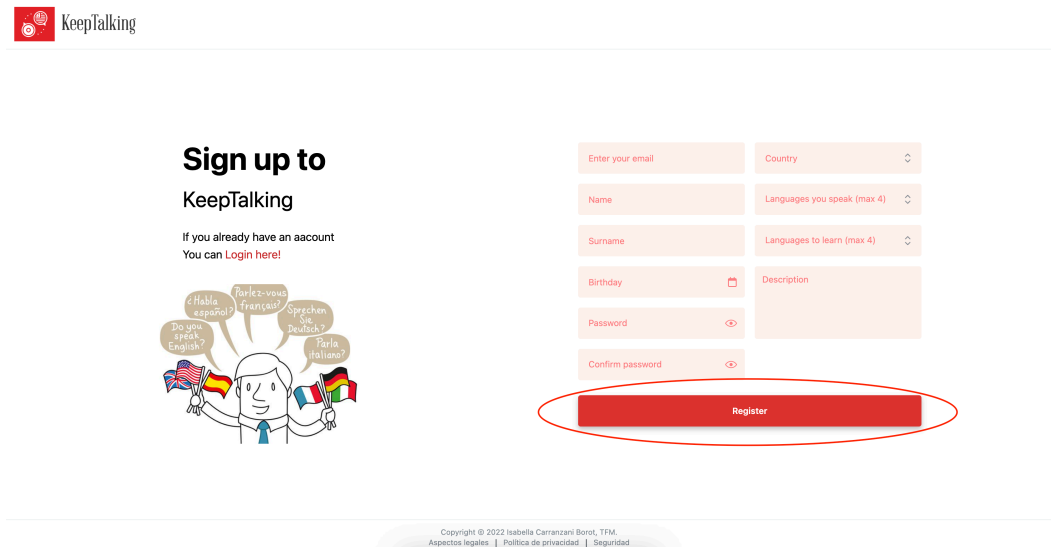



**Ilustración 40 - Guía de usuario, Página de registro**

Si el registro se ha completado correctamente o el usuario ya se había registrado en otra ocasión, se accederá a la página de inicio de sesión donde se deben rellenas los campos **"Email"** y **"Password".** Una vez rellenados se debe dar al

botón **“Login”** para acceder a la aplicación, se encuentra marcado en rojo en la ilustración inferior.

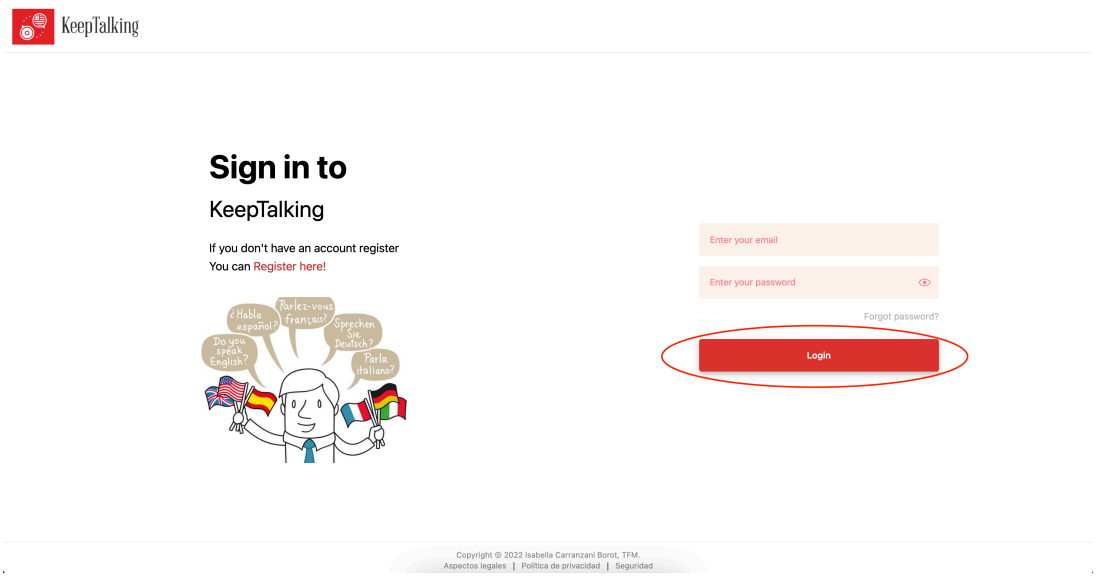



**Ilustración 41 - Guía de usuario, Página de inicio de sesión**

## Dashboard

Cuando el usuario accede a la aplicación la primera página es el dashboard, donde se encuentra el listado de personas que coinciden con sus requerimientos de aprendizaje.

En la imagen de debajo, marcado con flechas naranjas están los tres filtros para encontrar usuarios, de izquierda a derecha: Filtro por nombre de usuarios, Filtro por idiomas que el usuario sabe hablar y Filtro de idiomas que el usuario quiere aprender.

Marcado en azul hay un botón **“Initiate chat”** que lleva directamente a la conversación con ese usuario, pero marcado con la flecha azul, en el *header* se encuentra el enlace a la lista de chats activos que tiene el usuario autentificado.

Marcado en rojo, hay un botón **“Create meet”** que abre una modal para crear un evento. Pero marcado con la flecha roja, en el *header* se encuentra el enlace al calendario de usuario autenticado donde están todas sus reuniones agendadas.

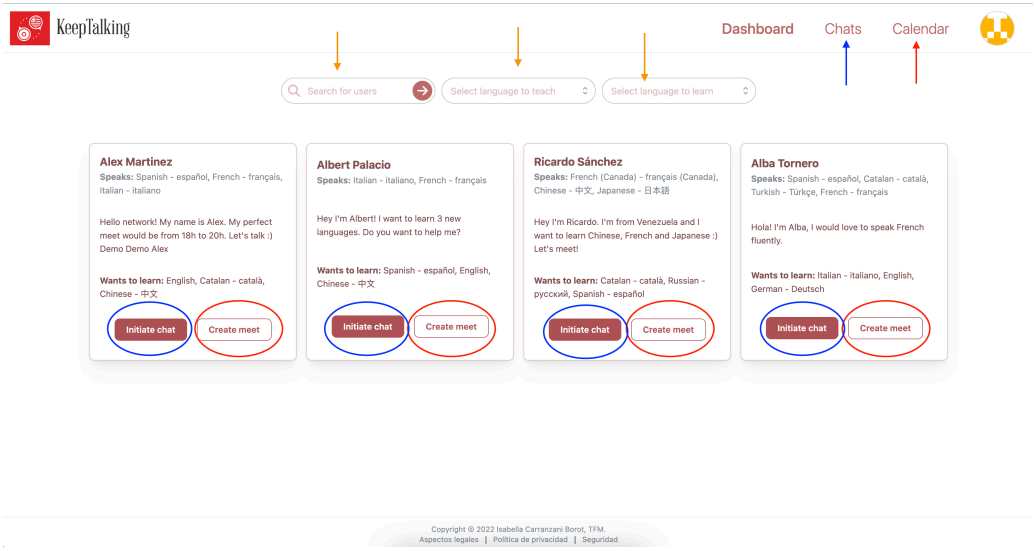



**Ilustración 42 - Guía de usuario, Dashboard**

## Listado de chats

Si el usuario quiere acceder al listado de sus chats desde el enlace del header, le aparecerá el listado con todos los chats que tiene iniciados.

En la parte superior de la lista se encuentra un botón, marcado con la flecha roja, **‘Delete all conversations’**, para eliminar todos los chats que tiene activos.

El nombre que aparece marcado en azul es el nombre de la persona con la que se está manteniendo la conversación y el icono del cesto de la basura que está marcado con la flecha roja en la imagen es para eliminar solo esa conversación.

Haciendo *clic* en la línea del chat se accede a la conversación con la persona con la que se quiere hablar y se abre el panel de chat.


**Ilustración 43 - Guía de usuario, Listado de chats**

## Chat Panel

En el panel de chat se ven todos los mensajes que se han enviado en tiempo real entre dos usuarios, en la parte inferior como se muestra en la imagen marcado en rojo, se encuentra el *input* para escribir el mensaje y el botón para enviarlo.

En la derecha del nombre hay un botón "**Add meet**" que tiene la misma función que el botón del *dashboard* "**Create meet**", abre la modal con el formulario de crear nuevos eventos.

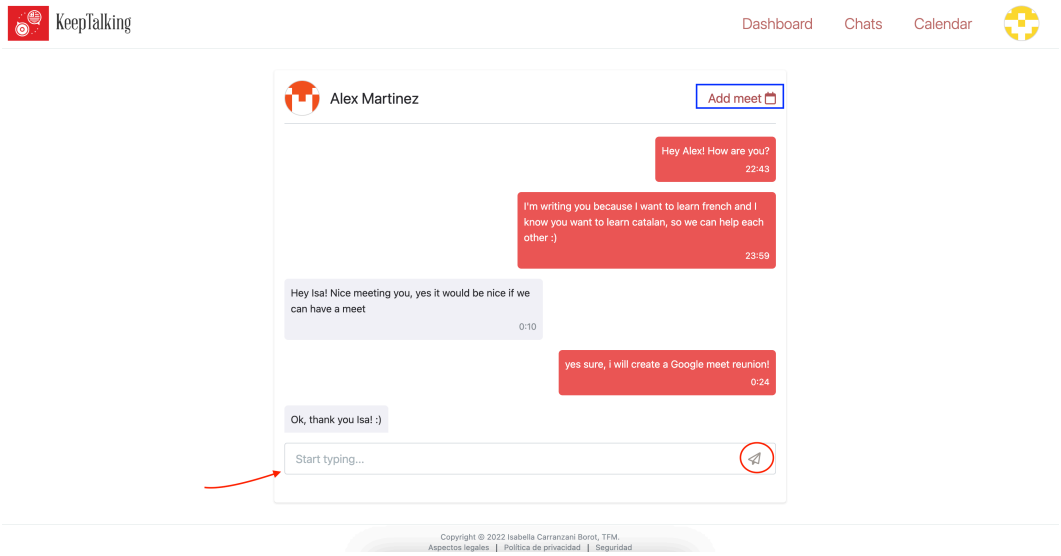



**Ilustración 44 - Guía de usuario, Chat**

## Calendario

En el calendario el usuario tiene la vista general de todos sus eventos programados. En la imagen se pueden ver tres marcas con tres colores diferentes:

- **Rectángulo naranja:** el usuario puede cambiar la vista del calendario, puede ser mensual, semanal y diario.
- **Flecha azul:** el usuario puede editar o eliminar un evento si le dan *clic* a los eventos que están en el calendario, aparece una modal como la de creación.
- **Flecha roja:** el usuario puede crear un evento si hacen *clic* en cualquier día del calendario, se abre la modal de creación.

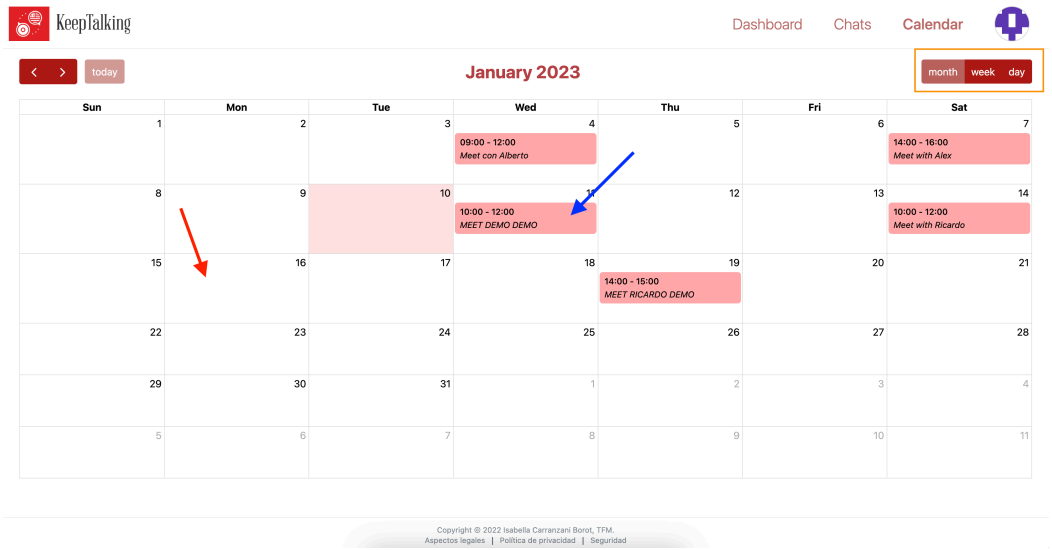



**Ilustración 45 - Guía de usuario, Calendario**

## Perfil de usuario y Logout

En el perfil de usuario aparece un formulario como el de registro, el usuario puede modificar sus datos de perfil y guardarlos, aparece marcado en azul en la imagen inferior.

Por último, como se puede ver en la ilustración, en color rojo, el usuario puede cerrar sesión y volver a la página de inicio desde el enlace "**Logout**".

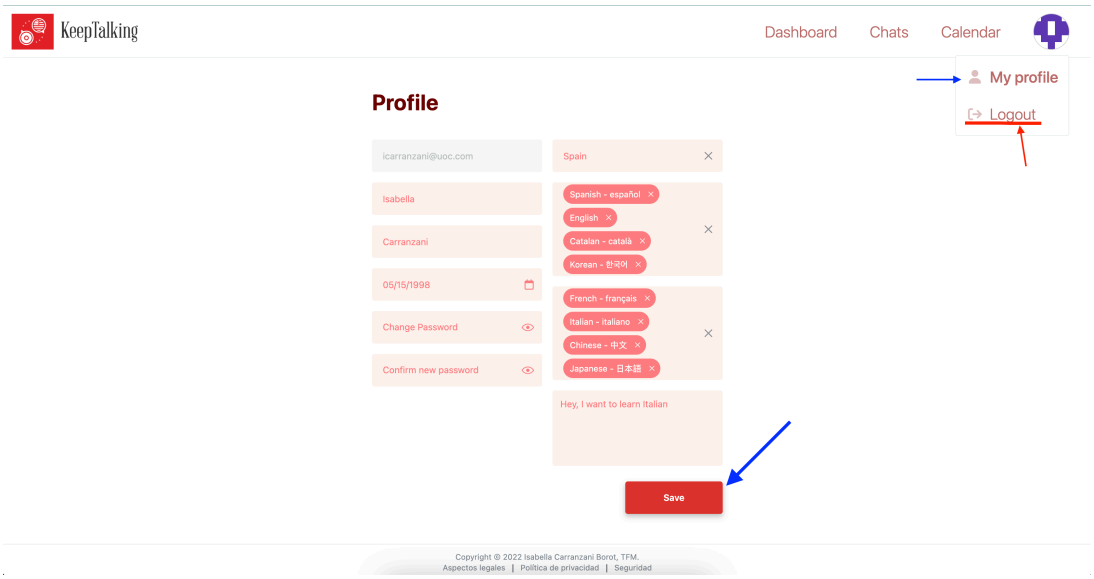



**Ilustración 46 - Guía de usuario, Perfil de usuario**