Logotipo

Descripción generada automáticamente

**UNIVERSIDAD ALFONSO X EL SABIO**

**TRABAJO DE FIN DE GRADO**

*Desarrollo full stack de aplicación móvil híbrida para el aprendizaje de idiomas*

**AUTOR: David Soriano Pardo**

**TUTOR: Antonio José Reinoso Peinado**

1. Índice

[2. Abstract 4](#_Toc105749079)

[3. Resumen y objetivos: 5](#_Toc105749080)

[4. Antecedentes 8](#_Toc105749081)

[4. Análisis y especificación de requisitos 18](#_Toc105749082)

[4.1 Brainstorming: 18](#_Toc105749083)

[4.2 Diseño de la aplicación: 19](#_Toc105749084)

[4.2.1 Aplicación híbrida: 19](#_Toc105749085)

[4.2.2 Aplicación de back-end (API Rest): 21](#_Toc105749086)

[4.2.3 Frameworks para Node.js: 23](#_Toc105749087)

[4.2.4 Bases de datos: 24](#_Toc105749088)

[4.2.5 Driver de base de datos: 30](#_Toc105749089)

[4.2.6 Despliegue de MVP 30](#_Toc105749090)

[4.2.7 Esquema de la base de datos 31](#_Toc105749091)

[5. Propuesta de solución 33](#_Toc105749092)

[6. Plan de trabajo 35](#_Toc105749093)

[6.1 Requisitos funcionales 35](#_Toc105749094)

[6.2 Requisitos no funcionales 36](#_Toc105749095)

[7. Desarrollo de la solución 38](#_Toc105749096)

[8. Despliegue e instalación 48](#_Toc105749097)

[8.1 Versiones funcionales desplegadas: 48](#_Toc105749098)

[8.2 Ejecutar en local: 49](#_Toc105749099)

[9. Evolución y trabajo futuro 51](#_Toc105749100)

[10. Bibliografía 52](#_Toc105749101)

<https://github.com/dawsnap/languages-quiz-expo-nest>

Expo Release:

<https://expo.dev/@dawsnap/hybrid-app-tfg-davidsoriano>

Nest Rest API:

<https://backend-tfg-davidsoriano.herokuapp.com/>

****

Figura 1. Escanea este código QR con la cámara y la aplicación Expo Go instalada para ejecutar la aplicación en tu dispositivo. La aplicación apunta a la instancia de Heroku de la API Rest.

# 2. Abstract

This project will consist of an application to learn and practice a foreign language using your smartphone. Consumer needs in terms of technology, social media and app usage are increasingly becoming more likely to expect technology will provide results faster, and by using the least amount of effort as possible. Thus, usually, when we speak about the target of a smartphone app of this kind, the use case for most people will always be using it when you are on the go, idle or you just want to spend a small amount of time into it every day and a way to keep it a habit.

Keeping that in mind I found the option of a native app a better choice over the available app alternatives. If your business depends on users to come back to your application every day, there is no way like them having your application installed. Browsers are moving forward experimenting about the approach to fully support PWAs (Progressive Web Apps) as a native application in different OS and even for both Desktop and Mobile. I feel it will be a reasonable alternative someday, but as today a hybrid app will be the best choice as this app, as there are few native/OS-exclusive features involved in the app behavior

It will be a full stack application. In a real-life scenario there would be plenty of data that would need to be handled server-side. In this app I will set a simple REST API back-end to be consumed by the app, which will provide the application of the state necessary to function properly.

The stack I am going to use for this project is going to be Expo, a React Native framework for the hybrid app, NestJS, a Node.js framework for the back-end service. This will be connected to a Postgres database using the Prisma ORM. I will also use git for source control during development and it will be published to GitHub

# 3. Resumen y objetivos:

Aprender un nuevo idioma no es una tarea fácil. Requiere dedicación, invertir muchas horas estudiando, repitiendo conceptos, confundiéndolos… además de normalmente, un propósito para aprenderlo ya sea por motivos personales, laborales o por amor al arte, muchas veces mantener la motivación para aprender un nuevo idioma no es la suficiente.

Gracias a la explosión de internet y la globalización aprender un nuevo idioma es una tarea que se ha vuelto mucho más accesible para todas aquellas personas que hasta ahora no tenían los recursos para hacerlo, que es en gran manera mucho más importante de la forma en la que también se ha facilitado dicho aprendizaje. Hoy en día existen una infinidad de recursos para poder aprender de una manera online e incluso gratuita, existen cursos, recursos lingüísticos, foros y lugares donde conectar con gente nativa o con tus mismos intereses con la que poder aprender y practicar, libros, videos y en definitiva no solo contenido que te permita aprenderlo sino también poder utilizar dicho idioma una vez aprendido.

Por supuesto acudir a clases presenciales para aprender un nuevo idioma es una opción viable para mucha gente, pero cada vez más gente que no tiene interés en asumir dicho esfuerzo; quiere añadir una actividad productiva a día a día; o les gustaría aprender un idioma, pero no lo suficiente como para ir a clases opta por hacerlo a través de internet.

Aquí es donde entran las aplicaciones de aprendizaje de idiomas, donde además de incitarte al hábito te permiten hacerlo a través de juegos interactivos y personalizados a tu nivel de conocimiento, algo que por supuesto, puede conseguirse en un aula, pero es obvio que el punto a favor de dicho método es el de poder hacerlo de manera inmediata a demanda, donde y cuando quieras utilizando tu smartphone, durante el tiempo que quieras

Existe una gran cantidad de aplicaciones de aprendizaje de idiomas, por lo que decidirse por una normalmente suele ser complicado, puedes basarte en la metodología, el precio o tu preferencia personal, aunque la realidad es que probablemente ninguna de ellas sirva por sí misma para aprender un idioma hasta hablarlo con soltura, sino más bien de modo introductorio. Es algo primordial que tener en cuenta a la hora de utilizar una aplicación de esta índole, lo más probable es que necesites algún tipo de inmersión lingüística, ya sean clases conversacionales o un estudio adicional pada poder adquirir un nivel adecuado, o incluso combinar el uso de varias de estas aplicaciones con cualquier método que complemente nuestro aprendizaje.

Muchas veces, de hecho, existen usuarios que quizá no tengan esto último en cuenta, y al comenzar a estudiar un nuevo idioma en una de estas aplicaciones se encuentren avanzando a una velocidad de aprendizaje menor de la que esperaban, o repitiendo conceptos básicos mientras se encuentran deseosos de avanzar en materia, lo cual muchas veces conlleva abandonar el uso de la aplicación por completo y con ello muchas veces el aprendizaje de la lengua por completo.

También existe otro problema añadido para aquellos que sí avanzan y aprenden utilizando una de estas aplicaciones, y son aquellos que aprenden rápidamente todos los conceptos que puede ofrecer una de estas aplicaciones, los ejercicios les salen bien a la primera, pueden avanzar hasta el final de la lección sin problema, incluso saltar lecciones que consideren demasiado fáciles, lo cual es común entre hispanohablantes aprendiendo lenguas romances, que son objetivamente más fáciles de aprender que para, por ejemplo, un angloparlante. Esto puede darle una falsa sensación de conocimiento que luego llevada a la práctica y al uso del lenguaje en una situación real puede no ser satisfactoria, ya que pensaba que ya tenía un manejo suficiente, pero se da cuenta de que no es así.

Es por ello, he marcado como objetivo crear una aplicación que de algún modo complemente este aprendizaje, a través de un quiz que permita evaluar tu evolución, evaluándote en cuestiones de vocabulario, obviando la metodología utilizada por todas las aplicaciones analizadas: la repetición espaciada[[1]](#footnote-2).

La aplicación no pretende competir, sino complementar el aprendizaje al utilizar las aplicaciones disponibles en el mercado para aprender idiomas, saltándose su principio fundamental, que es el orden del contenido: de más fácil a más difícil, se pretendería cubrir el aprendizaje de contenido desordenado y no específico, para complementar al aprendizaje ordenado y específico.

# 4. Antecedentes

Existen una gran cantidad de aplicaciones de aprendizaje de idiomas en el mercado, cada una con su propia metodología y experiencia de usuario.

Vamos a analizar las más importantes del mercado:

* Duolingo:

Icono

Descripción generada automáticamente

Es la app más utilizada para el aprendizaje de idiomas, lanzada en 2012, se estima que 9,6 millones de usuarios utilizan la aplicación diariamente y que cuentan con cerca de 500 millones de usuarios registrados en todo el mundo[[2]](#footnote-3). y ofrece la posibilidad de aprender entre 41 idiomas distintos.

En el año 2013, Apple premió a Duolingo como “aplicación de iPhone del año”, convirtiéndose en la primera aplicación educacional en recibir este galardón[[3]](#footnote-4)

Fue la primera aplicación que popularizó de esta manera la ludificación del aprendizaje de idiomas a través de una aplicación para dispositivos móviles, donde a través de ejercicios sencillos, utilizando la repetición espaciada[[4]](#footnote-5), donde a base de prueba y error, el usuario puede entender frases simples sin necesidad de leer textos largos ni hacer hincapié en la gramática. La repetición espaciada es utilizada de una manera u otra en todas las aplicaciones, ya que está demostrado que mejora el aprendizaje, y consiste en mostrar menos las palabras/oraciones que ya conoces, y más aquellas en la que has fallado.

Esto convierte a Duolingo en una estupenda forma de comenzar un nuevo idioma en el caso de querer familiarizarnos con un idioma que nunca hemos estudiado o practicado, aprender ortografía y adquirir nuevo vocabulario y pronunciación, aunque normalmente no es suficiente para desarrollar las habilidades comunicacionales verbales, por lo que se recomienda completar la práctica con otros métodos de aprendizaje como la inmersión lingüística total por ejemplo asistiendo a cursos de inmersión, leyendo cuentos para niños o viendo películas con subtítulos.

Utiliza un modelo freemium donde el usuario puede pagar la versión premium, con un precio de 13,99€ mensuales para eliminar los anuncios y desbloquear nuevas funcionalidades[[5]](#footnote-6).

Duolingo está disponible para web y para ambos iOS y Android, y se estima que el 80% de su tráfico viene de los usuarios de app móvil.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Figura 2: En esta imagen se puede apreciar el esquema del contenido de Duolingo, dividido en lecciones temáticas



*Figura 3: Duolingo distintos tipos de ejercicio, donde puedes traducir una frase del idioma estudiado a tu idioma o viceversa, seleccionando las palabras en el orden correcto, ejercicios de pronunciación, de vocabulario… etc.*

* Memrise:

Imagen que contiene parada, camión, naranja, firmar

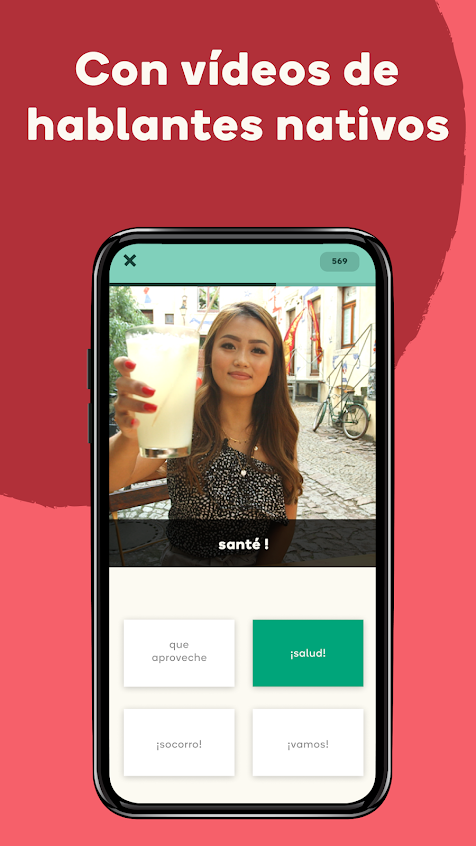
Descripción generada automáticamente

Esta app, creada en 2010 dos interesados en la memoria, uno de ellos fue Ed Cooke, premiado con el premio *Grand Master of Memory*

*(GMM)*, otorgado por la *International Association of Memory*. El galardón se consigue tras hazañas como memorizar 1000 dígitos aleatorios en el transcurso de una hora, el orden de 10 mazos de cartas en una hora y el orden de un mazo de cartas en dos minutos*[[6]](#footnote-7)*, y su compañero Greg Detre, neurocientífico especializado en la ciencia de la memoria y pérdida de los recuerdos por la universidad de Princeton, juntos quisieron crear una forma de aprendizaje a través de flashcards, utilizando también el aprendizaje espaciado[[7]](#footnote-8).

Lo que la hace especial es la posibilidad de acceder a cursos creados por la comunidad, y no solo sobre idiomas sino temas de todo tipo, como por ejemplo capitales del mundo, astronomía, historia, banderas, física, arquitectura, arte, cine, botánica, matemáticas… así como, por supuesto, la posibilidad de utilizar este método de aprendizaje para tus propias flashcards.

Actualmente cuentan con unos 60 millones de usuarios[[8]](#footnote-9) y siguen un modelo de negocio freemium, en el que la versión premium sirve para desbloquear acceso a todas las lecciones, todos los cursos de idiomas, 30.000 videos de nativos y eliminar la publicidad, por un precio de 6,99€



*Figura 4: Memrise hace hincapié en utilizar reglas mnemotécnicas, como imágenes, diagramas o conceptos visuales para retener vocabulario o distintos conceptos lingüísticos*



Figura 5: En esta imagen se representan palabras aprendidas y el progreso de nuestro aprendizaje, representado con una flor que florece cuantas más veces acertamos dicha palabra utilizando la repetición espacial, y una vez que la hemos aprendido por completo aparece con menor regularidad

* Babbel:

Un dibujo de una cara feliz

Descripción generada automáticamente con confianza media

Esta app, fundada en 2008 en Alemania, su modelo de negocio cambia con respecto a los anteriores ya que éste es un servicio premium basado en suscripción, ofrece alrededor de 15 idiomas diferentes y tu suscripción te permite acceder a los contenidos del idioma que elijas, por lo que si deseas aprender varios idiomas simultáneamente tendrás que adquirir una suscripción por cada idioma deseado­. En 2020 anunció que había vendido más de 10 millones de suscripciones[[9]](#footnote-10)

Los cursos están organizados por nivel: principiante, intermedio y avanzado, y ofrecen lecciones con contenido propio original elaborado por hablantes nativos, como por ejemplo trabalenguas, frases hechas, coloquialismos y dichos, así como vocabulario específico para el ámbito laboral, como inglés enfocado a relaciones públicas, márketing o negocios.

También ofrecen el servicio Babbel Live, donde se ofrecen cursos interactivos online por parte de profesores nativos, en grupos de 6 estudiantes, donde también puedes interactuar con los estudiantes que están en la misma clase online que tú.

Tanto el contenido de la plataforma como las lecciones en la aplicación suelen estar enfocados a escenarios específicos, como contestar el teléfono, tener una conversación con un desconocido, vida cotidiana…

Cuentan con suscripciones ronda entre los 9,90€ mensuales por tres meses, 5,40€ mensuales por doce meses y como ya he dicho el precio de la suscripción incluye únicamente un idioma.



*Figura 6: Babbel enfatiza en aprender a través de lecciones para situaciones específicas que puedes encontrarte en tu día a día, como pedir un café en una cafetería, presentarte o específico para el ámbito laboral.*

* Rosetta Stone:

Logotipo

Descripción generada automáticamente­­­­­­

Este software, actualmente disponible en forma de aplicación móvil y web, se trata de la versión modernizada de un software lanzado en 1992 en forma de CD. El nombre de la aplicación hace referencia a la Piedra de Rosetta, una piedra, actualmente expuesta en el museo británico de Londres, cuya inscripción es fue el primer texto plurilingüe descubierto en la era moderna y ayudó a descifrar los para entonces ininteligibles jeroglíficos egipcios ya que ofrecía una versión del mismo texto traducido al griego antiguo, demótico y egipcio.

Durante un ejercicio de Rosetta Stone, se utilizan una combinación de imágenes, texto y sonido, donde los niveles de dificultad aumentan a medida que el usuario avanza. El usuario asocia un sonido o texto con una imagen para aprender vocabulario y funciones gramaticales sin traducción alguna.

El objetivo del programa es que los estudiantes aprendan el idioma estudiado de la misma manera que aprendieron su lengua materna. Recibió algunas críticas por no respetar la identidad cultural de los idiomas al utilizar las mismas palabras y oraciones para todos los idiomas: Todas las lecciones para los distintos idiomas enseñan los mismos conceptos en el mismo orden, incluso utilizando las mismas imágenes que no se identifican con la cultura de la lengua estudiada.

Al igual que Babbel, es un servicio con suscripción cuyos precios oscilan entre los 11,99€ mensuales para tres meses, 9,99€ en el caso de la suscripción de doce meses o 199€ para la versión vitalicia.

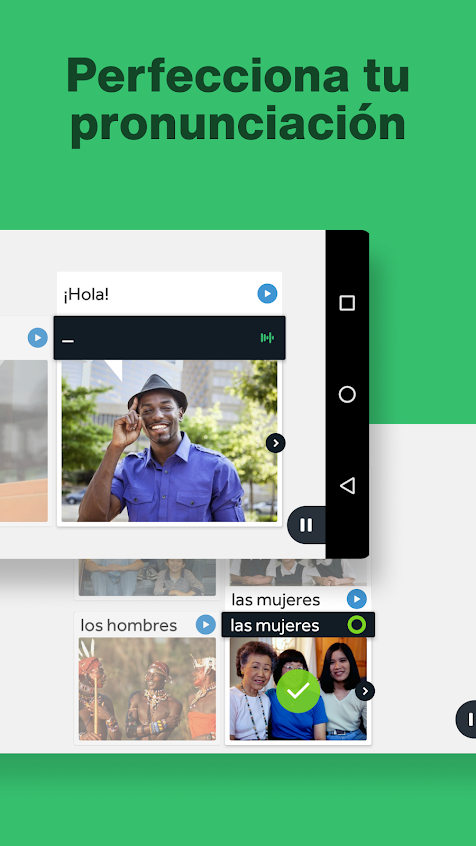


Figura 7: Rosetta Stone utiliza también imágenes para memorizar conceptos y ser capaces de retenerlos con mayor facilidad

* Busuu:

Icono

Descripción generada automáticamente con confianza media

Fundada en 2008 y de origen español, Busuu cuenta con más de 100 millones de usuarios y su nombre procede de una lengua casi extinta de origen camerunés con nombre idéntico.

Ofrece cursos basados en los niveles del marco común europeo de referencia para las lenguas, con niveles A1, A2, B1 y B2, donde el contenido se divide en lecciones con escenarios comunes, con preguntas de selección múltiple, de hablar y escribir y ofrece hasta 13 idiomas diferentes.

Utiliza lecciones creadas por hablantes nativos, adaptadas al marco mencionado anteriormente, así que es una excelente opción en el caso de estés estudiando un idioma con propósito de obtener un certificado, ofrece consejos por parte de la comunidad, recordatorios de palabras que podrías estar olvidando y consejos sobre el idioma.

En el caso de que ya conozcas un idioma, te ofrece la opción de comprobar tus conocimientos y adecuar así las lecciones a tus puntos flojos de conocimiento[[10]](#footnote-11).

Busuu ofrece hasta 13 idiomas distintos en su versión “Premium plus”, cuyo precio oscila entre los 11,66€ mensuales para una suscripción de tres meses, o 6,25€ para una suscripción de 24 meses, así como una opción “Premium”, limitada a 1 idioma, cuyo precio oscila entre los 10€ mensuales para la suscripción de tres meses, hasta 5,42€ para la suscripción bianual. Se estima que han alcanzado la cifra de 100 millones de usuarios registrados[[11]](#footnote-12)

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

Figura 8: Busuu ofrece contenido creado por nativos, así como una comunidad con la que intercambiar consejos de pronunciación y gramática.



Figura 9: Busuu también ofrece un seguimiento del progreso alineado con e marco común europeo de referencia para las lenguas, por lo que es una excelente opción si necesitas obtener uno de estos certificados

# 4. Análisis y especificación de requisitos

El proyecto consiste en una aplicación para el aprendizaje de idiomas, aquí detallo algunas ideas relevantes que sirven como semilla del planteamiento prototipo el cual será detallado en la propuesta de solución. Solo detallo las ideas que me han parecido más interesantes basándome en mi uso personal de las aplicaciones ya existentes para el aprendizaje de idiomas.

## 4.1 Brainstorming:

1. Aplicación tipo *quiz*: donde el usuario recibe una palabra y ha de elegir la opción correcta entre una lista de palabras incorrectas.
   1. En el caso de acertar el significado de la palabra un número determinado de veces dicha palabra debería disminuir su frecuencia de aparición para así dejar paso a otras palabras que el usuario desconozca, asumiendo que dicha palabra está ya aprendida
   2. El usuario puede repasar las palabras más difíciles o aquellas en las que ha fallado más veces.
2. Aplicación tipo *flashcards*: donde el usuario simplemente estudia las tarjetas, con una palabra o frase junto a su traducción, dispone de un botón para pasar a la siguiente tarjeta
   1. De ser esta la opción escogida se podría optar por utilizar imágenes en alguna de las tarjetas o en todas, así como diagramas Aprendizaje por imágenes[[12]](#footnote-13)
   2. También, se podría optar por una opción híbrida entre el estudio y la interactividad, ya que, analizando el gran éxito de las aplicaciones de aprendizaje de idiomas ya existentes podemos concluir con bastante seguridad que el usuario medio prefiere la interactividad a la hora de utilizar este tipo de aplicaciones, y no simplemente estudiar como lo haría estudiando unas tarjetas físicas con traducciones.
3. Aplicación tipo *aprendizaje dividido en leccione*s: donde el usuario tendría una serie de lecciones donde elegir y estudiar vocabulario específico del temario seleccionado (básicos, comida, adjetivos, objetos, direcciones…)

Una vez estudiados los distintos planteamientos de la aplicación en sí podemos pasar al análisis técnico. Como ya explico en el *abstract,* se decide hacer una aplicación nativa por lo particular del tipo de aplicación, ya que desde mi punto de vista el usuario de una aplicación para el aprendizaje de idiomas prefiere dedicarle cierto tiempo al día.

Duolingo recomienda dedicar entre 15 y 30 minutos diarios[[13]](#footnote-14). Rosetta Stone recomienda unos 30 minutos diarios[[14]](#footnote-15) Mientras que otras aplicaciones como Babbel recomienda entre 10 y 15 minutos diarios[[15]](#footnote-16) y Busuu lo incluso más recomendando estudiar 10 minutos diarios[[16]](#footnote-17)

## 4.2 Diseño de la aplicación:

## 4.2.1 Aplicación híbrida:

El usuario medio por lo general prefiere acceder a aplicaciones con un uso “casual” como es el caso a través de una aplicación ya instalada en su teléfono y no a través del navegador, por simple accesibilidad y facilidad. Es por ello por lo que se decide que la mejor opción es una aplicación nativa.

El proyecto no tiene grandes dependencias de funcionalidades específicas de sistema operativo, sino que por el contrario puede basarse en una interfaz gráfica simple de input output donde el usuario interactúa para avanzar en el contenido, es por ello considero la mejor opción que la aplicación móvil sea una aplicación híbrida que pueda fácilmente ser desplegada en varios sistemas operativos usando el mismo código fuente[[17]](#footnote-18). De esta manera podría, en este caso, lanzar una aplicación para ambos Android y iOS sin depender de rehacer el código fuente en dos proyectos y dos lenguajes de programación independientes.

¿Pero que es una aplicación híbrida?

Los 3 frameworks de aplicaciones nativas más populares son, por popularidad en GitHub, Flutter; React Native e Ionic.

* El framework Flutter es la más popular en GitHub y está enfocándose a ser completamente multiplataforma, de hecho, sirve incluso para construir aplicaciones para los sistemas operativos más populares de escritorio, Windows, macOS y Linux. Ubuntu está integrándolo próximamente en algunas funcionalidades en próximas versiones, como el nuevo instalador de Ubuntu[[18]](#footnote-19), y parece que tiene potencial de cara al futuro. Utiliza el lenguaje de programación de Google, Dart, por lo que la curva de aprendizaje puede ser mayor para alguien que nunca haya utilizado este lenguaje.
* Ionic, lanzado en 2013, fue una de las primeras opciones open source disponibles para crear aplicaciones híbridas para móvil[[19]](#footnote-20). Inicialmente utilizaba el para entonces revolucionario (y ahora deprecado) framework AngularJS que introdujo el Modelo Vista-Controlador (MVC) al desarrollo en JavaScript y que junto con la API de Apache Cordova permitía utilizar funcionalidades nativas de un dispositivo móvil. Hoy en día ya no depende de esas dependencias y permite usar cualquier framework de front-end, ya sea React.js, Angular o Vue.js, por lo que se puede utilizar HTML, CSS y JavaScript para compilar a una aplicación nativa.
* Finalmente tenemos React Native, el cual, fue lanzado en 2015 (dos años más tarde que React) por Meta (ex Facebook). Se trata de un framework para desarrollar interfaces de usuario basado en React para crear aplicaciones nativas para Android y iOS entre otros, por lo que la curva de aprendizaje es mucho menor en el caso de que ya conozcas el entorno de React y JavaScript

Por esto último me decanto a utilizar React Native para el desarrollo del proyecto, debido a mi conocimiento y mi interés por probar esta tecnología en concreto, ya que he utilizado el ecosistema de React con anterioridad, pero no React Native.

Existe un framework para React Native llamado Expo[[20]](#footnote-21) el cual nos provee de herramientas para facilitar el desarrollo de una aplicación en React Native.

## 4.2.2 Aplicación de back-end (API Rest):

Como se trata de una aplicación full stack, deberemos conectarlo con un servicio de back-end para manejar toda la lógica e información que queramos retener en el servidor. En el caso de esta aplicación manejarán funcionalidades como conexión con una base de datos y exposición de datos a través de endpoints de una API Rest.

Hay una infinidad de herramientas para crear una API REST, en mi caso me decanto por mi lenguaje de programación predilecto, JavaScript/TypeScript por lo que mis dos opciones son Node.js o Deno

* Node.js es un entorno de ejecución multimedia basado en el motor V8 de Google para ejecutar JavaScript fuera de un navegador web[[21]](#footnote-22). Te permite utilizarlo para escribir herramientas de líneas de comandos (CLI), scripting, y para crear servidores web. Al tener una arquitectura basada en eventos es capaz de manejar lectura y escritura asíncrona, lo cual es óptimo para optimizar la escalabilidad. Node.js se creó en 2009 con estas bases desde el comienzo de su desarrollo, debido a que su creador, Ryan Dahl[[22]](#footnote-23) se mostró crítico por que el principal servidor web de la época, Apache HHTP Server difícilmente podía superar las 10.000 conexiones concurrentes y la principal forma de programación de la época era la programación secuencial (o síncrona), lo cual bloqueaba el hilo de ejecución para el proceso entero o implicaba la utilización de soluciones alternas en cuanto a lo que respecta a la escalabilidad.

Node.js es hoy en día el entorno predilecto en cuanto a la creación de proyectos con JavaScript, y sin duda su éxito no podría haberse dado de no ser por la integración con npm[[23]](#footnote-24), (Node Package Manager) el cual facilitó enormemente la creación de software extendiendo de paquetes open source publicados por la comunidad.

Un paquete puede ser una funcionalidad reusable o un CLI, y un paquete puede instalarse simplemente dentro de tu proyecto o de manera global para utilizarlo a través de la terminal de tu equipo como si fuese un CLI nativo, ejecutado a través de Node.js. Actualmente hay más de 1.3 millones de paquetes disponibles[[24]](#footnote-25)

* Deno – (Su nombre es un juego de palabras invirtiendo el orden de las sílabas de Node) nace en 2018 precisamente como alternativa al mismo poniendo el foco en la seguridad, algo que Node.js podría haber dejado en manos del desarrollador.

La principal diferencia con Node.js es que no permite acceso al sistema de archivos, el acceso de red, la ejecución de otros scripts y el acceso a variables de entorno sin proporcionar permiso explícito durante la ejecución[[25]](#footnote-26), lo cual implica poder utilizar solo los permisos necesarios y así impedir que una vulnerabilidad de uno de esos tipos en un paquete dependiente no afecte a la seguridad de tu aplicación. También pretende acabar con la centralización de paquetes, rompiendo su dependencia de NPM (el cual es un servicio centralizado y controlado por una organización) y acceder a un paquete a través de la url donde esté alojado el paquete.

Entre estos dos entornos, escojo Node.js por mi experiencia utilizándolo y por su mayor madurez, ya que Deno está listo para producción, pero considero que es preferible esperar a un grado de madurez y esta decisión puede afectar a la hora de encontrar bugs que dificulten el desarrollo.

Una vez escogido Node.js, hay una gran lista de frameworks web que nos facilitarán el desarrollo de nuestra API Rest[[26]](#footnote-27).

## 4.2.3 Frameworks para Node.js:

* Express.js es el framework web predilecto para Node.js, dispone de una utilidad HTTP y un middleware para aplicar una lógica global a todas las peticiones salientes y entrantes (autenticación, custom headers…) y permite crear una API sólida de manera rápida y sencilla[[27]](#footnote-28).
* Meteor es un framework full stack orientado a la creación de aplicaciones completas, abarcando ambos, front-end y back-end. Esto facilita enormemente el desarrollo ya que nos provee de un montón de herramientas “out of the box", como puede ser la autenticación[[28]](#footnote-29). Al ser un framework web full-stack lo hace una opción inviable para nuestro stack ya que su propósito no es el de hacer un back-end desprendido.
* NestJS es un framework web que utiliza Express.js bajo el capó y le añade funcionalidades propias para hacerlo un framework web progresivo, con un CLI muy similar al de Angular que permite el *scaffolding*, pudiendo crear módulos, servicios, controladores y middlewares con un solo comando a través de su CLI previamente mencionado.

Existen también un montón de paquetes creados específicamente para Nest.js[[29]](#footnote-30) con los que puedes extender la funcionalidad de tu back-end para casos específicos, como la creación de microservicios, Websockets, documentación de endpoints con Swagger… así como versiones de librerías populares adaptadas a NestJS como Sequelize, GraphQL, TypeORM o Passport

Entre los distintos frameworks me decanto por utilizar NestJS ya que también proporciona una clara estructura desde el principio del proyecto, que te permite añadir funcionalidades independientes en un proyecto estructurado, aprovecha el uso de TypeScript para tipar todos sus métodos y es una opción genial para crear un microservicio que se conecte con una base de datos y exponga unos endpoints a través de una API REST que devuelva la información necesaria a la aplicación.

## 4.2.4 Bases de datos:

En cuanto a la base de datos y como vamos a conectarnos a ella hay una infinidad de opciones, distinguiendo entre base de datos relaciones y no relacionales (también llamadas NoSQL)[[30]](#footnote-31).

A la hora de escoger entre una base de datos relacional o no relacional depende mucho de la preferencia personal del desarrollador, sobre todo cuando se trata de proyectos pequeños que van a tener una base de usuarios concurrentes constante y no superior a cuatro cifras. Cuando tener demasiados usuarios concurrentes comienza a ser algo de lo que preocuparse es cuando esta decisión podría marcar el desarrollo de todo el proyecto, pero la decisión de utilizar un sistema de base de datos u otro (o incluso un servicio de bases u otro) nunca será el único punto por el que una aplicación no es escalable, sino que, en muchas ocasiones, puede ser un problema de planteamiento, de programación, del servidor…

Además, existen una infinidad de soluciones para adaptarse a mayor carga, muchas veces a nivel de arquitectura y no de código o de stack. Algunos de ellos podrían ser utilizar cachés para almacenar información estática, replicación de bases de datos, escalado horizontal…

Comparemos los dos paradigmas: Relacionales y No relacionales

* Base de datos relacional[[31]](#footnote-32): También llamadas bases de datos SQL, nombradas así por el lenguaje utilizado para interactuar con ellas (Structured Query Language), las más populares son, entre otras MySQL, MariaDB, PostgreSQL, SQLite, Microsoft SQL Server y Oracle SQL.

Este tipo de bases de datos mantienen la información dividida en tablas, representando una entidad, mientras que cada una de ellas mantiene una estructura basada en columnas, donde cada una de las columnas son el atributo o propiedad de la entidad, y una fila contendrá uno, varios o todos los atributos de esas columnas. Cada fila es representada con un id único, habitualmente asociado de forma incremental y puede asociarse un id de una tabla ajena, donde, por ejemplo, en un sistema de gestión de hospitales, existe una tabla con los distintos hospitales y su información, así como una tabla para los empleados. En la tabla de los empleados se utilizará una referencia, también llamada clave foránea, donde se asocia a un empleado con el id del hospital en el que trabaja.

Utilizando SQL puedes filtrar de forma eficiente la información que quieres obtener, insertar, modificar o eliminar filtrando a través de cualquier dato de cualquiera de las 2 entidades.

Los puntos positivos de utilizar una base de datos relacional incluyen poder añadir información independientemente de los registros ya existentes, la precisión a la hora de consultar datos, la integridad, ya que cada columna solo puede albergar un tipo de dato y cada columna solo almacena un atributo sobre la entidad, la posibilidad de gestionar permisos por usuario y host, así como la concurrencia, al poder acceder a la misma base de datos simultáneamente desde varios lugares al mismo tiempo.

* Base de datos no relacional[[32]](#footnote-33): o como ya he dicho, también llamadas bases de datos NoSQL, entre las más populares se encuentran algunas como MongoDB, Redis, Apache Cassandra, DynamoDB and Hypertable o Firebase.

La diferencia principal entre ambos paradigmas es el modo en el cual se almacena la información en las mismas. En una base de datos NoSQL la ésta puede almacenarse en estructuras de datos como documentos. Un documento es bastante similar a un objeto en programación, donde, por ejemplo, en una colección (conjunto de documentos) de pacientes, se almacena la información de los clientes, con atributos sobre el paciente, y, por ejemplo, un objeto que contenga un array de todas las enfermedades del paciente junto con los diagnósticos, tratamientos, medicación…

Las bases de datos NoSQL son una buena opción para manejar una enorme cantidad de información compleja y diversa, y es mucho más flexible en cuanto a las propiedades o atributos que un documento puede albergar. Generalmente son más rápidas que las bases de datos relacionales ya que como la información no está almacenada en varias tablas no se necesita recopilar la información de varios sitios para hacer consultas.

Algunos de los beneficios de usar una base de datos no relacional son la posibilidad de almacenar una enorme cantidad de información sin mayor complicación, sin que la velocidad de las consultas no se vea afectada. También, como ya he dicho, nacen con la premisa de que la información es mutable, por lo que se debería ofrecer mutabilidad a la hora de introducir datos (esto en ocasiones puede implicar acabar con una base de datos con información incoherente o no consistente, pero existen distintos métodos para prevenir este problema). También permiten almacenar cualquier tipo de información: cadenas de textos, números, así como archivos multimedia como fotografías o vídeos, estructuras JSON … algo que las bases de datos SQL no permiten en una primera instancia.

Pasaremos a analizar algunos de los servicios de bases de datos más populares, para ver las características de cada uno, tanto relacionales como no relacionales:

* MySQL[[33]](#footnote-34) es considerada la base de datos más popular, nace en 1994 y es posteriormente comprado por Oracle Corporation y posee una licencia doble, GPL (General Public License) y también comercial, por lo que también se distribuye en varias versiones, la versión Communityy Enterprise. Es muy utilizado por ser la base de datos utilizadas para grandes CMS como Wordpress, Drupal o Joomla, que ocupan una enorme proporción de las webs. Si hablamos únicamente de Wordpress estamos hablando del 37% de todas las páginas web del mundo[[34]](#footnote-35) y durante mucho tiempo estuvo ligado al desarrollo en PHP, que sigue de algún modo u otro estando presente en 8 de cada 10 páginas webs del mundo[[35]](#footnote-36) aunque este número está claramente en declive con la adopción de nuevos stacks, la adopción del Cloud contra los hostings tradicionales, compartidos y VPS auto configurables.

Destacar también MariaDB, lo cual es un “fork” de MySQL que también ganó una enorme popularidad.

* MongoDB[[36]](#footnote-37) es la base de datos no relacional por excelencia, creada en 2009, es una base de datos orientada a documentos que funciona almacenando la información en una estructura de datos llamada BSON (Binary JSON) con un esquema dinámico, lo que significa que no necesitan tener un esquema definido. Cada documento está identificado con una clave única, también llamada ObjectId.

Además de esta clave, MongoDB permite consultas por campo, por rangos o utilizando expresiones regulares o RegExpr, lo cual permite filtrar documentos específicos o hacer consultas complejas. Otra de las funcionalidades que ofrece es la replicación y balanceo de carga, permite crear varias instancias de la misma base de datos replicada para evitar un apagón en caso de fallo del servidor.

Cuando un servidor primario falla, un servidor secundario se convierte automáticamente en el primario para mantener la base de datos operativa, además de ofrecer escalado horizontal, repartiendo la información entre servidores. Entre estas dos funcionalidades MongoDB ofrece que tu base de datos siempre estará operativa para mantener el servicio funcionando independientemente de la carga o de fallos de hardware.

* PostgreSQL: También comúnmente llamado Postgres en su web se autodefinen como “La base de datos open source más avanzada del mundo[[37]](#footnote-38)” y a diferencia de las anteriores, es totalmente open source, por lo que no tiene una organización detrás gestionando su desarrollo, sino que es mantenido por la comunidad. Su creador, Michael Stonebraker lideró un proyecto universitario llamado “Ingres” que pretendía implementar un motor de bases de datos relacional. Poco después, en 1985 se creó un nuevo proyecto que pretendía resolver algunos problemas de la época llamado Post-Ingres que derivó en su actual nombre, Postgres[[38]](#footnote-39).

Postgres ofrece algunas ventajas con respecto al resto de sistemas de bases de datos tradicionales[[39]](#footnote-40), como soporte nativo para varios tipos como direcciones de red (direcciones IP, direcciones MAC, direcciones junto con máscaras de subred…), tipos geométricos, tipos monetarios con soporte local y formateo automático, rangos temporales, así como arrays multidimensionales, soporte horario…

Lo que más destaca a la hora de decidirse por utilizar esta opción de base de datos sobre otra es su soporte por paradigmas propios de las bases de datos no relacionales, utilizando JSONB o incluso pares clave - valor, lo cual nos permite para trabajar con ambos, datos relacionales y no relacionales. En cuanto a funcionalidades propias de las bases de datos relacionales también ofrece más funcionalidades que MySQL según una comparación[[40]](#footnote-41)

* Redis: Me parecía interesante hablar sobre Redis ya que es un sistema con una aplicación completamente distinta y específica. Redis ofrece muchas funcionalidades extensibles a través de módulos (algunos de ellos requieren licencia de pago) para ser utilizado como una base de datos principal, pero por defecto es una base de datos no relacional que utiliza una estructura de claves – valor. Es una base de datos en memoria, además, permite de forma nativa establecer la caducidad de los registros, por lo que es una opción especialmente acertada para ser usado como caché, base de datos clave – valor (lo cual requiere un escenario específico donde se requiera una base de datos de ese tipo), motor de streaming o message broker.

Tras analizar las distintas opciones de bases de datos me decanto por PostgreSQL por su versatilidad a la hora de poder abarcar ambos paradigmas, de bases de datos relacional y no relacional para casos específicos y por ser es el sistema de bases de datos y ofrecer mayores funcionalidades.

Pasaremos a hablar de drivers, o como vamos a hacer que nuestro servicio de NestJS se comunique con la base de datos de PostgreSQL. La mejor práctica para estos casos es utilizar variables de entorno para poder establecer dinámicamente las credenciales y dirección URL donde se encuentra nuestra base de datos, sin necesidad de subirlo a ningún repositorio y evitar el filtrar credenciales al, por ejemplo, subir nuestro código fuente a un repositorio. NestJS y el entorno de Node.js permiten el uso de variables de entorno así que será el método utilizado para inicializar nuestro controlador con la conexión a la base de datos.

## 4.2.5 Driver de base de datos:

Para conectarnos con nuestra base de datos utilizaremos un ORM, (Object Relational Mapping). Un ORM es una técnica que te permite comunicarte y manipular una base de datos utilizando el paradigma de la programación orientada a objetos. Esto permite comunicarte con la base de datos sin necesidad de utilizar el lenguaje propio de la misma, sino que puedes utilizar el mismo lenguaje de programación[[41]](#footnote-42). En este caso al usar TypeScript con Postgres, que es SQL, podríamos escribir código en TypeScript para realizar consultas, etc. Existen varios ORMs para Typescript[[42]](#footnote-43), como Prisma, TypeORM, Sequelize… El ORM que utilizaremos será Prisma[[43]](#footnote-44).

## 4.2.6 Despliegue de MVP

También, una vez el proyecto esté terminado, podemos utilizar Heroku para desplegar una versión de producción de la parte del back-end, gracias a que ofrece una capa gratuita[[44]](#footnote-45) para casos de uso básico, productos mínimos viables o aplicaciones no comerciales en su plan “Free and Hobby”. Así simplemente tendríamos que instalar la aplicación en nuestro dispositivo para que funcione, sin tener que configurar todo el servicio en nuestro equipo.

Expo tiene un entorno que nos permite publicar una versión de preproducción de nuestra aplicación, un paso previo a compilarla para la App Store/Play Store. Para poder acceder a ella, es necesario tener una cuenta de Expo.dev, y una vez publicada dispondremos de una URL que contendrá un código QR para escanear con la app Expo Go desde nuestro smartphone. Esa versión apuntará a la instancia del back-end desplegada en Heroku para poder funcionar.

## 4.2.7 Esquema de la base de datos

Por la parte del esquema de la base de datos deberemos utilizar varias tablas con relaciones entre sí para relacionar la información

* Tabla *languages*: Donde se almacenarán los idiomas activos en los que está disponible realizar el quiz. Esta información será utilizada para renderizar dinámicamente unos botones / opciones en un selector que permitirán seleccionar el idioma en el que vas a realizar el quiz, por lo que deberá incluirse información para pintar, como el nombre del idioma y un icono que simbolice el idioma. De esta forma se podrán añadir idiomas dinámicamente sin tener que cambiar el código fuente de la aplicación simplemente añadiendo registros en la base de datos.

El esquema de la tabla quedaría de la siguiente manera:

* + id: int con clave primaria del idioma
  + name: string de nombre del idioma
  + icon: string path que lleve al icono de dicho idioma
* Tabla *languages\_words:* Donde se almacenará una lista de palabras de cada uno de los idiomas, se usará una clave foránea que haga referencia a la tabla *languages* para identificar el idioma del que se trata.

La aplicación solo está disponible para que un hispanohablante tome un quiz en una lengua extranjera, por lo que, al tratarse de una tabla con relación 1:N, podemos simplemente almacenar la palabra y su significado en castellano, independientemente del idioma que se trate.

Para una futura implementación de localización de la aplicación, donde se hipotéticamente se permita tomar un quiz a una persona no hispanohablante habría que cambiar esta parte del esquema por una relación N:M, utilizando una tabla adicional para relacionar la palabra y su significado con los dos idiomas que correspondan en el caso de una estructura de base de datos relacional, o aprovechando la compatibilidad de BSON de PostgreSQL, donde podríamos almacenar un array con la palabra con su significado en varios idiomas simultáneamente.

El esquema de la tabla para una relación 1: N quedaría de la siguiente manera:

* + id: int clave primaria de la palabra
  + language\_id: int de clave foránea que referencie a la tabla *languages*
  + word: string de palabra extranjera
  + meaning: string de palabra en castellano
* Tabla *quiz\_resuls*: Una vez terminado un quiz, en esta tabla se almacenará el resultado del quiz, junto con el nombre o nickname con el que el usuario quiera identificarse. Se guardará una referencia del idioma en el que se ha realizado, el usuario y la puntuación obtenida.

También se almacenará la información en crudo utilizada para el quiz, ya que al ser generado automáticamente nos permitirá tener un log para debuggear en el caso de que una pregunta fuese errónea. En el caso de contar con una tabla de usuarios, una lógica de registro y autenticación deberíamos utilizar una referencia al id de usuario que ha realizado el quiz.

El esquema de la tabla quedaría de la siguiente manera:

* + id: int clave primaria del resultado del quiz
  + language\_id: int de clave foránea que referencie a la tabla *languages*
  + finish\_time: timestamp con hora de la finalización del quiz
  + username: string con nombre/nickname introducido por el usuario
  + raw\_quiz: JSON con preguntas y respuestas que ha dado el usuario durante el quiz
  + score: int con puntuación obtenida por el usuario en el quiz

# 5. Propuesta de solución

Tras analizar los distintos planteamientos enumerados en la sección anterior concluyo con la que me parece más adecuada:

* Desarrollo de aplicación híbrida con React Native, utilizando el framework Expo
* Desarrollo de API Rest en Node.js, utilizando el framework NestJS, junto con Prisma para la conexión con la base de datos.
* Configuración de base de datos PostgreSQL conectada a nuestro servicio de back-end en NestJS

La aplicación finalmente consistirá en una serie de preguntas tipo “quiz”. Se presentará al usuario un menú sencillo donde podrá comenzar el test para el idioma seleccionado. Una vez comenzado el quiz, en la pantalla aparecerá una palabra en castellano y 4 opciones, siendo una de ellas la traducción correcta. El quiz constará de 10 preguntas y habrá un temporizador que servirá para calcular una puntuación cuando el usuario termine el quiz. Una vez haya terminado el usuario podrá guardar su puntuación en un ránking y consultar posteriormente sus mejores puntuaciones.

La primera tarea será el “seeding”, lo que consiste en sembrar una base de datos con la información inicial necesaria para que la aplicación funcione

El ORM que utilizaremos, Prisma, soporta el seeding creando un archivo

Empezaremos por buscar en internet un listado de palabras junto con su traducción en español. Repetiríamos este proceso hasta conseguir una base de datos con palabras de todos los idiomas que queramos soportar. Deberíamos construir una aplicación dinámica en la que podamos introducir un nuevo idioma a través de base de datos sin necesidad de cambiar el código.

No será necesaria una interfaz gráfica para introducir las palabras ya que se hará en el proceso de seeding. En el caso de querer introducir más palabras, ya sea para una lengua ya existente o una nueva, simplemente deberemos borrar la información ya existente y volver a volcar el seed en la base de datos.

Crearemos nuestro servicio de NestJS utilizando su CLI para generar un proyecto desde cero, configuraremos la conexión con la base de datos a través de Prisma e inicializaremos la base de datos con el seed previamente generado con la información recopilada manualmente

Una vez finalizada, habrá un MVP publicado en internet, para poder acceder al proyecto sin necesidad de instalar nada en nuestro equipo.

# 6. Plan de trabajo

## 6.1 Requisitos funcionales

* RF1: Pantalla que te permita elegir un idioma, y con respecto al idioma seleccionado, iniciar el quiz o bien ver el ránking.
* RF2: Endpoint que devuelva los idiomas en las que está disponible realizar el test.
* RF3: Pantalla donde se realizará el quiz.
* RF4: Endpoint que devuelva una lista de palabras que posteriormente serán utilizadas para las preguntas y respuestas del quiz.
* RF5: Pantalla que permita, una vez terminado el test, ver la puntuación obtenida, guardar tu nombre en un ránking y volver a la home.
* RF6: Endpoint que permita guardar la puntuación obtenida junto con tu nombre en la base de datos
* RF7: Pantalla de visualización de datos que muestre los datos de puntuaciones guardados por los jugadores
* RF8: Endpoint que devuelva una lista de todos los datos de puntuaciones
* RF9: Botones de navegación que permitan navegar entre todas las pantallas

Diagrama

Descripción generada automáticamente

Figura 10: Diagrama de flujo funcional de la aplicación

## 6.2 Requisitos no funcionales

* RNF1: Creación de memoria, investigación y análisis, documentación del desarrollo en la sección “desarrollo de la solución”
* RNF2: Configuración e inicialización de proyectos de Expo y NestJS, así como investigación sobre las librerías necesarias y su compatibilidad con los requisitos
* RNF3: La aplicación deberá ser soportada tanto en dispositivos Android como en iOS
* RNF4: Para evitar problemas de compatibilidad por parte de librerías exclusivas para funcionar en dichas plataformas, se deshabilitará el soporte web ofrecido por defecto por Expo.
* RNF5: Creación de un script que prepare la base de datos automáticamente para que la aplicación pueda funcionar correctamente
* RNF6: No se deberá poder empezar un quiz sin haber seleccionado un idioma previamente
* RNF7: El quiz tendrá una duración de 10 preguntas, y como máximo se tendrán 10 segundos para contestar cada una de las cuestiones, entre 4 opciones distintas
* RNF8: Habrá una barra de progreso que marque el tiempo restante para contestar a la pregunta, y una vez pasado ese tiempo la pregunta se marcará como errónea
* Escala de tiempo

  Descripción generada automáticamenteRNF9: Despliegue de MVP­­

Figura 11: Diagrama de Gantt

# 7. Desarrollo de la solución

Primero construiremos el esqueleto de la aplicación híbrida, creamos un nuevo proyecto de Expo. Instalamos el CLI globalmente para poder crear un nuevo proyecto y levantarlo. Sigo la documentación para crear un nuevo proyecto[[45]](#footnote-46)

Esto nos permite generar un boilerplate o proyecto simple funcional a partir del cual comenzar a hacer cambios, partiendo de ahí limpio el proyecto e instalo algunas librerías como React Navigation, que nos permite navegar entre screens, para crear las dos primeras pantallas, la HomeScreen, donde se encontrará el menú de selección de idioma, poder acceder al ránking y comenzar un nuevo Quiz.

El CLI de Expo nos permite utilizar la aplicación Expo Go desde nuestro dispositivo móvil para ver los cambios de nuestro IDE en él en tiempo real, se puede utilizar en la misma red LAN o a través de tunneling escaneando un código QR, lo cual es estupendo si no tenemos un emulador en nuestro equipo o si queremos ver cómo quedará la app en nuestro dispositivo.

Una vez creadas las dos primeras pantallas con información estática, para continuar con el desplegable de selección de idioma, necesito conectarme con la base de datos para renderizar dinámicamente una lista de idiomas en los que está disponible realizar el Quiz, por lo que es el momento de crear nuestra aplicación de NestJS de la cual recoger esta información a partir de la API Rest.

Instalamos globalmente el CLI de NestJS e creamos un nuevo boilerplate utilizando el comando nest new.

Posteriormente usamos npx[[46]](#footnote-47) (Node Package Execute), el ejecutable que nos permite utilizar paquetes de Node.js sin instarlos localmente. Esto es especialmente útil para herramientas como CLI para scaffolding o que no requieren ocupar espacio del proyecto para ser utilizadas, ya que solo se utilizan durante el desarrollo. Probablemente uno de los paquetes de node más utilizados será create-react-app, cuya única funcionalidad es crear un boilerplate de una aplicación con React. Al no ser necesario instalarlo, utilizando npx evitamos instalarlo en nuestra máquina.

La siguiente tarea será crear un script para el seeding. Esto consiste en crear un archivo fuente en el cual se almacenará la información necesaria para que la aplicación funcione. Esto es un paso opcional dependiendo del cómo se plantee la aplicación, pero es algo muy positivo cuando existe información estática que quiere manejarse desde la base de datos. Esto hace que podamos levantar el proyecto sin necesidad de importar una base de datos o crearla manualmente, ya que la información se importa mientras se instalan las dependencias de la aplicación.

Prisma, el ORM que hemos escogido, permite el seeding creando un script, el cual puede ser en cualquier lenguaje de programación, pero usaremos TypeScript como en el resto del proyecto.

Una vez creado el script y teniendo la información lista para ser importada en una constante en formato JSON debemos indicar el comando “ts-node prisma/seed.ts” que corre el script para el seeding en nuestro package.json.

Una vez listo el script podemos ejecutar el comando “npx prisma db seed” para importar la información del código fuente a la base de datos a la que apunte nuestro archivo “.env”. El script imprimirá por consola el progreso de la importación y se cerrará cuando haya terminado.

Prisma ofrece un GUI para interactuar con nuestra base de datos, así que una vez haya finalizado el proceso de importación, podemos ejecutarlo utilizando el comando “npx prisma studio”. Esto nos abrirá un panel de gestión de nuestra base de datos, donde podremos comprobar que los datos se han importado correctamente.

Captura de pantalla de un celular

Descripción generada automáticamente

Figura 12: Prisma Studio es una interfaz gráfica que nos permite acceder a la información de la base de datos directamente en el navegador. En la captura se muestra como la información se ha importado correctamente a través del script de seeding.

Una vez comprobado que la base de datos cuenta con esa información, crearemos el primer Endpoint que devuelva una lista de los idiomas disponibles en los cuales puedes realizar el Quiz.

Para ello, NestJS nos permite usar su CLI para generar módulos, controladores y servicios. Normalmente un módulo envuelve los controladores y servicios con funcionalidades similares, y en este caso, todo podría encontrarse dentro del mismo ya que se trata de un proyecto que no tiene demasiadas dependencias.

Ejecutamos el comando “nest generate module languages” para generar un nuevo módulo, este se importará en nuestro módulo padre “app.module” automáticamente.

Un controlador en NestJS es un router, donde abriremos un Endpoint y especificaremos los requerimientos de la request, como método GET/POST, parámetros, cuerpo… etc. Un servicio es donde se debe alojar la lógica al llamar a un controlador, manejando input y output.

Como queremos que ambos, controlador y servicio se encuentren dentro del módulo de languages previamente creado, deberemos navegar hasta la carpeta que contenga dicho módulo, y podemos crear ambos utilizando los comandos “nest generate controller” y “nest generate service”. Al hacerlo dentro de la carpeta del módulo se importarán automáticamente en el mismo.

Como queremos utilizar Prisma en el servicio de languages, deberemos importarlo dentro del servicio antes de crear el primer Endpoint, al que se accederá con el método GET y te devolverá una lista de los idiomas disponibles en los que el usuario puede realizar el Quiz.

En el servicio, donde utilizaremos Prisma para hacer queries a nuestra base de datos, instanciamos el cliente de Prisma en el controlador

Para utilizar Prisma en NestJS, existe una práctica recomendada en su documentación[[47]](#footnote-48), que consiste en crear un servicio que posteriormente instanciaremos en otro servicio cuando queramos utilizar Prisma.

Prisma se conecta a la base de datos de forma asíncrona, por lo que es importante tener en cuenta que serán funciones asíncronas. Utilizaremos async/await para esperar a que la consulta haya finalizado. Teniendo esto en cuenta declaramos el primer método, que devuelva una lista de la tabla languages.

Una vez creado el método, podemos utilizar Postman para comprobar el output del método.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Figura 13: Podemos comprobar que el output del endpoint está conectado con nuestra base de datos

Ahora, con el primer endpoint funcionando, podemos volver a nuestra aplicación de Expo para apuntar a este endpoint y crear un menú de selección de idioma en el cual realizar el Quiz cuya información venga dinámicamente de la base de datos a través de la API Rest.

Instalaremos algunas librerías necesarias como dependencias en nuestra aplicación, “axios-hooks[[48]](#footnote-49)” y “axios”, que servirá para realizar request a la API, y “react-native-dropdown-picker[[49]](#footnote-50)”, que nos permitirá seleccionar uno de los idiomas del array obtenido y “react-native-dotenv[[50]](#footnote-51)” que nos permitirá usar variables de entorno para, por ejemplo, la URL de nuestra API Rest.

Una vez configurado el react-native-dotenv correctamente, podemos utilizar axios para hacer la llamada a la API usando la variable de entorno.

Una vez configurada la llamada a nuestro back-end en una primera instancia nos da fallo por CORS (Cross-origin resource sharing). Esto es un sistema de seguridad habilitado por defecto en todas las API Rest que evita que se hagan peticiones desde un dominio distinto a aquel donde está alojado el servicio. En este caso, la aplicación está corriendo en el puerto 19006 mientras que nuestro back-end en el puerto 3000, lo cual son dominios distintos. NestJS provee un método para activarlo sin necesidad de instalar una dependencia externa y permitir requests desde cualquier origen.

Para poder atacar a nuestra API Rest desde la red local, deberemos configurar el firewall de nuestro equipo para aceptar conexiones entrantes desde nuestro dispositivo móvil donde estemos probando la aplicación.

En un primer momento, en el seeder de lenguajes utilicé una ruta relativa a un icono para poder mostrarlo dinámicamente junto al idioma seleccionado dentro del dropdown de selección de idioma. Al parecer los assets han de estar disponibles en el momento de la compilación, por lo que si no están referenciados directamente en el código no se incluirá en el bundle. Uso un workaround utilizando un getter con un switch en el que queda la ruta *hardcodeada* para poder utilizar los iconos.

Una vez funcionando la primera ruta, despliego el proyecto en Heroku, con integración continua para que el proyecto se auto-despliegue cada vez que se haga un commit en la rama máster. Se crea un archivo Procfile para que cada vez que se despliegue, además de compilar el proyecto e iniciarlo, se ejecuten las migraciones en la base de datos de Heroku, la cual, al no estar expuesta a internet, no puedo manejar si no es a través del proyecto desplegado. Esto permite hacer cambios en el schema y que se desplieguen automáticamente, y también gestionar la lista de palabras disponibles para los Quiz simplemente añadiéndolas al código fuente y desplegando los cambios.

Posteriormente, habilito que los idiomas disponibles para el Quiz sean el inglés y francés, y añado una lista de palabras con la traducción en castellano para los Quiz.

En la aplicación híbrida, en la vista principal ahora aparece un botón para iniciar el Quiz y un dropdown que te permite elegir el idioma en el que quieres realizar el quiz: inglés y francés. El botón aparecerá como deshabilitado hasta que selecciones uno de los dos idiomas disponibles, y una vez seleccionado uno de los dos, dicho id de idioma será enviado como parámetro a la siguiente View de Iniciar el Quiz, para poder conservar el estado y conocer el idioma en el que se debe iniciar el mismo.

Una vez llegado al punto de iniciar el quiz, debo crear un nuevo endpoint que nos devuelva un array generado de forma aleatoria con las preguntas y respuestas.

Cada quiz tendrá 1 palabra en el idioma seleccionado y 4 opciones en castellano, una correcta y 3 erróneas.

Para eso hago una función que genere 40 números aleatorios comprendidos entre 1 y el número máximo de palabras que haya para el idioma seleccionado. Una vez con una lista que contenga los ids aleatorios de las palabras, los introducimos en un array para hacer una única query “where in” a la base de datos, sin necesidad de iterar en la función de Prisma.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente con confianza media

Figura 14: Estructura en JSON que se utilizará para renderizar un Quiz. La integración continua nos permite desplegar los cambios en Heroku automáticamente haciendo commit en la rama master.

Después, una vez generada la estructura, se comprueba que ninguna palabra está duplicada, ya que puede darse el caso que se ofrezcan dos veces la misma opción, la misma palabra, y una de ellas sea el resultado correcto y otra no. Esta respuesta deberá ser mapeada de nuevo para renderizar la estructura del Quiz, mostrando una pregunta y cuatro botones con opciones de respuesta, donde una de ellas será la correcta.

Creamos un componente reutilizable llamado Answerbutton, que contendrá un Pressable de React-native, un elemento que permite manejar un toque en el elemento por parte del usuario. Se puede estilizar del mismo modo que un botón estándar, permitiendo cambiar el estilo dinámicamente cuando el usuario está manteniéndolo apretado, lo cual es positivo en términos de experiencia de usuario, ya que ofrece feedback inmediato de la acción que estás realizando. Al pulsar comprobará que la respuesta elegida coincide con la pregunta, y es, por ende, la respuesta correcta. Si la respuesta es correcta se calculará un puntaje con respecto al tiempo que has tardado en responder, siendo un puntaje mayor al responder más rápido, y menor al responder más despacio.

El siguiente paso es crear una barra de progreso, el tiempo límite para responder cada una de las diez preguntas será de 10 segundos, pasado ese tiempo la pregunta contará como no respondida o errónea y no sumará puntos a la puntuación final. La barra de progreso ha de reiniciarse cada vez que cambie el índice del quiz, o bien, se avance a la siguiente pregunta, y, además, al acabarse la barra de progreso se deberá avanzar a la siguiente pregunta. Para renderizar la barra de progreso utilizo una animación Animated de React-native, esto permite que la animación sea mucho más suave que implementando una función que reste los segundos en un intervalo, aunque también implica que deberemos implementar también un contador cuyo valor sea accesible a través de una variable en el caso de querer utilizar los segundos transcurridos para generar una puntuación cuyo valor dependa de la velocidad a la que contestes.

Ahora tenemos dos formas de avanzar en las diez preguntas de cada quiz: a través de la pulsación en uno de los cuatro botones de respuesta, o dejando que el tiempo termine. Esto lo he conseguido generando un array e iterando con el índice de este, por lo que debemos controlar que el índice no se incremente más de lo necesario, ya que nos daría una excepción. Además, llegar al final del array significará que el Quiz ha finalizado, lo cual podremos utilizar posteriormente para navegar a otra screen y mostrar la puntuación obtenida.

También, modificamos el comportamiento por defecto de React-native en cuanto a la navegación, ya que en cualquier momento del Quiz podemos interrumpirlo pulsando el botón o gesto de “atrás” tanto en Android como en iOS. Cambiamos el comportamiento por defecto a mostrar una alerta que requiera confirmar salir de la aplicación en su totalidad, y no retroceder, y en el caso de aceptar, cerrar la aplicación.

Llegado a este punto, donde la mayor parte de la lógica de la aplicación está desarrollada, me centro en mejorar un poco la interfaz de usuario. Creo un componente llamado CustomButton, que quiero usar como un botón reutilizable con un estilo personalizado. Dicho botón tendrá distintas funcionalidades, como navegar entre screens y también, por ejemplo, guardar tu nickname en el ránking una vez has terminado el quiz, por lo que la función se declara fuera de la misma, y dentro de un Pressable de React-native bindeamos la función onPress con la función que le pasemos a través de props. Visualmente intento que se parezca a los cuatro botones del quiz para que la interfaz sea consistente en toda la aplicación.

En la API Rest ahora, creo un nuevo endpoint que admita el método post en el cual enviaremos la información para ser almacenada en el ránking: se guardan la puntuación obtenida en el test, el usuario o nickname introducido, el id del lenguaje en el que se ha realizado el test y las preguntas y respuestas que se han realizado. Una vez creado, en la screen Finish, debemos crear un InputText que permita al usuario escribir su nombre con el que quiere dejar guardada su puntuación en el quiz, para ello utilizamos el componente reutilizable mencionado antes, y damos también la opción de navegar de nuevo a la Home sin guardar la puntuación.

De nuevo en la home, controlamos que el nuevo botón utilizado para empezar el quiz se esconda para que solo aparezca en el caso de que hayas previamente seleccionado un idioma dentro del dropdown, y aprovechamos esa misma lógica para mostrar un botón que te permita acceder a la screen del ránking.

También creamos un nuevo endpoint en la API Rest que nos devuelva una lista de las puntuaciones guardadas, ordenadas por puntuación de mayor a menor.

Creamos la vista del ránking y utilizamos un ScrollView de React native para poder desplazarnos por la lista, ya que al haber más elementos de los que caben en pantalla deberemos ofrecer la opción de hacer scroll para navegar a través de ellos. Creamos un componente reutilizable y lo estilizamos para mostrar el puesto en el ránking, el usuario y la puntuación. También almacenamos la fecha en la que se guarda la puntuación en un timestamp, así que debemos utilizar la librería “date-fns”[[51]](#footnote-52) para formatear la fecha en el formato deseado. Una vez listo tenemos una lista en la que podemos ver todas las puntuaciones de mayor a menor, con la fecha en la que se realizó y el usuario que la guardó.

De nuevo en la screen del ránking, volvemos a nuestro componente de botón reutilizable para ofrecer la funcionalidad de volver atrás, quedando así el flujo de navegación cerrado.

Por último, instalamos una última librería llamada “React-native-toast-notifications”[[52]](#footnote-53) que nos permitirá mostrar un pequeño mensaje de feedback sobre si la respuesta que hemos dado al quiz era correcta o incorrecta según avancemos en el quiz.

Una vez terminado el desarrollo, añado una lista de palabras mayor al seed de palabras en inglés, y utilizo el traductor para traducirlas al francés. Compruebo que el idioma francés también funciona como debería y añado un tercer idioma para comprobar el planteamiento inicial para añadir idiomas programáticamente, sin necesidad de hacer cambios en el desarrollo.

# 8. Despliegue e instalación

El repositorio se trata de un monorepo que contiene dos proyectos, el primero es hybrid-app-source, y el segundo next-back-end-api.

Hybrid-app-source es un proyecto de Expo, un framework de React-native para crear aplicaciones híbridas para Android, iOS y la web.

Nest-back-end-api es un proyecto de NestJS, un framework de Express para crear aplicaciones de back-end y APIs Rest.

## 8.1 Versiones funcionales desplegadas:

* Interfaz de usuario gráfica, Texto, Aplicación

  Descripción generada automáticamenteEn el caso del proyecto Hybrid-app-source, hay una versión de producción subida en la plataforma Expo Snack. Para accer a ella es necesario instalar la aplicación Expo Go en nuestro smartphone y escanear el código QR que aparece en la página web. Para hacer un despliegue, (ha de hacerse manualmente) solo debes levantar el proyecto con el comando “expo-start”, y abrir los Expo Developer Tools. (localhost:19002).

Figura 15: Expo Devtools

La url donde se despliega el acceso a través de un código QR es https://expo.dev/@dawsnap/hybrid-app-tfg-davidsoriano

* En el caso del projecto Nest-back-end-api, hay una versión de producción subida en la plataforma Heroku, así que en el caso de no tener que realizar cambios en dicho servicio, podríamos apuntar a dicha URL pública y evitar tener que levantar todo el proyecto.

Nuestra instancia de Heroku está conectada con su propia base de datos y además tiene habilitada integración continua (CI) y migraciones automáticas, por lo que cada cambio efectuado en la rama master hará que el proyecto se despliegue automáticamente con los nuevos cambios, y que, al desplegarse, la base de datos se reinicie e introduzca de nuevo la información del seeding. Esto permite cambiar la información almacenada en la base de datos cambiando el archivo seed asegurándonos de que la información siempre está actualizada

La url de la API Rest es https://back-end-tfg-davidsoriano.herokuapp.com/

## 8.2 Ejecutar en local:

Ambos, al tratarse de proyectos de Node.js creados con npm, existe dentro de cada uno un archivo package-json con todas las dependencias necesarias para funcionar. Para instalar dichas dependencias y poder arrancar los proyectos en nuestra máquina local deberemos tener Node.js instalado y npm, y correr el comando “npm install”. Esto leerá dicho archivo e instalará las dependencias en una carpeta llamada node\_modules que siempre está excluida de git.

* En el caso del proyecto Hybrid-app-source, una vez instaladas las dependencias podemos utilizar CLI de expo para levantar el servicio en nuestro equipo. Necesitaremos un dispositivo móvil físico o un emulador para correr la aplicación, y que apunte a nuestra API Rest. En el caso de un dispositivo físico necesitaremos instalar la app Expo Go en nuestro dispositivo, la cual nos permitirá escanear un código QR para ejecutar la aplicación en el dispositivo.
* En el caso del proyecto Nest-back-end-api, una vez instaladas las dependencias podemos utilizar el CLI de NestJS para levantar el servicio en nuestro equipo. Para correr necesitará efectuar una conexión con una base de datos de Postgres, y previamente habremos realizado el proceso de seeding de nuestra base de datos para introducir la información necesaria para que la app funcione en nuestra base de datos.

Primero deberemos instalar PostgreSQL en nuestro equipo, o bien, levantarlo a través de Docker descargando la imagen oficial desde Dockerhub.

Una vez con una instancia de Docker corriendo, deberemos establecer la variable de entorno de la conexión que hemos creado en el proyecto y comprobar que se conecta a la base de datos levantando el servicio.

Cuando la conexión se haya efectuado correctamente, podemos pasar al proceso de seeding, o de introducción de datos estáticos en la base de datos. La información está almacenada dentro de unos ficheros en la carpeta Prisma, y contiene una lista de lenguajes habilitados en los que realizar el quiz, así como una lista de palabras y su significado en castellano.

Para preparar nuestra base de datos deberemos ejecutar los comandos “npx prisma db push” para generar el esquema con las tablas y columnas necesarias y posteriormente “npx prisma db seed” para introducir la información dentro de las tablas. Una vez realizado ese proceso la base de datos estará lista para conectarnos a ella.

# 9. Evolución y trabajo futuro

Una vez finalizada la aplicación, hay muchas tareas que podrían realizarse para aumentar las funcionalidades de la aplicación.

* Implementación de nuevos idiomas en los que realizar un quiz: La aplicación tiene disponibles tomar tests en francés e inglés, aunque se podrían añadir nuevos lenguajes programáticamente con tan solo añadir nueva información al seeder, y subiendo el icono del nuevo idioma en la aplicación.
* Implementación de autenticación, los usuarios podrían loguearse con su cuenta para luego implementar diversas funciones sociales,
* Nuevos niveles de dificultad, ya sea filtrando palabras por su nivel de dificultad, como reduciendo el tiempo disponible para contestar cada pregunta, aumentando a más de cuatro el número de respuestas posibles…
* Omitir palabras ya aprendidas
* Listado de palabras que hemos respondido y cuantas veces hemos acertado
* Funcionalidad para ver el total de palabras que nos han aparecido en un quiz alguna vez con respecto a todas las palabras disponibles en la aplicación.
* Formulario para sugerir una nueva combinación de palabra y significado en castellano
* Multilenguaje: Poder cambiar tu lengua principal y realizar un quiz sobre dicha lengua

# 10. Bibliografía

https://blog.busuu.com/how-long-does-it-take-to-learn-a-language/#:~:text=Based%20on%20our%20learners%20on, w. (s.f.). Obtenido de https://blog.busuu.com/how-long-does-it-take-to-learn-a-language/#:~:text=Based%20on%20our%20learners%20on,will%20bring%20the%20best%20results

https://blog.logrocket.com/best-typescript-orms/. (s.f.). Obtenido de https://blog.logrocket.com/best-typescript-orms/

https://blog.logrocket.com/what-is-deno/. (s.f.). Obtenido de https://blog.logrocket.com/what-is-deno/

https://blog.npmjs.org/post/615388323067854848/so-long-and-thanks-for-all-the-packages.html. (s.f.). Obtenido de https://blog.npmjs.org/post/615388323067854848/so-long-and-thanks-for-all-the-packages.html

https://date-fns.org/. (s.f.). Obtenido de https://date-fns.org/

https://docs.expo.dev/get-started/create-a-new-app/. (s.f.). Obtenido de https://docs.expo.dev/get-started/create-a-new-app/

https://docs.nestjs.com/recipes/prisma. (s.f.). Obtenido de https://docs.nestjs.com/recipes/prisma

https://en.wikipedia.org/wiki/Busuu. (s.f.). Obtenido de https://en.wikipedia.org/wiki/Busuu

https://en.wikipedia.org/wiki/Grand\_Master\_of\_Memory. (s.f.). Obtenido de https://en.wikipedia.org/wiki/Grand\_Master\_of\_Memory

https://en.wikipedia.org/wiki/Node.js. (s.f.). Obtenido de https://en.wikipedia.org/wiki/Node.js

https://en.wikipedia.org/wiki/Npm\_(software). (s.f.). Obtenido de https://en.wikipedia.org/wiki/Npm\_(software)

https://en.wikipedia.org/wiki/Ryan\_Dahl. (s.f.). Obtenido de https://en.wikipedia.org/wiki/Ryan\_Dahl

https://en.wikipedia.org/wiki/Spaced\_repetition. (s.f.). Obtenido de https://en.wikipedia.org/wiki/Spaced\_repetition

https://es.duolingo.com/plus. (s.f.). Obtenido de https://es.duolingo.com/plus

https://es.wikipedia.org/wiki/MySQL. (s.f.). Obtenido de https://es.wikipedia.org/wiki/MySQL

https://es.wikipedia.org/wiki/PostgreSQL. (s.f.). Obtenido de https://es.wikipedia.org/wiki/PostgreSQL

https://es.wikipedia.org/wiki/T%C3%A9cnicas\_de\_estudio. (s.f.). Obtenido de https://es.wikipedia.org/wiki/T%C3%A9cnicas\_de\_estudio

https://eu.usatoday.com/story/tech/columnist/talkingtech/2013/12/17/duolingo-apples-iphone-app-of-the-year/4042469/. (s.f.). Obtenido de https://eu.usatoday.com/story/tech/columnist/talkingtech/2013/12/17/duolingo-apples-iphone-app-of-the-year/4042469/

https://expo.dev/. (s.f.). Obtenido de https://expo.dev/

https://expressjs.com/. (s.f.). Obtenido de https://expressjs.com/

https://github.com/arnnis/react-native-toast-notifications. (s.f.). Obtenido de https://github.com/arnnis/react-native-toast-notifications

https://hackr.io/blog/nodejs-frameworks. (s.f.). Obtenido de https://hackr.io/blog/nodejs-frameworks

https://help.busuu.com/hc/es/articles/360000351117--Qu%C3%A9-es-la-prueba-de-nivel-. (s.f.). Obtenido de https://help.busuu.com/hc/es/articles/360000351117--Qu%C3%A9-es-la-prueba-de-nivel-

https://kinsta.com/blog/is-php-dead/#:~:text=According%20to%20W3Techs'%20data%2C%20PHP, u. (s.f.). Obtenido de https://kinsta.com/blog/is-php-dead/#:~:text=According%20to%20W3Techs'%20data%2C%20PHP,using%20PHP%20in%20some%20way

https://stackoverflow.com/questions/1279613/what-is-an-orm-how-does-it-work-and-how-should-i-use-one. (s.f.). Obtenido de https://stackoverflow.com/questions/1279613/what-is-an-orm-how-does-it-work-and-how-should-i-use-one

https://techcrunch.com/2020/09/21/language-learning-service-babbel-says-it-has-now-sold-over-10m-subscriptions/. (s.f.). Obtenido de https://techcrunch.com/2020/09/21/language-learning-service-babbel-says-it-has-now-sold-over-10m-subscriptions/

https://www.babbel.com/en/magazine/how-much-time-study-each-day-learn-language#:~:text=The%20Bottom%20Line, r. (s.f.). Obtenido de https://www.babbel.com/en/magazine/how-much-time-study-each-day-learn-language#:~:text=The%20Bottom%20Line,right%20track%20toward%20language%20proficiency

https://www.envisagedigital.co.uk/wordpress-market-share/#:~:text=WordPress%20powers%2037%%20of%20all%20websites%20and%2062%%20of%20CMS%20sites%20alone. (s.f.). Obtenido de https://www.envisagedigital.co.uk/wordpress-market-share/#:~:text=WordPress%20powers%2037%%20of%20all%20websites%20and%2062%%20of%20CMS%20sites%20alone

https://www.geeksforgeeks.org/what-are-the-differences-between-npm-and-npx/#:~:text=NPX%3A%20The%20npx%20stands%20for, w. (s.f.). Obtenido de https://www.geeksforgeeks.org/what-are-the-differences-between-npm-and-npx/#:~:text=NPX%3A%20The%20npx%20stands%20for,without%20even%20installing%20that%20package.

https://www.guru99.com/what-is-mongodb.html. (s.f.). Obtenido de https://www.guru99.com/what-is-mongodb.html

https://www.heavy.ai/technical-glossary/relational-database. (s.f.). Obtenido de https://www.heavy.ai/technical-glossary/relational-database

https://www.heroku.com/pricing. (s.f.). Obtenido de https://www.heroku.com/pricing

https://www.iflexion.com/blog/top-hybrid-mobile-app-frameworks. (s.f.). Obtenido de https://www.iflexion.com/blog/top-hybrid-mobile-app-frameworks

https://www.makeuseof.com/make-the-most-of-duolingo-lessons/#:~:text=To%20optimize%20your%20learning%2C%20aim, o. (s.f.). Obtenido de https://www.makeuseof.com/make-the-most-of-duolingo-lessons/#:~:text=To%20optimize%20your%20learning%2C%20aim,on%20the%20app%20each%20day.&text=If%20you're%20struggling%20to,five%20more%20in%20the%20evening

https://www.memrise.com/about#:~:text=Founded%20in%202010%2C%20Memrise%20has, w. (s.f.). Obtenido de https://www.memrise.com/about#:~:text=Founded%20in%202010%2C%20Memrise%20has,with%20the%20world%20around%20them.

https://www.meteor.com/. (s.f.). Obtenido de https://www.meteor.com/

https://www.mongodb.com/databases/non-relational. (s.f.). Obtenido de https://www.mongodb.com/databases/non-relational

https://www.muylinux.com/2021/08/04/nuevo-instalador-ubuntu. (s.f.). Obtenido de https://www.muylinux.com/2021/08/04/nuevo-instalador-ubuntu

https://www.npmjs.com/~nestjscore. (s.f.). Obtenido de https://www.npmjs.com/~nestjscore

https://www.npmjs.com/package/axios-hooks. (s.f.). Obtenido de https://www.npmjs.com/package/axios-hooks

https://www.npmjs.com/package/react-native-dotenv. (s.f.). Obtenido de https://www.npmjs.com/package/react-native-dotenv

https://www.npmjs.com/package/react-native-dropdown-picker. (s.f.). Obtenido de https://www.npmjs.com/package/react-native-dropdown-picker

https://www.plainconcepts.com/es/app-nativa-vs-hibrida-ventajas-y-desventajas/#:~:text=La%20principal%20ventaja%20de%20la, m. (s.f.). *https://www.plainconcepts.com/es/app-nativa-vs-hibrida-ventajas-y-desventajas/#:~:text=La%20principal%20ventaja%20de%20la,market%E2%80%9D%20es%20mucho%20m%C3%A1s%20reducido*.

https://www.pluralsight.com/blog/software-development/relational-vs-non-relational-databases. (s.f.). Obtenido de https://www.pluralsight.com/blog/software-development/relational-vs-non-relational-databases

https://www.postgresql.org/. (s.f.). Obtenido de https://www.postgresql.org/

https://www.prisma.io/. (s.f.). Obtenido de https://www.prisma.io/

https://www.prisma.io/dataguide/postgresql/benefits-of-postgresql. (s.f.). Obtenido de https://www.prisma.io/dataguide/postgresql/benefits-of-postgresql

https://www.rosettastone.com/product/faq. (s.f.). Obtenido de https://www.rosettastone.com/product/faq

https://www.sql-workbench.eu/dbms\_comparison.html. (s.f.). Obtenido de https://www.sql-workbench.eu/dbms\_comparison.html

*https://www.statista.com/statistics/1248025/duolingo-dau-worldwide/#:~:text=In%202021%2C%20language%20learning%20mobile,million%20daily%20active%20users%20worldwide.* (s.f.). Obtenido de https://www.statista.com/statistics/1248025/duolingo-dau-worldwide/#:~:text=In%202021%2C%20language%20learning%20mobile,million%20daily%20active%20users%20worldwide.

1. (https://en.wikipedia.org/wiki/Spaced\_repetition, s.f.) [↑](#footnote-ref-2)
2. (https://www.statista.com/statistics/1248025/duolingo-dau-worldwide/#:~:text=In%202021%2C%20language%20learning%20mobile,million%20daily%20active%20users%20worldwide., s.f.) [↑](#footnote-ref-3)
3. (https://eu.usatoday.com/story/tech/columnist/talkingtech/2013/12/17/duolingo-apples-iphone-app-of-the-year/4042469/, s.f.) [↑](#footnote-ref-4)
4. (https://en.wikipedia.org/wiki/Spaced\_repetition, s.f.) [↑](#footnote-ref-5)
5. (https://es.duolingo.com/plus, s.f.) [↑](#footnote-ref-6)
6. (https://en.wikipedia.org/wiki/Grand\_Master\_of\_Memory, s.f.) [↑](#footnote-ref-7)
7. (https://en.wikipedia.org/wiki/Spaced\_repetition, s.f.) [↑](#footnote-ref-8)
8. (https://www.memrise.com/about#:~:text=Founded%20in%202010%2C%20Memrise%20has, s.f.) [↑](#footnote-ref-9)
9. (https://techcrunch.com/2020/09/21/language-learning-service-babbel-says-it-has-now-sold-over-10m-subscriptions/, s.f.) [↑](#footnote-ref-10)
10. (https://help.busuu.com/hc/es/articles/360000351117--Qu%C3%A9-es-la-prueba-de-nivel-, s.f.) [↑](#footnote-ref-11)
11. (https://en.wikipedia.org/wiki/Busuu, s.f.) [↑](#footnote-ref-12)
12. (https://es.wikipedia.org/wiki/T%C3%A9cnicas\_de\_estudio, s.f.) [↑](#footnote-ref-13)
13. (https://www.makeuseof.com/make-the-most-of-duolingo-lessons/#:~:text=To%20optimize%20your%20learning%2C%20aim, s.f.) [↑](#footnote-ref-14)
14. (https://www.rosettastone.com/product/faq, s.f.) [↑](#footnote-ref-15)
15. (https://www.babbel.com/en/magazine/how-much-time-study-each-day-learn-language#:~:text=The%20Bottom%20Line, s.f.) [↑](#footnote-ref-16)
16. (https://blog.busuu.com/how-long-does-it-take-to-learn-a-language/#:~:text=Based%20on%20our%20learners%20on, s.f.) [↑](#footnote-ref-17)
17. (https://www.plainconcepts.com/es/app-nativa-vs-hibrida-ventajas-y-desventajas/#:~:text=La%20principal%20ventaja%20de%20la, s.f.) [↑](#footnote-ref-18)
18. (https://www.muylinux.com/2021/08/04/nuevo-instalador-ubuntu, s.f.) [↑](#footnote-ref-19)
19. (https://www.iflexion.com/blog/top-hybrid-mobile-app-frameworks, s.f.) [↑](#footnote-ref-20)
20. (https://expo.dev/, s.f.) [↑](#footnote-ref-21)
21. (https://en.wikipedia.org/wiki/Node.js, s.f.) [↑](#footnote-ref-22)
22. (https://en.wikipedia.org/wiki/Ryan\_Dahl, s.f.) [↑](#footnote-ref-23)
23. (https://en.wikipedia.org/wiki/Npm\_(software), s.f.) [↑](#footnote-ref-24)
24. (https://blog.npmjs.org/post/615388323067854848/so-long-and-thanks-for-all-the-packages.html, s.f.) [↑](#footnote-ref-25)
25. (https://blog.logrocket.com/what-is-deno/, s.f.) [↑](#footnote-ref-26)
26. (https://hackr.io/blog/nodejs-frameworks, s.f.) [↑](#footnote-ref-27)
27. (https://expressjs.com/, s.f.) [↑](#footnote-ref-28)
28. (https://www.meteor.com/, s.f.) [↑](#footnote-ref-29)
29. (https://www.npmjs.com/~nestjscore, s.f.) [↑](#footnote-ref-30)
30. (https://www.pluralsight.com/blog/software-development/relational-vs-non-relational-databases, s.f.) [↑](#footnote-ref-31)
31. (https://www.heavy.ai/technical-glossary/relational-database, s.f.) [↑](#footnote-ref-32)
32. (https://www.mongodb.com/databases/non-relational, s.f.) [↑](#footnote-ref-33)
33. (https://es.wikipedia.org/wiki/MySQL, s.f.) [↑](#footnote-ref-34)
34. (https://www.envisagedigital.co.uk/wordpress-market-share/#:~:text=WordPress%20powers%2037%%20of%20all%20websites%20and%2062%%20of%20CMS%20sites%20alone, s.f.) [↑](#footnote-ref-35)
35. (https://kinsta.com/blog/is-php-dead/#:~:text=According%20to%20W3Techs'%20data%2C%20PHP, s.f.) [↑](#footnote-ref-36)
36. (https://www.guru99.com/what-is-mongodb.html, s.f.) [↑](#footnote-ref-37)
37. (https://www.postgresql.org/, s.f.) [↑](#footnote-ref-38)
38. (https://es.wikipedia.org/wiki/PostgreSQL, s.f.) [↑](#footnote-ref-39)
39. (https://www.prisma.io/dataguide/postgresql/benefits-of-postgresql, s.f.) [↑](#footnote-ref-40)
40. (https://www.sql-workbench.eu/dbms\_comparison.html, s.f.) [↑](#footnote-ref-41)
41. (https://stackoverflow.com/questions/1279613/what-is-an-orm-how-does-it-work-and-how-should-i-use-one, s.f.) [↑](#footnote-ref-42)
42. (https://blog.logrocket.com/best-typescript-orms/, s.f.) [↑](#footnote-ref-43)
43. (https://www.prisma.io/, s.f.) [↑](#footnote-ref-44)
44. (https://www.heroku.com/pricing, s.f.) [↑](#footnote-ref-45)
45. (https://docs.expo.dev/get-started/create-a-new-app/, s.f.) [↑](#footnote-ref-46)
46. (https://www.geeksforgeeks.org/what-are-the-differences-between-npm-and-npx/#:~:text=NPX%3A%20The%20npx%20stands%20for, s.f.) [↑](#footnote-ref-47)
47. (https://docs.nestjs.com/recipes/prisma, s.f.) [↑](#footnote-ref-48)
48. (https://www.npmjs.com/package/axios-hooks, s.f.) [↑](#footnote-ref-49)
49. (https://www.npmjs.com/package/react-native-dropdown-picker, s.f.) [↑](#footnote-ref-50)
50. (https://www.npmjs.com/package/react-native-dotenv, s.f.) [↑](#footnote-ref-51)
51. (https://date-fns.org/, s.f.) [↑](#footnote-ref-52)
52. (https://github.com/arnnis/react-native-toast-notifications, s.f.) [↑](#footnote-ref-53)