ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA (ICAI)

GRADO EN INGENIERÍA TELEMÁTICA

INTRODUCCIÓN DE LA INTELIGENCIA ARTIFICIAL EN LA METODOLOGÍA DE APRENDIZAJE INDIVIDUALIZADA

Autor: Marcos Jiménez Gutiérrez

Director: Jaime Pereña Pinedo

Madrid

Junio 2018

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título Introducción de la inteligencia artificial en la metodología de enseñanza individualizada en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el curso académico 2017/18 es de mi autoría, original e inédito y no ha sido presentado con anterioridad a otros efectos.

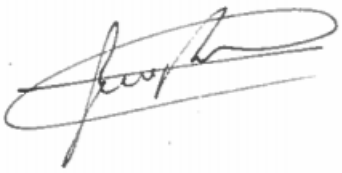
El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido tomada de otros documentos está debidamente referenciada.

Fdo.: Marcos Jiménez Gutiérrez Fecha: 16/ 08/ 2018

Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO

Fdo.: Jaime Pereña Pinedo Fecha: 24/08/2018



AUTORIZACIÓN PARA LA DIGITALIZACIÓN, DEPÓSITO Y DIVULGACIÓN EN RED DE PROYECTOS FIN DE GRADO, FIN DE MÁSTER, TESINAS O MEMORIAS DE BACHILLERATO

1º. Declaración de la autoría y acreditación de la misma.

El autor D.Marcos Jiménez Gutiérrez

DECLARA ser el titular de los derechos de propiedad intelectual de la obra: Introducción de la inteligencia artificial en la metodología de aprendizaje individualizado, que ésta es una obra original, y que ostenta la condición de autor en el sentido que otorga la Ley de Propiedad Intelectual.

2º. Objeto y fines de la cesión.

Con el fin de dar la máxima difusión a la obra citada a través del Repositorio institucional de la Universidad, el autor **CEDE** a la Universidad Pontificia Comillas, de forma gratuita y no exclusiva, por el máximo plazo legal y con ámbito universal, los derechos de digitalización, de archivo, de reproducción, de distribución y de comunicación pública, incluido el derecho de puesta a disposición electrónica, tal y como se describen en la Ley de Propiedad Intelectual. El derecho de transformación se cede a los únicos efectos de lo dispuesto en la letra a) del apartado siguiente.

3º. Condiciones de la cesión y acceso

Sin perjuicio de la titularidad de la obra, que sigue correspondiendo a su autor, la cesión de derechos contemplada en esta licencia habilita para:

a) Transformarla con el fin de adaptarla a cualquier tecnología que permita incorporarla a internet y hacerla accesible; incorporar metadatos para realizar el registro de la obra e incorporar "marcas de agua" o cualquier otro sistema de seguridad o de protección.
b) Reproducirla en un soporte digital para su incorporación a una base de datos electrónica, incluyendo el derecho de reproducir y almacenar la obra en servidores, a los efectos de garantizar su seguridad, conservación y preservar el formato.
c) Comunicarla, por defecto, a través de un archivo institucional abierto, accesible de modo libre y gratuito a través de internet.
d) Cualquier otra forma de acceso (restringido, embargado, cerrado) deberá solicitarse expresamente y obedecer a causas justificadas.
e) Asignar por defecto a estos trabajos una licencia Creative Commons.
f) Asignar por defecto a estos trabajos un HANDLE (URL *persistente).*

4º. Derechos del autor.

El autor, en tanto que titular de una obra tiene derecho a:

a) Que la Universidad identifique claramente su nombre como autor de la misma
b) Comunicar y dar publicidad a la obra en la versión que ceda y en otras posteriores a través de cualquier medio.
c) Solicitar la retirada de la obra del repositorio por causa justificada.
d) Recibir notificación fehaciente de cualquier reclamación que puedan formular terceras personas en relación con la obra y, en particular, de reclamaciones relativas a los derechos de propiedad intelectual sobre ella.

5º. Deberes del autor.

El autor se compromete a:

a) Garantizar que el compromiso que adquiere mediante el presente escrito no infringe ningún derecho de terceros, ya sean de propiedad industrial, intelectual o cualquier otro.
b) Garantizar que el contenido de las obras no atenta contra los derechos al honor, a la intimidad y a la imagen de terceros.
c) Asumir toda reclamación o responsabilidad, incluyendo las indemnizaciones por daños, que pudieran ejercitarse contra la Universidad por terceros que vieran infringidos sus derechos e intereses a causa de la cesión.

d) Asumir la responsabilidad en el caso de que las instituciones fueran condenadas por infracción de derechos derivada de las obras objeto de la cesión.

6º. Fines y funcionamiento del Repositorio Institucional.

La obra se pondrá a disposición de los usuarios para que hagan de ella un uso justo y respetuoso con los derechos del autor, según lo permitido por la legislación aplicable, y con fines de estudio, investigación, o cualquier otro fin lícito. Con dicha finalidad, la Universidad asume los siguientes deberes y se reserva las siguientes facultades:

- La Universidad informará a los usuarios del archivo sobre los usos permitidos, y no garantiza ni asume responsabilidad alguna por otras formas en que los usuarios hagan un uso posterior de las obras no conforme con la legislación vigente. El uso posterior, más allá de la copia privada, requerirá que se cite la fuente y se reconozca la autoría, que no se obtenga beneficio comercial, y que no se realicen obras derivadas.
- La Universidad no revisará el contenido de las obras, que en todo caso permanecerá bajo la responsabilidad exclusive del autor y no estará obligada a ejercitar acciones legales en nombre del autor en el supuesto de infracciones a derechos de propiedad intelectual derivados del depósito y archivo de las obras. El autor renuncia a cualquier reclamación frente a la Universidad por las formas no ajustadas a la legislación vigente en que los usuarios hagan uso de las obras.
- La Universidad adoptará las medidas necesarias para la preservación de la obra en un futuro.
- La Universidad se reserva la facultad de retirar la obra, previa notificación al autor, en supuestos suficientemente justificados, o en caso de reclamaciones de terceros.

Madrid, a 18 de Agosto de 2018

ACEPTA

Fdo Marcos Jiménez

Motivos para solicitar el acceso restringido, cerrado o embargado del trabajo en el Repositorio Institucional:

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA (ICAI)

GRADO EN INGENIERÍA TELEMÁTICA

INTRODUCCION DE LA INTELIGENCIA ARTIFICIAL EN LA METODOLOGÍA DE ENSEÑANZA INDIVIDUALIZADA

Autor: Marcos Jiménez Gutiérrez

Director: Jaime Pereña Pinedo

Madrid

Agosto 2018

Agradecimientos

Doy en esta página mi gratitud a todas las personas que me han ayudado a lo largo del desarrollo de este proyecto, para que el paso del tiempo no me permita olvidarlas.

Mi agradecimiento más especial a **D. Jaime Pereña Pinedo**, mi tutor en este trabajo, que ha seguido muy de cerca, con su profesionalidad, entrega, cercanía y afecto, la evolución de mi trabajo. Además de sus valiosas observaciones, a él le debo la oportunidad de haber podido conocer la sede de Microsoft en Seattle y a las personas que allí viven la Inteligencia Artificial como un reto de vida. Todas estas personas que me ofrecieron su su tiempo para escuchar mi idea de proyecto y darme sus opiniones sobre la manera adecuada de desarrollar el mismo. Gracias a **Cesar de la Torre, Paul Stubbs**, **Giampaolo Battaglia**, **Luis Cabrera**, **Carlos Castro**, **Bo Yan**, **Anupam Vij** y **Riham Mansour**.

Mi agradecimiento al elenco de profesores de ICAI que han contribuido a mi formación universitaria, por su permanente estímulo a mi trabajo. A la comunidad universitaria en su conjunto que hace posible que nuestra escuela sea nuestra casa.

A mis amigos por soportar mis ausencias necesarias para que todo este trabajo pudiera salir adelante.

Por último, quiero dar las gracias a mis padres y a mis hermanos **Alejandro** y **Nicolás** sin cuyo apoyo, paciencia y comprensión no habría podido escribir ni un renglón de este proyecto.

INTRODUCCIÓN DE LA INTELIGENCIA ARTIFICIAL EN LA METODOLOGÍA DEL APRENDIZAJE INDIVIDUALIZADO

Autor: D. Marcos Jiménez Gutiérrez.
Director: D. Jaime Pereña Pinedo.
Entidad Colaboradora: ICAI – Universidad Pontificia Comillas

RESUMEN DEL PROYECTO

Durante el desarrollo del proyecto se ha llevado a cabo el diseño e implementación de un sistema experto con la intención de mejorar las habilidades de aprendizaje, aplicado sobre un método de iniciación al aprendizaje básico de lectura. Para ello se ha creado un Bot conversacional que es capaz de reconocer la voz y emociones del usuario con las cuales guiarle a través de una serie de ejercicios de lectura.

Palabras clave: IA, Bot, Reconocimiento de voz, Reconocimiento de emociones, enseñanza.

1. Introducción

La inteligencia artificial es una tecnología surgida hace pocos años y que actualmente tiene una infinidad de aplicaciones en ámbitos como las finanzas, la industria o la medicina. La enseñanza es otro terreno en la que las nuevas tecnologías se está empezando a introducir. Desafortunadamente, la presencia de la IA en el mundo de la educación solo aparece de forma teórica.

Según expertos en pedagogía, el uso de esta tecnología bien aplicada puede conllevar grandes beneficios para el proceso de enseñanza y el aprendizaje. El profesorado dispondría de ayuda sobre que técnicas o material usar en distintos contextos, mientras que los alumnos contarían con recursos para el aprendizaje que se adaptaran a su forma de pensar y razonar. Esto, visto en el conjunto, genera una enseñanza en la cual la personalización es la prioridad, y cada estudiante dispondría del proceso de enseñanza adaptado a sus cualidades

2. Definición del proyecto

El objetivo principal del proyecto es el desarrollo de una solución basada en inteligencia artificial, aplicada a la docencia, con las siguientes características:

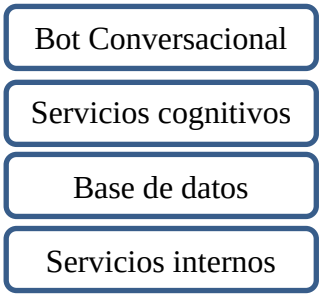
- Personalizada: Tiene que ser capaz de analizar las emociones de los estudiantes, adaptando el discurso de la actividad a realizar a estas emociones.
- Interactiva: Con esta aplicación se tiene que conseguir que los usuarios se involucren en la actividad lectora para mantener la atención y mejorar los resultados del ejercicio
- Divertida: La inclusión de la enseñanza en las nuevas tecnologías implica el acercamiento de este proceso a los medios de entretenimiento del usuario,

permitiendo cambiar la concepción del aprendizaje y consiguiendo mezclar los juegos y la enseñanza en uno.

Para conseguir estos objetivos se desarrollará un Bot conversacional, que pueda guiar al usuario a través de la actividad a desarrollar. Este Bot consigue que el aprendizaje sea interactivo, haciendo que el proceso de aprendizaje parezca una conversación. Para conseguir que la enseñanza sea personalizada, se incluye un módulo de reconocimiento de emociones en el Bot. Este módulo es capaz de reconocer emociones de una foto, mediante las cuales tomara decisiones respecto a la longitud del ejercicio y la dificultad de este. Con este módulo también se consigue decidir cuando el usuario no está implicado en el ejercicio, que tome decisiones sobre las pausas para realizar otros tipos de actividades lúdicas, y luego volver a la aplicación.

3. Descripción de la solución

En la Ilustración 1 se puede apreciar la estructura de la solución diseñada e implementada para cumplir los objetivos descritos anteriormente que se compone de los siguientes bloques:



1. Bot Conversacional
   El marco en el que se desarrolla el proyecto es el de un bot conversacional que permita desarrollar la actividad de enseñanza similar a una conversación. La tecnología usada (Bot Framework de Microsoft), nos permite crear diálogos, por los cuales los usuarios avanzan. En este caso, se han creado dos diálogos principales: un dialogo de autenticación del usuario y otro de desarrollo del ejercicio.
   El dialogo de autenticación se lanza automáticamente al conectar con el Bot. Este dialogo permite la autenticación de usuarios existentes y el registro de nuevos usuarios a la aplicación
   Tras la autenticación, se procede a lanzar el dialogo principal, durante el cual se van dando ejercicios al usuario para que los lea. En este dialogo el Bot usa los servicios cognitivos e internos para la personalización del ejercicio.

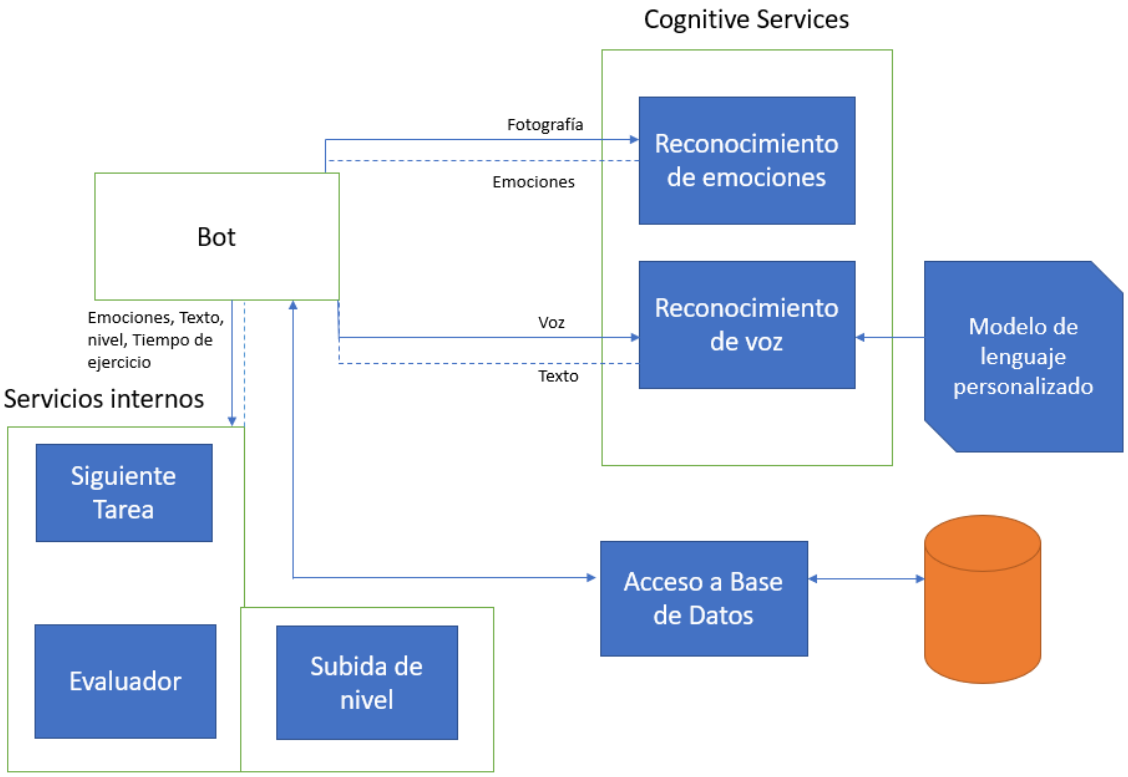


Ilustración 1 - Esquema de los componentes que conforman la solución

2. Servicios cognitivos

   Estos servicios son un conjunto de redes neuronales desarrolladas y entrenadas por Microsoft, los cuales permiten el uso de estos mediante suscripciones de pago. Esta solución usa dos modelos: El servicio de reconocimiento de emociones y el servicio de conversión de voz a texto.

   El servicio de reconocimiento de emociones permite recibir un listado de emociones con sus correspondientes coeficientes tras mandarle una fotografía al servicio

   Con el servicio de conversión de voz a texto se puede reconocer lo que lee cada usuario tras recibir el texto a leer. Este modelo neuronal es necesario reentrenarlo con un modelo de lenguaje personalizado, que permite reconocer los posibles fallos de lectura del usuario, aparte de posibles palabras que no están incluidas en el diccionario.

3. Base de datos

   La base de datos de esta aplicación consiste de las siguientes tablas:

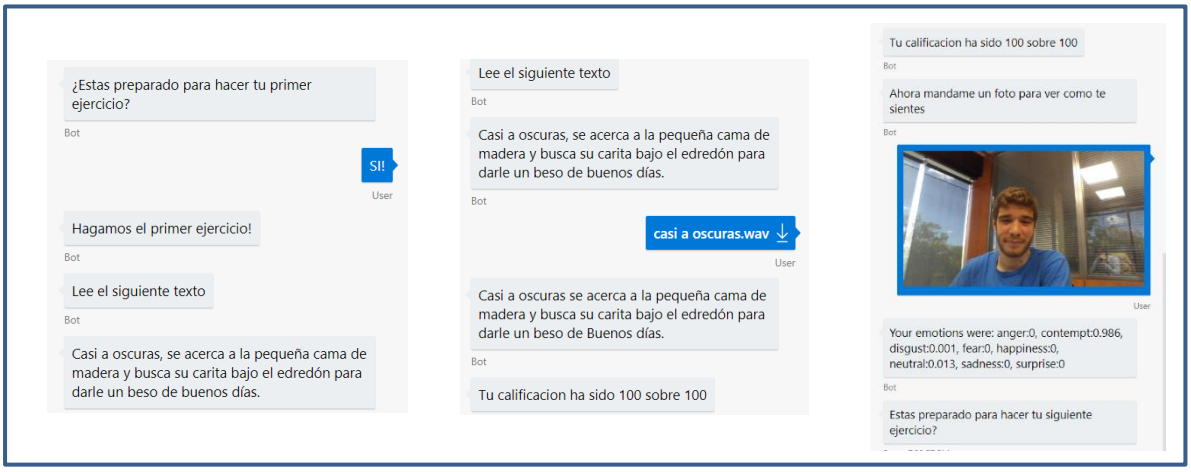
   - Usuarios: Permite mantener un registro de usuarios y su nivel de lectura
   - Ejercicios: Se almacenan los ejercicios que se realizaran por los usuarios junto con su nivel de dificultad
   - Registro de actividad: En esta tabla se almacenan un registro de los ejercicios realizados por cada usuario junto con la calificación obtenida. Esto permite decidir que ejercicios asignar a los usuarios y cuando estos suben de nivel.

4. Servicios internos
   Estos servicios son consisten de un conjunto de reglas que deciden como adaptar el desarrollo de la actividad al usuario.
   - *SiguienteTarea*: Este módulo decide qué nivel de ejercicio se realizara tras la finalización del anterior. Para esto tiene en cuenta las emociones del usuario, su calificación y el número de ejercicios que ha realizado.
   - *Evaluador*: El Bot invoca a este servicio para que evalúe la actividad que ha realizado el usuario. Para ello, este evaluador hace una comparativa a nivel de palabra del ejercicio realizado y el original.
   - *SubidaDeNivel*: Al final de un módulo de aprendizaje, el Bot invoca a este módulo para decidir si el usuario ha adquirido un nivel necesario como para realizar ejercicios de un nivel superior de forma consistente.

Mediante estos componentes, la solución propone ejercicios de lectura al usuario, el cual los completa consecutivamente. Cuando el componente *SiguienteTarea* decida, el módulo acaba. Esto normalmente conlleva un total de seis ejercicios de lectura por módulo.

4. Resultados

Aunque es imposible conseguir resultados sobre la mejoría de las habilidades lecturas de un usuario durante un periodo de tiempo corto, si se ha podido observar como los usuarios son capaces de realizar las actividades propuestas consistentemente y el Bot toma decisiones correctas con los datos recogidos



Capturas de simulaciones de un modulo de aprendizaje

5. Conclusiones

Se puede concluir que la solución desarrollada cumple los objetivos que se le habían impuesto como necesarios y por lo tanto se observa que la IA está suficientemente avanzada como para aparecer en la enseñanza y contribuir a esta de manera significativa. En este caso se ha usado como sistemas maestros, el cual puede llegar a ser una herramienta que no solo enseñe a leer, sino que se puede aplicar a otras disciplinas como las matemáticas, la historia o la programación. Aparte de esto, existen

una gran variedad de formas de aprender como el descubrimiento guiado, los cuales gracias a la IA podrán ser posibles sin la necesidad de una persona para enseñar.

6. Referencias

[1] *Begoña Gross*. *La inteligencia artificial y su aplicación en la enseñanza*. Origen: CL&E, 1992, 13, pp. 73-80

7. Recursos

[1] Documentación Bot framework: *https://docs.microsoft.com/en-us/azure/bot-service/*

[2] Documentación servicios cognitivos: *https://docs.microsoft.com/es-es/azure/cognitive-services/*

INTRODUCTION OF ARTIFICIAL INTELIGENCE TO INDIVIDUALISED LEARNING METHOLOGIES

Author: D. Marcos Jiménez Gutiérrez.
Supervisor: D. Jaime Pereña Pinedo.
Collaborating entity: ICAI – Universidad Pontificia Comillas

ABSTRACT

During the development of this project, the design and implementation of an expert system which will improve the reading abilities of the user has been done. For this purpose, a conversational bot which can recognize both the voice and emotions of the user, has been created. With these capabilities, the bot guides the user across a group of reading exercises.

Keywords: AI, bot, voice recognition, emotion recognition, teaching process.

1. Introduction

AI is a relatively new technology which has infinite uses in areas like economy, industry or medicine. AI is also starting to appear in the world of education, although unfortunately, this presence is only theorical.

According to pedagogic experts, the use of this new technology will result in a great advantage for the teaching and learning process. While the teaching body would be advised about what exercise or methodology to use in each situation, the students would have to their disposition adapted exercises to their way of learning and thinking. All these advantages generate a new kind of education, personalized around the student and its capabilities.

2. Project definition

The main objective of this project was the development of an AI based solution applied to teaching which fulfilled the following requirements:

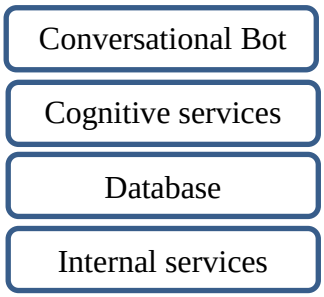
- Personalized: The solution had to be able to analyze the user's emotions, adapting the exercises to be done.
- Interactive: It had to involve the user into the reading process to maintain focus on the exercise and therefore, achieve better results.
- Fun: By implementing teaching into the new technologies, the learning and entertainment worlds get closer. This allows to overcome the common mistake that learning is boring while been able to mix games and exercises in the same activity.

To achieve these goals, a conversational that can guide the user through the exercises has been developed. This bot allows the learning process to be interactive, making the whole length of exercises as if it was a conversation. To make the teaching personalized, an emotion recognition module has been included in the solution, which

allows to recognize user's emotions from pictures and act according to them, deciding the length and difficulty of the exercises. Finally, this module also allows the solution to know when the user is interested in the exercises and advise the user to do other activities and later come back again to the reading exercises.

3. Solution description

In picture 1 it can be seen the structure of the solution with all the different components that allow to achieve the goals described before. This solution is formed by the following blocks:

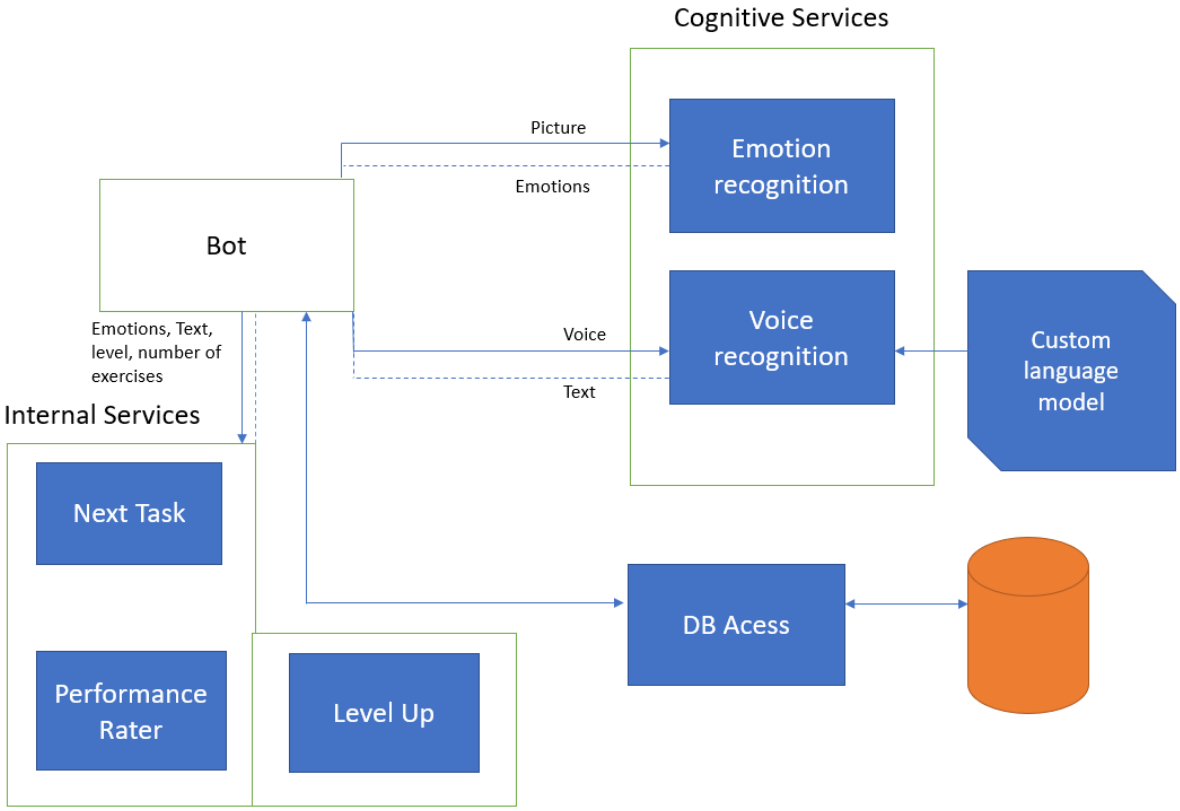


1. Conversational bot

   The frame in which the solution is developed is a conversational bot that permits to navigate through the exercise as in a conversation. The technology used (Bot framework by Microsoft) allows the possibility to create dialogs through which the users advance. For this purpose, two dialogs have been created: an authentication dialog and the exercise flow dialog.

   The authentication dialog launches automatically when the user connects to the bot. This dialog allows the authentication of existing users and the register of new users to the app.

   After authentication, the user is redirected to the exercise flow dialog. During this dialog, the user is been given exercises to read. In this dialog, the bot uses different cognitive services to personalize the exercise.



Picture 3 – Schema of the main components of the solution

2. Cognitive services

   These services represent a group of neural networks created and trained by Microsoft, who allow the use of them through paid subscriptions. The solution uses two of these services: EmotionAPI, which allows emotion recognition, and SpeechToText for voice recognition

   EmotionAPI permits a list of the user's emotions with percentage coefficients when a picture is being sent to the service.

   By using SpeechToText service, it is possible to know what the user read from a given exercise. It is needed to train again this neural network with a custom language model. This new model is created from possible mistakes the user can make during the reading exercise and some words that may not appear in the universal model.

3. Database

   The solution database consists of the following tables:

   - Users: Contains each user with their reading level.
   - Exercises: The exercises are stored in this table, with the text and its level of difficulty.
   - Activity Log: whenever a user solves an exercise, a log of the exercise id, the user id and the performance achieved by the user is saved in this table.

This allows to decided future exercises for the user and when he/she is ready to do exercises of the next level.

4. Internal services

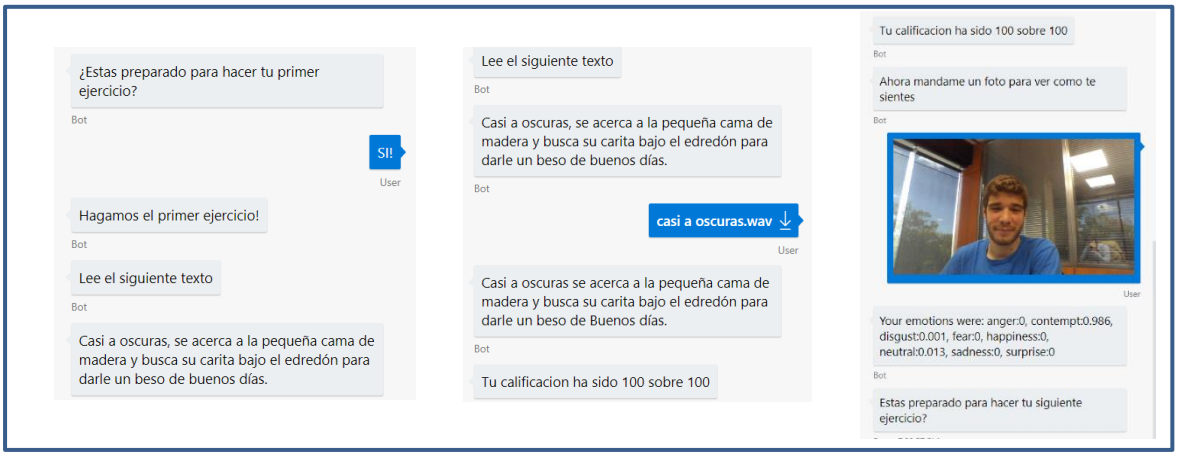
   These services consist of a group of rules that make the solution be able to decide how to adapt the exercises to the user.

   - *NextTask*: This module decided the level of the exercise that the user will do next. It considers the user's emotions, his/her performance, the number of exercises already done.
   - *PerformanceRater*: The bot calls this service to evaluate how well a user did an exercise. For this purpose, the module carries on a word level comparative between the original exercise and the one done.
   - *LevelUp*: At the end of a learning module, the solution calls this module to decide whether the user has acquired enough skill to do exercises of a higher level consistently. If that is the case, the user will step up and in the next learning module will do exercises of one level higher.

All these components allow the solution to give the user reading exercises, who completes them consequently. When *NextTask* module decides so, the learning module ends. This happens after the user has done around six exercises.

4. Results

Although it is impossible to obtain result concerning the improvement of the reading skills of some user, is has been possible to see how the users can do the exercises proposed and how the solution takes correct decisions according to the data that it gathered.



Picture 4 – Captures of a simulation of a learning module

5. Conclusions

It can be stated that the solution developed achieves the goals that were considered necessary, and therefore, it is a fact that AI is sufficiently evolved to appear in teaching and learning methodologies and contribute to them. In the case of the project, AI helped build an expert system, which can not only help to learn how to read, but also can help to develop abilities in different learning areas such as mathematics, history or programming. More so, there are many other learning methods like guided discovery where AI could even be necessary for them to work properly.

6. References

[1] *Begoña Gross. La inteligencia artificial y su aplicación en la enseñanza.* Source: CL&E, 1992, 13, pp. 73-80

7. Resources

[1] Bot framework documentation: *https://docs.microsoft.com/en-us/azure/bot-service/*

[2] Cognitive services documentation: *https://docs.microsoft.com/es-es/azure/cognitive-services/*

Índice de la memoria

Índice de figuras

Índice de tablas

Capítulo 1. INTRODUCCIÓN

La Inteligencia artificial es una tecnología que ha surgido hace pocos años y que actualmente tiene una infinidad de aplicaciones en ámbitos como las finanzas, la industria o la medicina. La enseñanza es otro terreno en la que las nuevas tecnologías se están empezando a introducir. Desafortunadamente, la presencia de la IA en el mundo de la educación solo aparece de forma teórica.

Según expertos en pedagogía, el uso de esta tecnología adecuadamente aplicada puede conllevar grandes beneficios para el proceso de enseñanza y aprendizaje. El profesorado dispondría de ayuda sobre que técnicas o material usar para distintos alumnos, mientras que los alumnos contarían con unos recursos para el aprendizaje que se adapten a su forma de pensar y razonar. Esto visto en el conjunto, genera una enseñanza en la cual la personalización es la prioridad, y cada estudiante dispondría del proceso de enseñanza adaptado a sus cualidades.

Dentro de la inteligencia artificial, la rama que nos permite trasladar el mundo de la enseñanza a un entorno tecnológico son los sistemas expertos. Estos sistemas consisten en un conjunto de conceptos, procedimientos y técnicas que permiten asistir en el análisis de problemas complejos, buscando simular el comportamiento de un experto en un campo específico del conocimiento, en este caso el profesor y la enseñanza. Para conseguir la capacidad de entendimiento del profesor, el campo de Machine Learning ayuda a identificar los comportamientos de los alumnos, así como las respuestas correctas ante diferentes carencias cognitivas de los alumnos.

Aparte de la IA, otras tecnologías nuevas son necesarias para que la aplicación funcione y los módulos de inteligencia artificial dispongan de suficiente información para analizar: El reconocimiento de voz es esencial para que una aplicación sea capaz de leer. Sin esta capacidad, nunca podría saber si el estudiante está leyendo correctamente.

Además, se está desarrollando actualmente el reconocimiento de intención, lo cual permite recaudar información sobre el estado de ánimo.

Para poder saber cómo se siente el estudiante tras los fallos o aciertos en los ejercicios, la complejidad de estos, o la duración de estudio respecto a descansos, es necesario analizar las emociones de estos. Para este objetivo, Microsoft ha desarrollado una API la cual aporta información sobre el estado de ánimo de las personas basándose en una fotografía facial.

1.1 MOTIVACIÓN DEL PROYECTO

Tras observar las tecnologías emergentes, se puede concluir que se tienen suficientes recursos como para empezar a desarrollar aplicaciones que sean capaces de analizar a los alumnos casi como lo hace un profesor y así generar un proceso de aprendizaje más personalizado hacia este. Aparte, se han analizado extensamente los beneficios que la IA aporta a la educación, y, sin embargo, no existe ninguna aplicación real que extrapole todos estos conocimientos y beneficios a la práctica.

Por esto, mi trabajo de fin de grado consistirá en una aplicación informática que, ayudándose del reconocimiento por voz, reconocimiento de emociones y IA, consiga potenciar y personalizar el proceso de aprendizaje de la lectura.

Capítulo 2. ESTADO DE LA CUESTIÓN

Como ya se ha afirmado anteriormente, la existencia de inteligencia artificial en aplicaciones educativas es nula. En la actualidad, las aplicaciones de uso educativo se basan principalmente en repositorios de ejercicios desde los cuales los alumnos con acceso pueden realizarlos o descargarlos. A pesar de la posibilidad de la presencia de la IA en estas aplicaciones para aspectos como las recomendaciones o ejercicios guiados, la presencia de esta aun no es conocida.

La reciente introducción de material informático personal, como Tablets, en el entorno escolar, ha permitido el crecimiento de este tipo de aplicaciones informáticas, cuyos propietarios no parecen ver más allá de la publicación de sus libros en estas.

En referencia a la pedagogía, numerosos artículos hablan sobre las posibilidades de la inteligencia artificial. Los autores en general están de acuerdo una afirmación: La inteligencia artificial, propiamente aplicada, va a conllevar un gran beneficio en la enseñanza y el aprendizaje. En su opinión, la inteligencia artificial va a revolucionar los métodos de enseñanza actuales, llevándolos a donde los docentes siempre han querido, la enseñanza centrada en el alumno y sus capacidades.

Pero no solo van a revolucionar los métodos actuales, sino que aparecerán otros cuyos beneficios potenciarán habilidades difíciles de mejorar en el sistema educativo actual. Por ejemplo, métodos como el descubrimiento guiado o los micromundos podrán existir de manera consistente a un coste bajo, y permitirán el desarrollo de dotes de investigación desde edades anteriores a la universidad.

Capítulo 3. DEFINICIÓN DEL TRABAJO

3.1 JUSTIFICACIÓN

Tras todo esto, se llega a dos distintas conclusiones:

3.1.1 AVANCES TECNOLÓGICOS

Los avances tecnológicos actuales ya permiten el reconocimiento de voz o el reconocimiento de emociones. Estos son fácilmente accesibles gracias a los servicios proporcionados por grandes empresas como Microsoft o Amazon, los cuales ponen a disponibilidad del consumidor a través de sus plataformas de servicios en la nube.

Este tipo de servicios son necesarios para el desarrollo de sistemas maestros en el ámbito de la enseñanza de lectura. Sin ellos, es imposible la personalización del proceso de aprendizaje al alumno.

3.1.2 EXISTENCIA DE APLICACIONES DE ENSEÑANZA DE LECTURA

No existe ninguna aplicación informática comercializada que no solo funcione como repositorio, sino que sea capaz de decidir los ejercicios a realizar por el estudiante y actúe en función de sus resultados y emociones.

Por lo tanto, esta aplicación puede no solo ser un gran avance en la relación entre la inteligencia artificial y la educación, sino que convertiría a la empresa que la lanzara en una referencia en estos aspectos.

3.2 OBJETIVOS

Esta aplicación está enfocada a mejorar las habilidades de lectura de alumnos en la etapa de primaria, por lo tanto, la aplicación buscará un proceso de enseñanza con las siguientes características:

- Personalizada: La aplicación será capaz de analizar las emociones del estudiante, adaptando la duración del tiempo de trabajo y los descansos, y los conocimientos adquiridos, centrándose en completar los conocimientos del estudiante.
- Interactiva: Actualmente, para aprender a leer, los niños se ven obligados a leer libros en casa. Con esta aplicación, la actividad será la misma, pero los estudiantes serán capaces de saber si están cometiendo errores, aparte de realizar ejercicios acordes a su nivel actual.
- Divertida: Las nuevas generaciones usan muchos dispositivos electrónicos para el entretenimiento. Mediante el traslado de la educación a uno de estos dispositivos, nos permite crear un aprendizaje que incluya juegos que mantengan al estudiante interesado en el proceso.

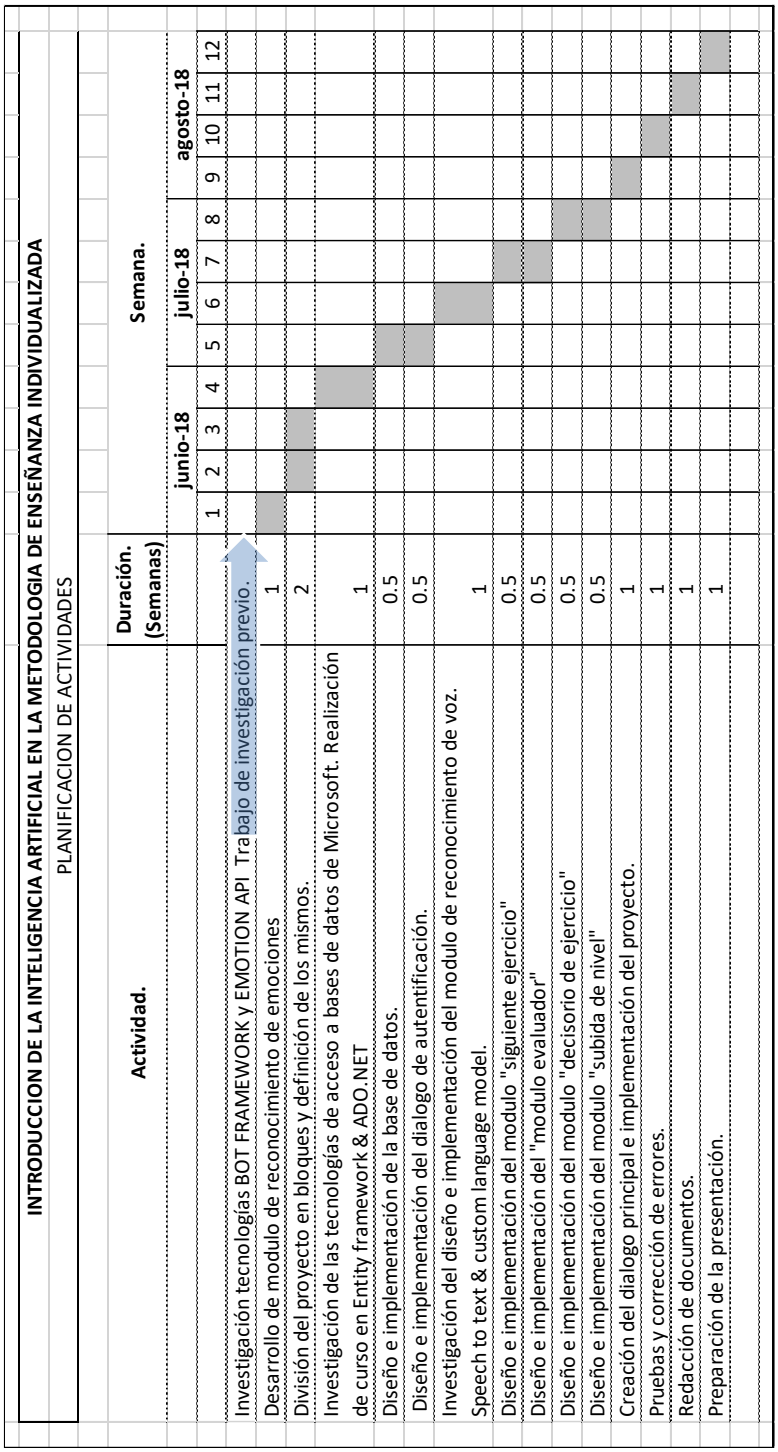
3.3 METODOLOGÍA

Previo al trabajo de diseño y gestión, es necesario dedicar tiempo a la investigación de las tecnologías necesarias para el proyecto.

Para la correcta creación de una solución informática, existen muchos pasos a seguir. En el caso de esta solución, se usará una metodología de diseño ágil, basada en dividir el proyecto en módulos. Tras esto, cada uno de los módulos será diseñado, implementado y probado a la vez. Finalmente, es necesaria una etapa de implementación final, en la que se pondrán a trabajar todos los módulos juntos y se probara la solución general.

Adicionalmente, esta metodología nos permite adecuar los periodos de trabajo y modificar la planificación si es necesario.

3.4 PLANIFICACIÓN



3.5 ESTIMACIÓN ECONÓMICA

El único coste que incurre del desarrollo de este proyecto es el coste de contratación de los servicios en la nube de Microsoft. Los 4 servicios a contratar para el desarrollo y pruebas de la solución son:

- Speech Service: Existe un nivel de suscripción que permite el acceso a este servicio gratuitamente si no se consumen mas de 10h de traducción al mes. Si este limite se supera, el usuario será cobrado según el uso (aproximadamente 0.70 €/hora de traducción más 20€ del modelo de lenguaje).
- EmotionAPI: Al igual que en SpeechService, existe una suscripción gratuita que permite el uso del servicio mientras no se superen 20 llamadas por minuto o 30.000 llamadas al mes. En caso contrario se cobrará al usuario 0.84 €/1000 llamadas
- Base de datos: se contrata el servicio básico de bases de datos en la nube, que conlleva una base de datos de máximo 2GB con una DTU de 5. El coste de esto es de 5 €/mes

Según estimaciones realizadas, no se espera superar las cuotas de uso de los servicios cognitivos y por tanto el coste total del desarrollo del proyecto seria de 10€ por los dos meses de uso de la base de datos.

Capítulo 4. DISEÑO DEL SISTEMA

4.1 OBJETIVOS DEL PROYECTO

Para conseguir los objetivos impuestos en el apartado anterior, se ha decidido usar un Bot conversacional como base del proyecto. Sobre este Bot se montará la actividad que realizara el usuario. Esta actividad consiste en una serie de ejercicios de lectura que el usuario hará consecutivamente.

Al conectar con el Bot, el usuario lo primero que realiza es autenticarse en la aplicación, para que la solución pueda recoger la información del usuario y así adecuar los ejercicios que se presentan a este.

Tras esto, el Bot presentará uno a uno los ejercicios a realizar por el usuario. El usuario debe mandar el audio al chatbot para que este pueda calificar el ejercicio realizado y guardar un registro de la actividad.

Entre ejercicio y ejercicio, también es necesario recoger las emociones que presenta el usuario. Esto nos permite usar más información del usuario y tomar decisiones concretas respecto de estas acciones.

La consecución de ejercicios acabará cuando la aplicación lo decida. Esta decisión estará influenciada por las emociones del usuario y el número de ejercicios realizados por este.

4.2 DISEÑO DE BLOQUES

Para conseguir los objetivos específicos del proyecto, se ha diseñado la solución dividida en bloques representada en la figura 1.

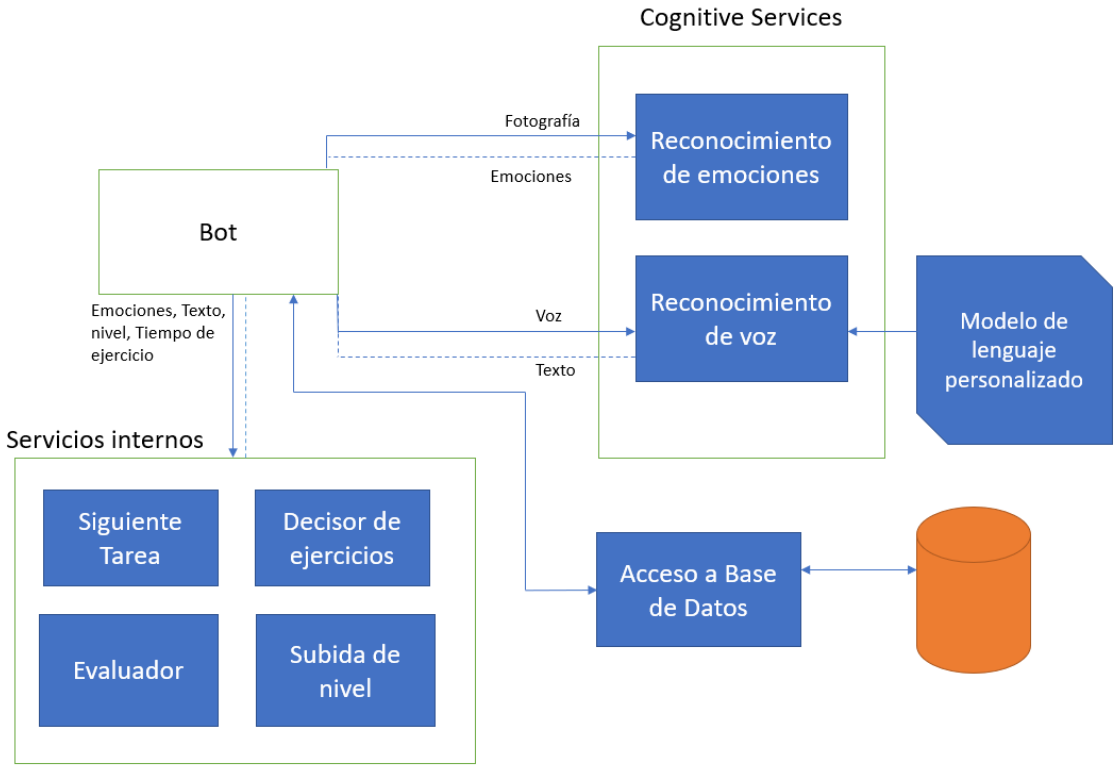


Figura 1: Esquema de bloques de la solución

La solución consta de tres bloques principales:

- Bloque de servicios cognitivos: Este bloque consta de las llamadas a los servicios cognitivos que permiten a la aplicación tener la capacidad de obtener información sobre las emociones del usuario y recoger en texto la lectura del ejercicio por parte del usuario
- Base de Datos: Necesaria en todo tipo de aplicaciones, en nuestro caso es crucial para almacenar información sobre los ejercicios del usuario y así poder usar esta información para personalizar la experiencia del usuario durante el uso de la solución

- Servicios internos: Este bloque consta de las clases necesarias para el correcto funcionamiento de la aplicación. Los distintos servicios internos consistirán a modo de ejemplo, desde un bloque para tomar decisiones sobre el nivel del ejercicio, hasta un bloque capaz de evaluar el ejercicio del usuario.

4.3 ASPECTOS PRIMARIOS PARA EL DESARROLLO E IMPLEMENTACIÓN DE LA SOLUCIÓN DISEÑADA

4.3.1 EL LENGUAJE DE PROGRAMACIÓN

Para el desarrollo de este proyecto, la tecnología base que se ha usado es C#. El uso de este lenguaje de programación nos da múltiples ventajas en contraste con otros como Java. En primer lugar, nos permite desarrollar el Bot usando la tecnología desarrollada por Microsoft, Bot Framework. Este framework permite el desarrollo de bots conversacionales sin tener que desarrollar múltiples clases e interfaces que en Java no existen. Aparte de esto, usando este framework, existe una Interfaz de Usuario (UI) creada para probar y conectar con bots de manera sencilla. La existencia de esta interfaz gráfica es importante aunque el alcance de este proyecto no contempla el desarrollo de dicha interfaz.

4.3.2 SERVICIOS EN LA NUBE

Otro aspecto a tener en cuenta es la posibilidad de usar servicios en la nube. Mientras que montar la solución entera en un servidor propio implica una dificultad adicional a la hora de implementar el proyecto, los servicios en la nube nos permiten crear bases de datos y otro tipo de servicios en poco tiempo que presentan la misma funcionalidad que los montados en un servidor propio.

Para hacer uso de servicios en la nube, se ha elegido el portal Azure de Microsoft. Este portal nos permite generar recursos en la nube bajo suscripciones de pago. En este portal se crearan los siguientes recursos para el desarrollo del proyecto:

1. Servidor con Windows server: Aunque no se necesitara configurar nada en el servidor, este es necesario para crear recursos de bases de datos en el portal.
2. Base de Datos SQL: Estará montada en el servidor. Se ha elegido SQL por ser la tecnología más común y con mayor información de uso disponible.
3. Servicio de reconocimiento de caras: Este servicio (FaceAPI) es el que permite obtener las emociones a partir de la foto de un usuario.
4. Servicio de reconocimiento de voz: Este servicio (SpeechService) permite obtener un texto que corresponde a un audio del usuario.

En la figura 2 se puede observar el grupo de recursos LearningAI, el cual contiene todos los servicios descritos anteriormente.

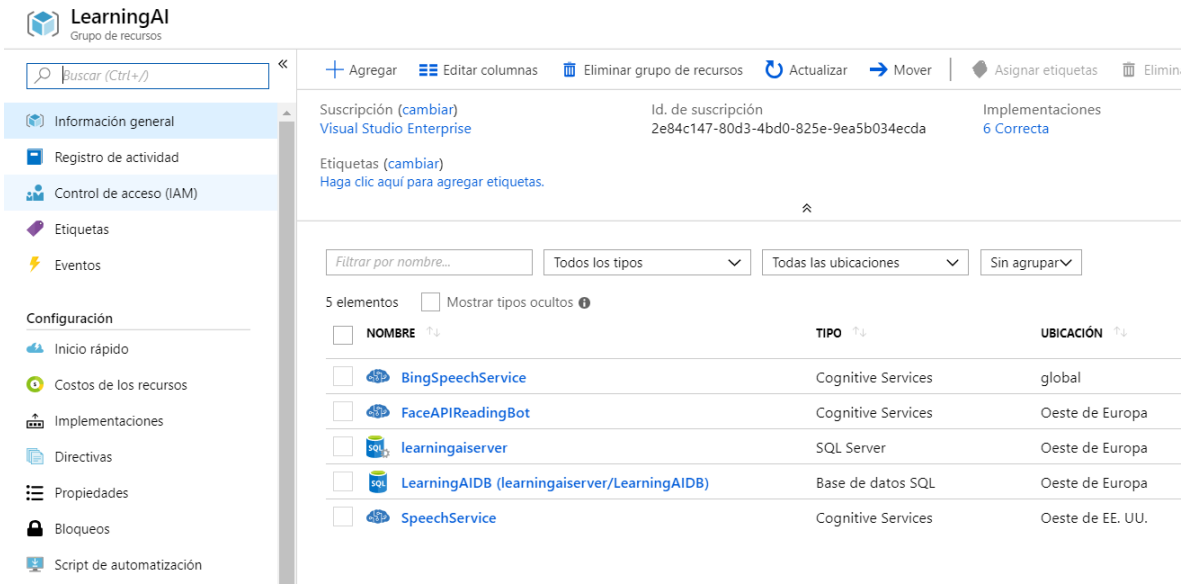


Figura 2: Grupo de recursos del proyecto en el portal Azure

Aparte de los recursos descritos anteriormente, es también necesario el uso de un portal llamado *cris.ai*. En este portal es donde se crean los modelos de lenguaje personalizados que se han visto necesarios durante el diseño del proyecto.

Capítulo 5. DESARROLLO DEL BOT Y DIÁLOGOS

5.1 SOBRE BOT FRAMEWORK

Un bot es un programa informático que repite ciertas tareas. En el caso de la solución objeto de este proyecto se crea un bot conversacional, los cuales se encargan de guiar al usuario a través de conversaciones preestablecidas llamadas diálogos. Para dialogar con este tipo de bots, se conecta al bot a través de http, y cada elemento de la conversación que el usuario quiere enviarle al bot es otra petición http.

En esta solución se usa un bot conversacional porque hace que el programa sea interactivo. La realización de ejercicios no se lleva a cabo fríamente, sino que gracias a esta tecnología, el usuario obtiene la sensación de mantener una conversación real con la máquina, consiguiendo un mayor nivel de atención.

Aparte de los diálogos, Bot Framework tiene otra clase importante llamada Controller. Esta clase se encarga de crear las distintas rutas de acceso a distintas partes del bot. Si el bot por ejemplo tiene un apartado de consultas y otro de atención al cliente, estos dos apartados se nombran en los controladores del bot para crear los distintos accesos http. En este proyecto solo se usa un controlador, ya que la única parte del bot es la parte de realización de ejercicios.

5.1.1 CREACIÓN DEL BOT

La creación del bot que se usa como base del proyecto consiste en el uso de una plantilla básica de Bot Framework que contiene un controlador y un dialogo principal. A partir de esta plantilla se ha desarrollado toda la solución, incluyendo los distintos bloques en esta.

5.2 DIÁLOGOS DE LA SOLUCIÓN

Como ya se ha descrito antes, los diálogos son las clases que forman la conversación a través de la cual el usuario se comunica con el bot. Estas clases son el corazón de la solución porque es donde se utiliza toda la funcionalidad creada en los distintos bloques de la solución.

Esta solución consta de dos diálogos:

5.2.1 DIALOGO DE AUTENTICACIÓN

Este dialogo nos permite reconocer al usuario cuando se conecta con el bot. Por lo tanto, el dialogo se lanza automáticamente al conectarse con el bot. Las tareas de este dialogo son:

1. Recoger la información del usuario: En el caso de la solución desarrollada, la autenticación es simple. Se usa el nombre y apellidos del usuario para reconocerle. Por lo tanto, lo primero que tiene que hacer este dialogo es recabar esta información
2. Recoger la información del usuario de la base de datos: Una vez el usuario ha dado la información al bot, este realiza una consulta en la base de datos para obtener toda la información del usuario
3. Añadir nuevos usuarios: Si es la primera vez que el usuario se conecta al bot, entonces es necesario crearle un nuevo perfil

En la figura 3 se puede observar un diagrama de flujo del dialogo.

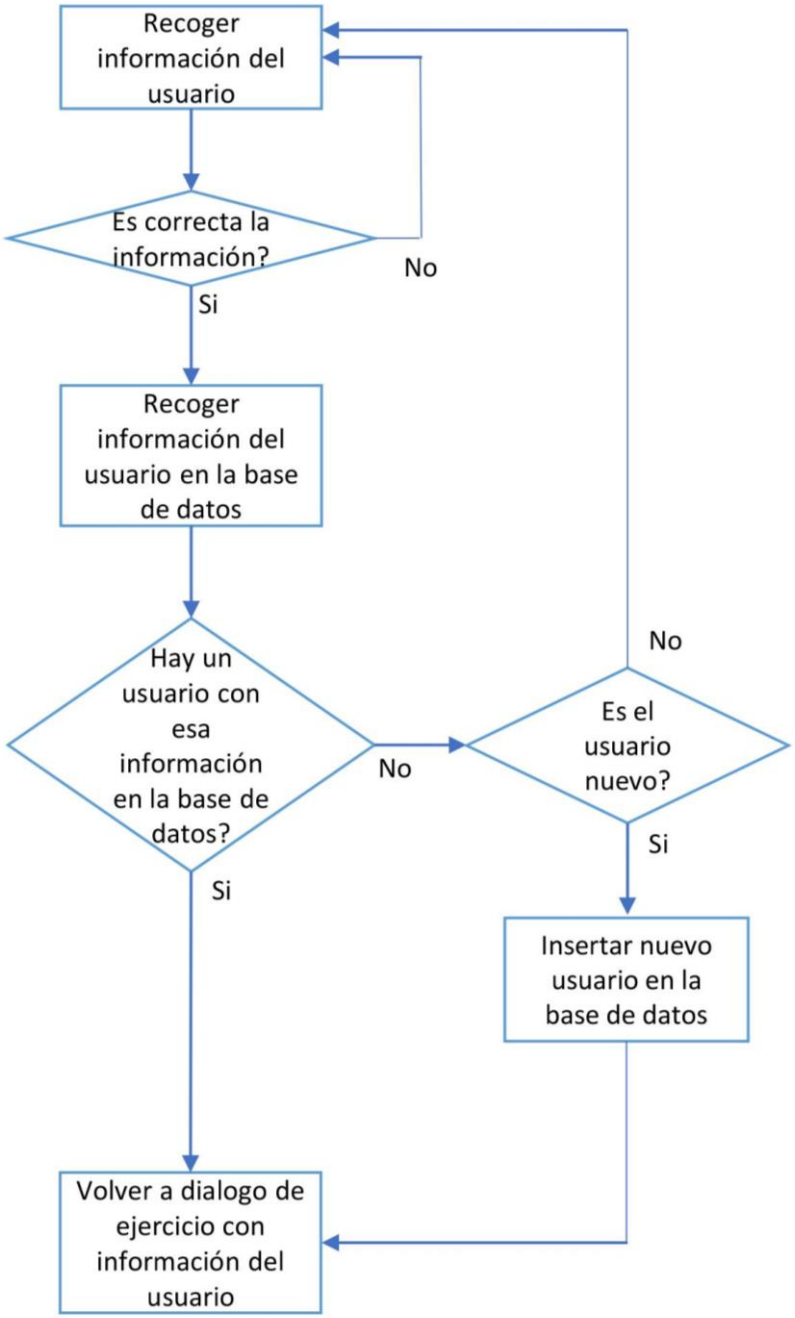


Figura 3: Diagrama de flujo del Dialogo de autenticación

5.2.2 DIALOGO DE RESOLUCIÓN DE EJERCICIOS

Este dialogo es el componente principal de la solución, dado que es en el que se presentan y resuelven los ejercicios de lectura. Por lo tanto, también hace uso de casi toda la funcionalidad implementada.

Este dialogo se invoca tras resolver la identificación del usuario. Cuando ya se ha obtenido la información en el dialogo de autenticación, este dialogo se termina y le pasa al diálogo de ejercicios la información del usuario recogida de la base de datos en forma de una clase definida.

Tras esto, el dialogo comienza presentando el primer ejercicio al usuario. En cada iteración de presentar el ejercicio, el dialogo tiene que realizar los siguientes pasos:

1. Calcular el nivel del ejercicio a realizar.
2. Obtener un ejercicio de la base de datos acorde al nivel y los ejercicios pasados del usuario.
3. Presentar el ejercicio al usuario.
4. Recibir el fichero de audio del usuario con el ejercicio resuelto y convertirlo a texto.
5. Evaluar el resultado del ejercicio, basándose en la comparación de la conversión de voz a texto con el ejercicio propuesto.
6. Guardar un registro del ejercicio realizado.
7. Obtener una foto del usuario para medir las emociones de este.
8. Recibir las emociones.
9. Decidir en función de todo lo anterior el nivel del ejercicio siguiente o si por el contrario, el usuario a realizado suficientes ejercicios y puede descansar.
10. Decidir si el usuario debe hacer ejercicios de un nivel superior.

En la figura 4 se puede observar el diagrama de flujo de este dialogo junto con los bloques unidos a las funciones, que desempeñan las tareas en cada situación

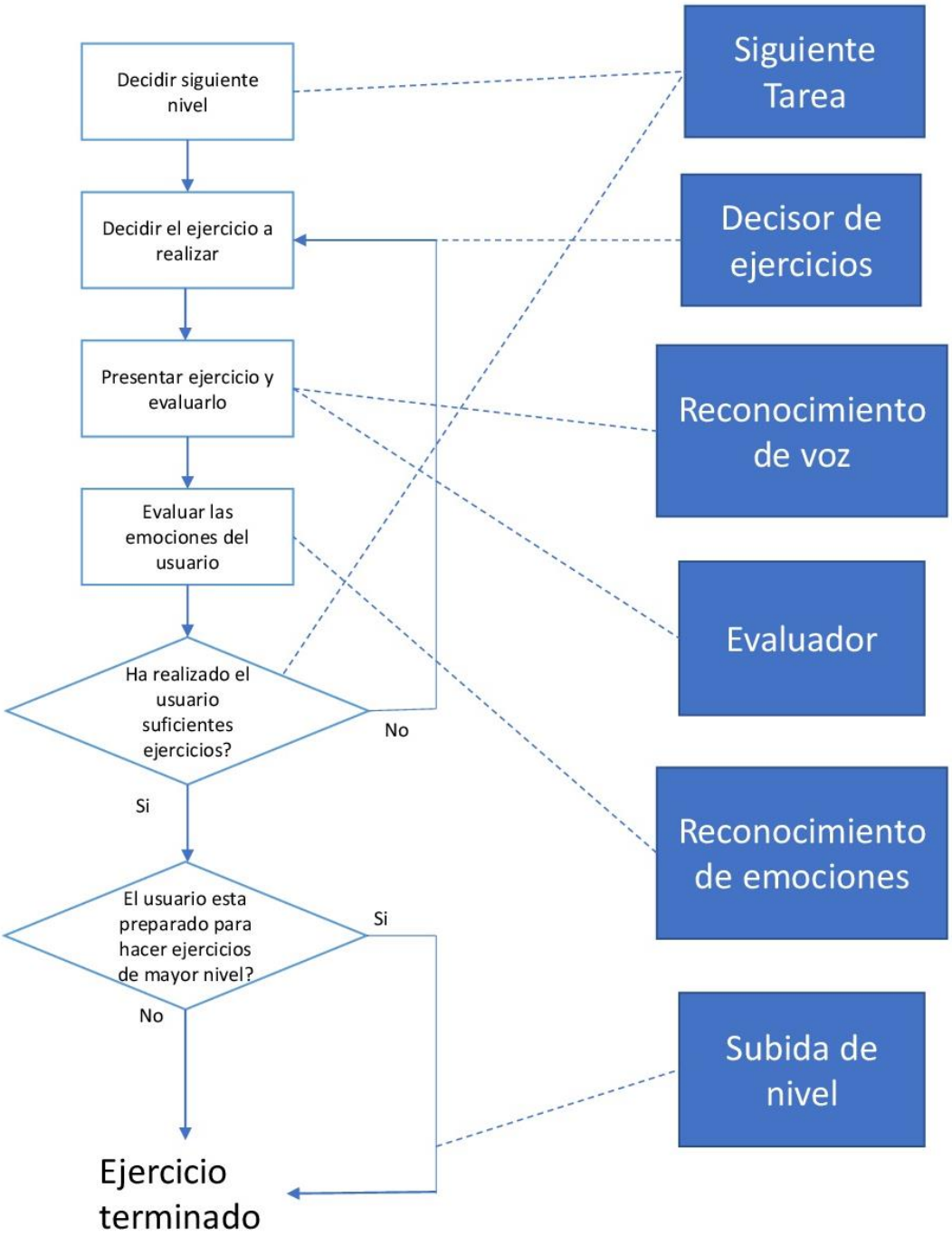


Figura 4: Diagrama de flujo del dialogo de ejercicios con los bloques que realizan cada funcionalidad

Capítulo 6. BASE DE DATOS

En esta sección se describen todos los aspectos de la base de datos usada en la aplicación. Primero se describe el diseño de la base de datos, indicando las tablas incluidas, sus relaciones y su función. Posteriormente se describe la funcionalidad necesaria de la base de datos y las tecnologías usadas para implementarla.

6.1 DESCRIPCIÓN DE LA BASE DE DATOS

Para la solución diseñada, se ha diseñado una base de datos que permite almacenar la información necesaria para su correcto funcionamiento. En la figura 5 se puede observar un esquema de las tablas y sus relaciones. Las tablas de la base de datos son las siguientes:

Tabla de usuarios: La tabla de usuarios permite la persistencia de la información relativa a los usuarios tanto dentro de esta tabla como en otras que hacen uso de la clave principal de usuario. Los campos de esta tabla son los siguientes:

- Id: Clave primaria.
- Nombre: Se usa para referirse al usuario de manera cercana. En este caso también se usa para autenticarse
- Apellidos: De igual funcionalidad que el nombre
- Progreso: Campo con contenido de numero entero para almacenar el nivel de lectura del usuario. Esto permite asignarle ejercicios de su nivel desde el primer ejercicio a realizar.
- Id de la cara: Aunque no se usa, este campo se ha creado para poder autenticarse en la aplicación mediante reconocimiento facial.

Tabla de ejercicios: En esta tabla se almacenan todos los ejercicios posibles a realizar por el usuario. Los campos de esta tabla son:

- Id

- Texto: Contiene el ejercicio en formato texto, el cual se entrega al usuario para que lo lea.
- Dificultad: A cada ejercicio se le asigna una dificultad en formato numérico que corresponde con el progreso del usuario

Tabla de progreso/dificultad: Esta tabla se usa para crear una clave para cada nivel de dificultad y asignarle una descripción textual. Campos:

- Id
- Texto: Contiene una palabra que describe el nivel de dificultad que simboliza el id

Tabla de registro de actividad: En esta tabla se almacenan registros sobre todos los ejercicios realizados por todos los usuarios. Aparte de ser útil para poder observar la progresión de los alumnos, es usado para decidir que ejercicio realizara el usuario y decidir si el usuario ha superado un nivel de dificultad. Los campos de esta tabla son:

- Id
- Id de usuario: Asocia el registro al usuario que ha realizado el ejercicio
- Id del ejercicio: Asocia el registro al ejercicio que ha realizado el usuario
- Evaluación: Contiene una asignación numérica que corresponde a la evaluación del ejercicio realizado por el usuario
- Fecha y hora: Permite observar la evolución temporal del usuario aparte de que ayuda en la decisión del ejercicio a realizar por el usuario

Existen múltiples relaciones en estas tablas que permiten el acceso ordenado a la información contenida en la base de datos.

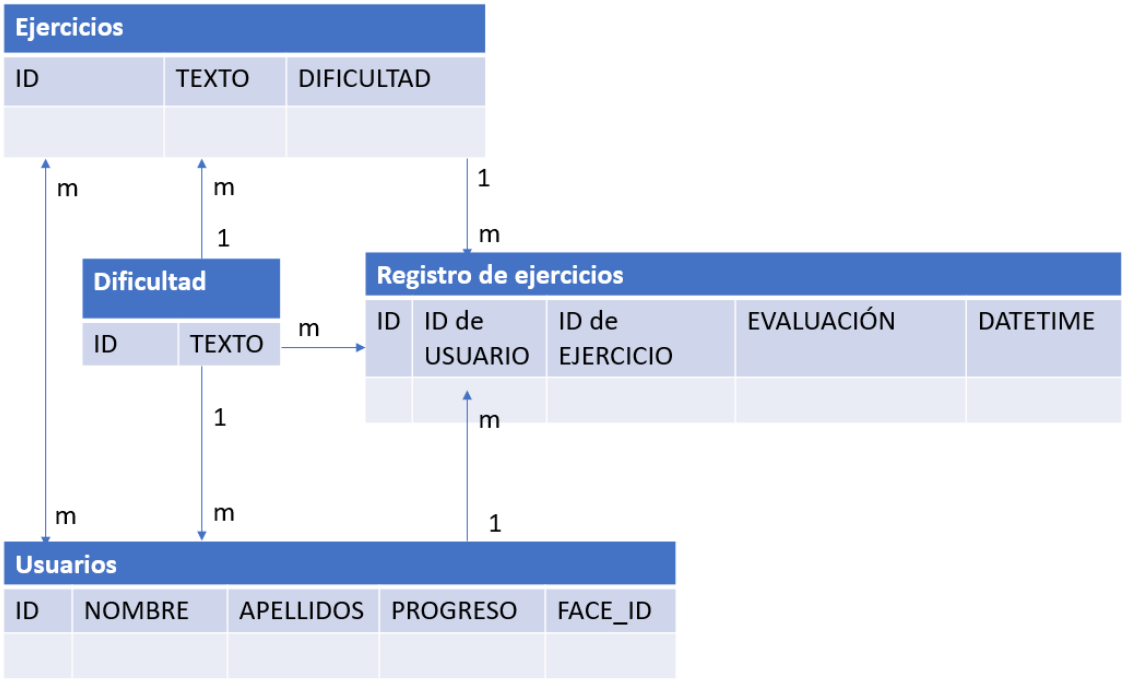


Figura 5: Esquema de las tablas de la BD y su relación

6.2 TECNOLOGÍAS E IMPLEMENTACIÓN DE LA BASE DE DATOS

Como se describió en el capitulo 4, la base de datos ha sido implementada en SQL. En referencia a la tecnología a usar para la implementación de la funcionalidad de acceso a la base de datos, se utiliza un entorno llamado Entity Framework. Este entorno permite la creación de modelos de las tablas de la base de datos. Estos modelos están introducidos en un contexto que se crea al iniciar la conexión con la base de datos. Esto nos permite realizar consultas sobre las clases del contexto en vez de realizar consultas SQL contra el servidor. Además, se utiliza una funcionalidad del entorno que permite realizar consultas conjuntas de dos tablas con relaciones superiores a 1-m, recuperando listas de la tabla asociada como campo de la consulta. En el anexo X se muestran los documentos de creación de la base de datos y el código de los modelos, consultas y contexto.

La funcionalidad implementada para el acceso a esta base de dato, clasificada por tablas es la siguiente:

Tabla de ejercicios:

- Recuperar lista de ejercicios de una dificultad con sus logs asociados: Esta consulta es una consulta compleja permitida gracias al uso de Entity Framework. Esto nos permite realizar una consulta sobre una tabla, y recuperar una lista de registros asociados a ese ejercicio como campo de esta. Esta funcionalidad es utilizada para elegir el ejercicio a entregar al usuario

Tabla de usuarios:

- Añadir usuario, necesario para añadir usuarios que usan por primera vez la aplicación
- Recuperar usuario por nombre y apellido. Se utiliza para autenticar a los usuarios que entrar en la aplicación
- Subir de dificultad al usuario: Cuando el bloque correspondiente lo cree conveniente, se invoca este método para guardar en la base de datos los cambios correspondientes.

Tabla de dificultades:

- Recuperar dificultad por id: Utilizado para mostrar información al usuario de manera más amigable

Tabla de Registros: No se necesita mas funcionalidad que la siguiente, dado que la consulta de esta tabla se realiza conjuntamente sobre la tabla de ejercicios

- Añadir registro: Para guardar la información de los ejercicios realizados

Capítulo 7. SERVICIOS COGNITIVOS

En esta sección se describen todos los servicios utilizados por la aplicación que consisten en redes neurales entrenadas por Microsoft.

7.1 SERVICIO DE RECONOCIMIENTO DE EMOCIONES

Face Api es un servicio en la nube de Microsoft contratado a través del portal de Azure. Este servicio permite obtener todo tipo de información relacionado con las caras que aparecen en una foto. En la figura 6 se puede observar un ejemplo de la información obtenida en texto tras enviar una fotografía al servicio.

La información necesaria que enviar al servicio es una fotografía con una o múltiples caras en ella. Para el caso de nuestra aplicación, la información solicitada al servicio son las emociones de las caras. Esta información se obtiene en forma de lista con emociones predeterminadas junto con coeficientes que indican el porcentaje de esa emoción representado en la cara. Estas emociones son: enfado, desprecio, asco, miedo, felicidad, desinterés, tristeza y sorpresa.

Adicionalmente, se utiliza una autenticación en el servicio a través de tokens, los cuales permiten el acceso al servicio durante un periodo de tiempo limitado, después del cual, es necesario renovar el token.

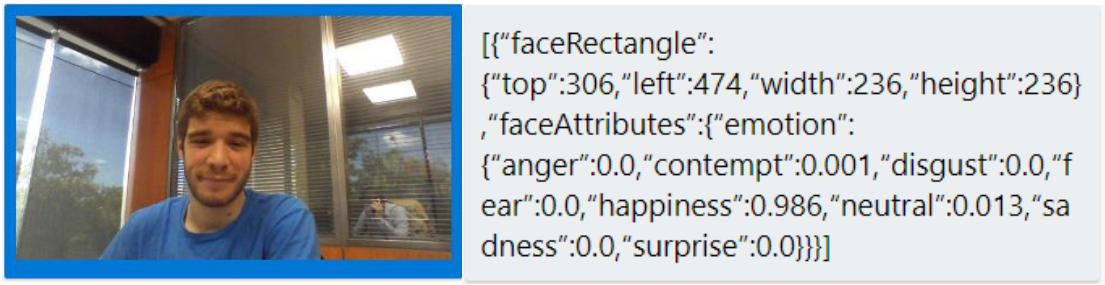


Figura 6: Ejemplo de Face API

7.2 SERVICIO DE RECONOCIMIENTO DE VOZ

Speech service, al igual que fase API, Api es un servicio en la nube de Microsoft contratado a través del portal de Azure. Este servicio tiene cuantiosa funcionalidad relacionada con reconocimiento por voz, conversión de voz a texto y viceversa, reconocimiento de intención y más. Para el caso de la solución, el servicio que se usa es el de conversión de voz a texto.

Como con Face API, previo a la solicitud de información, se solicita un token al servicio, con el cual se realizan las peticiones http al servicio.

La información que se le adjunta a la petición http para que el servicio funcione es un archivo audio en formato WAV. La respuesta a esta petición consiste en un texto que corresponde a lo que el servicio entiende que ha dicho el usuario en el archivo audio. Este texto es gramaticalmente correcto, es decir, consta solo de estructuras y palabras que existen en la lengua española

El hecho de que solo reconozca palabras de la lengua española representa un problema para esta aplicación. Esto es debido a que como la aplicación enseña a leer, los fallos son esperados y lo que se busca es la capacidad de identificarlos y así poder mostrárselos al usuario. Además, también resulta un problema porque en cuentos infantiles, generalmente se usan nombres o palabras que a veces no aparecen en el diccionario. Para solucionar este problema se recurre a un servicio adicional de Microsoft llamado Custom Language Model.

7.2.1 CUSTOM LANGUAGE MODEL

Este servicio es una plataforma montada sobre speech service que nos permite crear nuestros propios modelos de lenguaje. Esto significa que, adicionalmente a toda la lengua española, podemos añadirle al modelo de reconocimiento expresiones o palabras que esperamos que el usuario pueda utilizar, y el módulo reconocerá.

Para utilizar este servicio se han realizado los siguientes pasos:

1. Crear un documento de texto con las expresiones y palabras con errores. Después de seleccionar los ejercicios que se incluyen en la base de datos, se han recorrido para recoger palabras y expresiones que no son comunes en la lengua española para que la aplicación pueda reconocerlos. Estas expresiones se han introducido línea por línea en un archivo de texto. Adicionalmente se han extraído las palabras complicadas de leer de los ejercicios y se han introducido en el mismo archivo con errores (por ejemplo, bellena en vez de ballena).
2. Usando este archivo de texto, se ha creado un modelo de lenguaje customizado en el portal cris.ai.
3. Con el modelo creado se crea un nuevo endpoint para el uso de este servicio con un nuevo modelo de lenguaje.

En la figura 7 se puede observar como el usuario comete un error en la lectura y es captado por el servicio de traducción de voz a texto

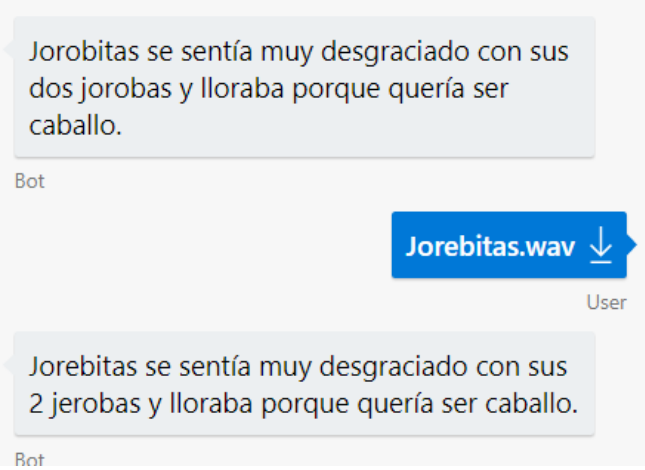


Figura 7: Ejemplo de traducción a texto con errores

Capítulo 8. SERVICIOS INTERNOS

Este capítulo describe todos los módulos internos de la solución que hacen posible esta. En la figura 1 se mostraban estos y en la figura 4 se puede observar la función que desempeña cada uno de ellos en el diagrama de flujo del dialogo principal.

8.1 MÓDULO DE SIGUIENTE TAREA

Este módulo cosiste en un conjunto de reglas que permiten tomar la decisión de que hacer después de terminar un ejercicio de lectura.

Las decisiones tomadas por este módulo tienen en consideración las emociones del usuario, en concreto neutralidad, tristeza, disgusto y felicidad, la evaluación del ejercicio anterior, el nivel del ejercicio anterior y el numero de ejercicios de lectura que ha realizado el usuario. En función de todas estas características, el módulo puede tomar 4 posibles decisiones:

1. Hacer un ejercicio del mismo nivel al anterior.
2. Hacer un ejercicio de un nivel menos al anterior.
3. Hacer un ejercicio de un nivel más al anterior.
4. Acabar la actividad de lectura.

Estas decisiones se toman siguiendo un conjunto de reglas que se pueden observar en el esquema de la figura 8.

Si se da el caso de que dos o mas de las reglas son verdad a la vez, entonces el orden de importancia de las reglas es:

1. Regla 1
2. Regla 2
3. Regla 4
4. Regla 3

Regla 1	# Ejercicios	3 +	4 +	5 +	Parar
	Neutralidad	0.85 +	0.75 +	0.65 +	
Regla 2	Tristeza	0.25 +			Bajar nivel
Regla 3	# Ejercicios	2 +			Subir nivel
	Felicidad o disgusto	0.70 +			
Regla 4	Evaluación	0.5 -			Bajar nivel

Tabla 1: Tabla de reglas del módulo de siguiente tarea

8.2 MÓDULO DE DECISIÓN DE EJERCICIO

Este módulo se invoca si, una vez llamado al módulo de siguiente tarea, se decide realizar un nuevo ejercicio. Se encarga de decidir el nuevo ejercicio a realizar.

La información que utiliza para realizar esta función es el nivel del ejercicio a realizar y el usuario que está realizando este ejercicio. Esta información se utiliza para procesar una lista de ejercicios que contienen los registros del usuario. Finalmente se obtiene un ejercicio que se presenta al usuario.

La figura 9 es un diagrama de flujo que representa todas las acciones que toma el módulo hasta obtener el ejercicio

Como se puede observar en el diagrama, este módulo favorece en primer lugar los ejercicios que el usuario no ha realizado nunca, consiguiendo así una práctica completa. Y en segundo lugar, presenta los ejercicios que más tiempo lleva sin realizar el usuario, lo que consigue no aburrirle.

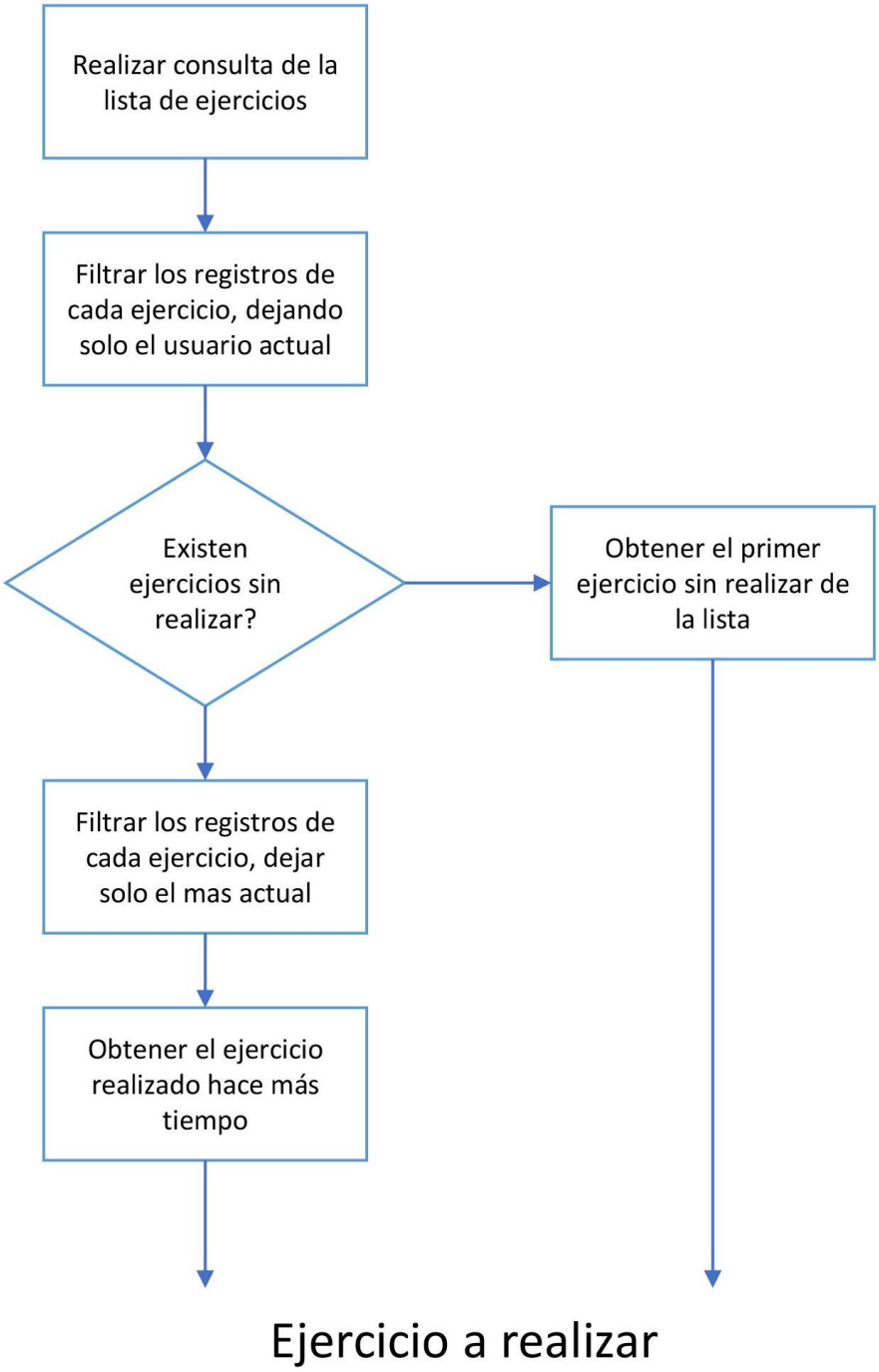


Figura 8. Diagrama de flujo del módulo de decisión de ejercicio

8.3 MODULO EVALUADOR

Este módulo se encarga de evaluar la realización del ejercicio por el usuario. Esto permite adecuar los ejercicios siguientes a las habilidades de ese día del usuario además de guardar un registro temporal de la realización de los ejercicios y poder mostrar al usuario sus aptitudes.

Para este objetivo, el módulo necesitado utiliza el texto original del ejercicio y el traducido por el modulo de reconocimiento por voz

Con esta información, el módulo realiza una comparación palabra a palabra de los dos textos. El sistema de evaluación cuenta cada fallo que se ha realizado y por cada uno, se resta 15 a la puntuación total sobre 100 hasta un mínimo de 10.

Como pueden existir desfases en los textos debido a fallos del módulo de reconocimiento por voz o por fallos del usuario (podría leer “de el” en vez de “del” por ejemplo), es necesario evaluar si existen desfases a medida que se van comparando las palabras de los dos textos. Para esto, como los textos son cortos, se ha decidido que solo se evaluarán los desfases en un sentido, es decir, mas palabras en el original que en el del usuario o viceversa.

Además, el modulo de conversión de voz a texto devuelve los números en formato numérico, y por lo tanto es necesario pasar estos números a letra antes de pasar el comparador.

8.4 MÓDULO DE SIGUIENTE NIVEL

Este módulo es invocado cuando el módulo de siguiente tarea decide terminar la actividad. Cuando esto sucede, el módulo de siguiente nivel decide si el usuario ha obtenido las suficientes habilidades lectoras como para realizar ejercicios del siguiente nivel consistentemente.

Para tomar esta decisión, el módulo utiliza el último registro que tiene el usuario sobre cada uno de los ejercicios. El módulo comprueba todos los registros buscando alguno que tenga alguna evaluación por debajo de 85, es decir, 2 fallos o más. Si existe algún registro que cumpla esta característica, se decide no subir al usuario de nivel. Si, por lo contrario, no existe ningún registro que cumpla esta condición, se realiza una consulta en la base de datos de usuarios en la que se modifica el nivel del usuario a uno superior.

Capítulo 9. RESULTADOS Y FUTURAS AMPLIACIONES

Es imposible conseguir resultados sobre la mejoría de las habilidades lecturas de un usuario durante un periodo de tiempo corto. Para la correcta recolección de información sobre el funcionamiento de la aplicación, en mi opinión, habría que seguir dos caminos:

1. Poner la aplicación en la nube y crear una interfaz gráfica amigable. Esto permitiría monitorizar a los usuarios y su aprendizaje durante un periodo de tiempo de alrededor de un año. Tras este periodo se crearían unos documentos de análisis con los datos de los registros que analizan el avance de nivel de los usuarios entre otros datos
2. Ofrecer la aplicación a colegios. Esto permitiría usar la aplicación como deberes de lectura, lo cual garantizaría un gran número de datos para analizar. Pero también permitiría el análisis de la evolución de los usuarios por parte de los docentes, para conocer si la aplicación les enseña o por lo contrario solo aplican los conocimientos en ella.

A pesar de esto, si se ha podido observar como los usuarios son capaces de mantener una conversación guiada con el bot. Es necesario que sea guiada por la falta de interfaz gráfica. Durante estas pruebas también se ha podido observar como el bot toma decisiones con toda la información que obtiene a lo largo de la actividad. Estas decisiones parecen ser las adecuadas para las situaciones en las que se toman.

Además, se ha podido observar como los usuarios se divierten mientras que usan la aplicación, lo cual cumple uno de los objetivos principales del proyecto.

Sobre las futuras ampliaciones, considero necesaria una interfaz gráfica visual para que el proyecto pueda ser lanzado al mercado. El estado actual del proyecto implica que es necesario un guía para ser capaz de conversar con el bot y enviar los archivos de audio. Gracias a esta implementación, el usuario no necesitaría la ayuda de nadie para mantener la

conversación, dado que, en primer lugar, el output de bot sería en texto y en segundo lugar, en vez de escribir, el bot estaría a la escucha del usuario.

Además de esta ampliación, existen dos más que, aunque no necesarias, creo que serían interesantes para seguir investigando los beneficios de la inteligencia artificial sobre la enseñanza y el aprendizaje.

Primero, introducir ejercicios de otros ámbitos de la enseñanza, para poder monitorizar que ámbitos son los más beneficiados por esta nueva tecnología

Segundo, crear un módulo de machine Learning entrenado para tomar las decisiones. Actualmente, las decisiones se toman en base a reglas que no se adaptan a los usuarios. Mediante la creación de un modelo creado para cada usuario y entrenado por sus acciones para conseguir la optima forma de educarle, la aplicación tendría formas de actuación completamente distintas para cada niño y permitirá analizar sus capacidades en distintos ámbitos.

Capítulo 10. CONCLUSIONES

Se puede concluir que la solución desarrollada cumple los objetivos que se le habían impuesto como necesarios y por lo tanto se observa que la IA está suficientemente avanzada como para aparecer en la enseñanza y contribuir a esta de manera significativa. En este caso se ha implementado un sistema maestro, el cual puede llegar a ser una herramienta que no solo enseñe a leer, sino que se puede aplicar a otras disciplinas como las matemáticas, la historia o la programación. Aparte de esto, existen una gran variedad de formas de aprender como el descubrimiento guiado, los cuales gracias a la IA podrán ser posibles sin la necesidad de una persona para enseñar.

Capítulo 11. BIBLIOGRAFÍA

[1] Bendezu, K.. "DIAGRAMA UML Y ARQUITECTURA DEL SISTEMA". Sistemas Distribuidos 2013. Febrero, 2013. http://comparape.blogspot.com.es/2013/02/diagrama-uml-y-arquitectura-del-sistema.html.

[2] Herrero Alcántara, T. "Big Data: ¿Moda u oportunidad de negocio para el emprendedor?", Think Big, Octubre 2014. http://blogthinkbig.com/big-data-emprendedor/.

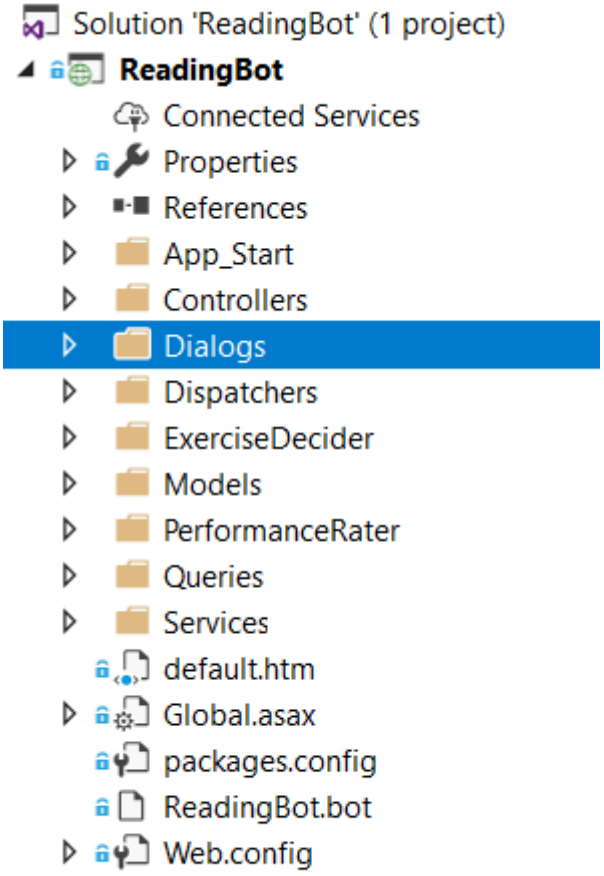
[3] Loeffler, B. "Cloud Computing: What is Infrastructure as a Service", Microsoft Technet Magazine, October 211. https://technet.microsoft.com/en-us/magazine/hh509051.aspx

[4] Vlassis, N.A.; Papakonstantinou, G.; Tsanakas, P. *Dynamic sensory probabilistic maps for mobile robot localization.* Source: Proceedings. 1998 IEEE / RSJ International Conference on Intelligent Robots and Systems. Innovations in Theory, Practice and Applications (Cat. No.98CH36190) New York, NY, USA: IEEE, 1998.p.718-23 vol.2 of 3 vol. xliv+2010 pp. 11.

ANEXO A

Este anexo contiene inserciones de código de todas las clases creadas para la solución informática.

Primero aparece una captura de la estructura del proyecto y luego las distintas clases



App start:

```
using Newtonsoft.Json;
using Newtonsoft.Json.Serialization;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web.Http;

namespace ReadingBot
{
    public static class WebApiConfig
    {
        public static void Register(HttpConfiguration config)
        {
            // Json settings
            config.Formatters.JsonFormatter.SerializerSettings.NullValueHandling =
NullValueHandling.Ignore;
            config.Formatters.JsonFormatter.SerializerSettings.ContractResolver = new
CamelCasePropertyNamesContractResolver();
            config.Formatters.JsonFormatter.SerializerSettings.Formatting =
Formatting.Indented;
            JsonConvert.DefaultSettings = () => new JsonSerializerSettings()
            {
                ContractResolver = new CamelCasePropertyNamesContractResolver(),
                Formatting = Newtonsoft.Json.Formatting.Indented,
                NullValueHandling = NullValueHandling.Ignore,
            };

            // Web API configuration and services

            // Web API routes
            config.MapHttpAttributeRoutes();

            config.Routes.MapHttpRoute(
                name: "DefaultApi",
                routeTemplate: "api/{controller}/{id}",
                defaults: new { id = RouteParameter.Optional }
            );
        }
    }
}
```

Controllers:

```
using System.Net;
using System.Net.Http;
using System.Threading.Tasks;
using System.Web.Http;
using Microsoft.Bot.Builder.Dialogs;
using Microsoft.Bot.Connector;

namespace ReadingBot
{
    [BotAuthentication]
    public class MessagesController : ApiController
    {
        /// <summary>
        /// POST: api/Messages
        /// Receive a message from a user and reply to it
        /// </summary>
        public async Task<HttpResponseMessage> Post([FromBody]Activity activity)
        {
            if (activity.GetActivityType() == ActivityTypes.Message)
            {
                await Conversation.SendAsync(activity, () => new Dialogs.RootDialog());
            }
            else
            {
                HandleSystemMessage(activity);
            }
            var response = Request.CreateResponse(HttpStatusCode.OK);
            return response;
        }

        private Activity HandleSystemMessage(Activity message)
        {
            string messageType = message.GetActivityType();
            if (messageType == ActivityTypes.DeleteUserData)
            {
                // Implement user deletion here
                // If we handle user deletion, return a real message
            }
            else if (messageType == ActivityTypes.ConversationUpdate)
            {
                // Handle conversation state changes, like members being added and
removed
                // Use Activity.MembersAdded and Activity.MembersRemoved and
Activity.Action for info
                // Not available in all channels
            }
            else if (messageType == ActivityTypes.ContactRelationUpdate)
            {
                // Handle add/remove from contact lists
                // Activity.From + Activity.Action represent what happened
            }
```

```
            else if (messageType == ActivityTypes.Typing)
            {
                // Handle knowing that the user is typing
            }
            else if (messageType == ActivityTypes.Ping)
            {
            }

            return null;
        }
    }
}
```

Dialogs:

```
using System;
using System.Collections.Generic;
using System.Threading;
using System.Threading.Tasks;
using Microsoft.Bot.Builder.Dialogs;
using Microsoft.Bot.Connector;
using ReadingBot.Dispatchers;
using ReadingBot.Models;
using ReadingBot.Queries;
namespace ReadingBot.Dialogs
{
    [Serializable]
    public class IdentificationDialog : IDialog<object>
    {
        String name;
        String surname;
        User user;
        public async Task StartAsync(IDialogContext context) =>
context.Wait(StartDialog);
        private async Task StartDialog(IDialogContext context,
IAwaitable<IMessageActivity> result)
        {

            await context.PostAsync("Dime tu nombre para saber quien eres");
            context.Wait(this.getName);
        }
        private async Task getName(IDialogContext context, IAwaitable<IMessageActivity>
result)
        {
            var message = await result as Activity;
            this.name = message.Text;

            await context.PostAsync($"Ahora dime tu apellido");
            context.Wait(this.getSurname);
        }
        private async Task getSurname(IDialogContext context,
IAwaitable<IMessageActivity> result)
        {
            var activity = await result as Activity;
            this.surname = activity.Text;
            await context.PostAsync($"Veo que te llamas {this.name} {this.surname}");
            await context.PostAsync("¿He cogido tu nombre y apellidos bien?");
            context.Wait(this.checkFullName);
        }
        private async Task checkFullName(IDialogContext context,
IAwaitable<IMessageActivity> result)
        {
            var activity = await result as Activity;
            if (activity.Text.ToLower().Contains("si"))
            {
                await context.PostAsync("Perfecto!");
                this.user = UserQuery.GetUserByName(new User(this.name, this.surname));
```

```
                if (user != null)
                {
                    await context.PostAsync($"Acabo de encontrarte en mi base de datos,
tu nivel es {user.Progress}");
                    context.Done<User>(this.user);
                }
                else
                {
                    await context.PostAsync("Parece que no estas registrado, ¿es esto
verdad?");
                    context.Wait(this.checkNewUser);
                }
            }
            else
            {
                await context.PostAsync("Oh vaya, entonces primero dime tu nombre otra
vez");
                context.Wait(this.getName);
            }
        }
        private async Task checkNewUser(IDialogContext context,
IAwaitable<IMessageActivity> result)
        {
            var activity = await result as Activity;
            if (activity.Text.ToLower().Contains("si"))
            {
                try
                {
                    await context.PostAsync("Vale, entonces te añado a la base de
datos");
                    UserQuery.AddUser(new User(this.name,this.surname));
                    await context.PostAsync("Has sido añadido a la base de datos :)");
                    this.user = UserQuery.GetUserByName(new User(this.name,
this.surname));
                }catch(Exception e)
                {
                    await context.PostAsync(e.StackTrace);
                    await context.PostAsync(e.Message);
                }
                context.Done(this.user);
            }
            else
            {
                await context.PostAsync("Vale, vamos a intentar identificarte otra vez.
Dime tu nombre");
                context.Wait(this.getSurname);
            }
        }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Threading;
using System.Threading.Tasks;
using Microsoft.Bot.Builder.Dialogs;
using Microsoft.Bot.Connector;
using ReadingBot.Dispatchers;
using ReadingBot.Models;
using ReadingBot.Queries;
using ReadingBot.ExerciseDecider;
using System.Linq;
using System.Net.Http.Headers;
using System.Net.Http;
using System.IO;
using ReadingBot.Services;

namespace ReadingBot.Dialogs
{
    [Serializable]
    public class RootDialog : IDialog<object>
    {
        User user;
        int level;
        int number;
        Exercis exercise;
        Performance performance;
        Emotion emotion;
        public async Task StartAsync(IDialogContext context)
        {
            // Root dialog initiates and waits for the next message from the user.
            // When a message arrives, call MessageReceivedAsync.
            await this.startIdentification(context);
        }
        private async Task startIdentification(IDialogContext context)
        {
            await context.PostAsync("Hola! Soy un bot para mejorar tu lectura. Ahora
procedere a autentificarte");
            context.Call(new IdentificationDialog(),
this.ResumeAfterIdentificationDialog);
        }
        private async Task ResumeAfterIdentificationDialog(IDialogContext context,
IAwaitable<object> result)
        {
            // Store the value that NewOrderDialog returned.
            // (At this point, new order dialog has finished and returned some value to
use within the root dialog.)
            var user = await result as User;
            await context.PostAsync($"Veo que tu nombre es {user.Name} {user.Surname} y
tu nivel actual es {user.Progress}");
            this.user = user;
            this.level = user.Progress;
            this.number = 0;
            // Again, wait for the next message from the user.
            await context.PostAsync("¿Estas preparado para hacer tu primer ejercicio?");
            context.Wait(this.FirstExercise);
        }
```

```
        private async Task FirstExercise(IDialogContext context, IAwaitable<object>
result)
        {

            var activity = await result as Activity;
            await context.PostAsync("Hagamos el primer ejercicio!");
            await context.PostAsync("Lee el siguiente texto");
            try
            {
                this.exercise =
ExerciseDecider.ExerciseDecider.DecideExerciseForUser(ExerciseQuery.GetExercisOfLevelWith
Logs(this.level), this.user);

            }
            catch(Exception e)
            {
                await context.PostAsync(e.StackTrace);
                await context.PostAsync(e.Message);
                await context.PostAsync(e.HelpLink);
            }
            await context.PostAsync(exercise.Text);

            context.Wait(this.ExerciseResponse);
        }


        private async Task ExerciseResponse(IDialogContext context, IAwaitable<object>
result)
        {
            var activity = await result as Activity;
            if (activity.Attachments.Count > 0 &&
activity.Attachments.First().ContentType.Equals("audio/wav"))
            {
                MicrosoftCognitiveSpeechService speechService = new
MicrosoftCognitiveSpeechService();
                string response = await speechService.ReturnTextFromAudio(activity);
                await context.PostAsync(response);
                this.performance =
PerformanceRater.PerformanceRater.RatePerformance(this.exercise.Text, response);
                this.number = this.number + 1;
                await context.PostAsync($"Tu calificacion ha sido {performance.rating}
sobre 100");
                foreach (WordError error in this.performance.errors)
                {
                    await context.PostAsync($"Has pronunciado mal la palabra
{error.originalWord}, has dicho {error.errorWord}");
                }
                LogQuery.AddNewLog(new Logs_Excercises(this.user.UserID,
this.exercise.ExerciseID, this.performance.rating));
                await context.PostAsync("Ahora mandame un foto para ver como te
sientes");
                context.Wait(this.GetPhoto);
            }
            else
            {
```

```
                await context.PostAsync("No he recivido ningun audio, asegurate de que
sea en formato WAV");
                context.Wait(this.ExerciseResponse);
            }
        }
        private async Task GetPhoto(IDialogContext context, IAwaitable<object> result)
        {
            var activity = await result as Activity;
            if (activity.Attachments.Count > 0 &&
activity.Attachments[0].ContentType.Equals("image/jpeg"))
            {
                this.emotion = await MicrosoftFaceApiService.getEmotions(activity);
                await context.PostAsync(emotion.toString());


                string nextTask = NextTaskDispatcher.NextTask(this.emotion,
this.performance.rating, this.level, this.number);
                if (nextTask == "stop")
                {
                    await context.PostAsync("Muy bien trabajado!! Ahora te toca
descansar:)");

if(ExerciseDecider.ExerciseDecider.LevelUpDecider(ExerciseQuery.GetExercisOfLevelWithLogs
(this.user.Progress), this.user))
                    {
                        await context.PostAsync("Felicidades, has subido de nivel :)");
                        UserQuery.LevelUpUser(this.user);
                    }
                    await context.PostAsync("Vuelve pronto!");
                    context.Done(this);
                }
                else
                {
                    this.level = Int32.Parse(nextTask);
                    await context.PostAsync("Estas preparado para hacer tu siguiente
ejercicio?");
                    context.Wait(this.IntroduceExercise);
                }
            }
            else
            {
                await context.PostAsync("No he recibido ninguna foto. Prueba a volver a
mandarmela");
                context.Wait(this.GetPhoto);
            }
        }
        private async Task dispatchNextExercise(IDialogContext context,
IAwaitable<object> result)
        {
            var activity = await result as Activity;

            string nextTask = NextTaskDispatcher.NextTask(this.emotion,
this.performance.rating, this.level, this.number);
            if(nextTask == "stop")
            {
```

```
                await context.PostAsync("Muy bien trabajado!! Ahora te toca
descansar:)");
                await context.PostAsync("Vuelve pronto!");
                context.Done(this);
            }
            else
            {
                this.level = Int32.Parse(nextTask);
                await context.PostAsync("Sigamos con el siguiente ejercicio");
                context.Wait(this.IntroduceExercise);
            }
        }
        private async Task IntroduceExercise(IDialogContext context, IAwaitable<object>
result)
        {
            var activity = await result as Activity;

            await context.PostAsync("Porfavor lee el siguiente texto");
            try
            {
                this.exercise =
ExerciseDecider.ExerciseDecider.DecideExerciseForUser(ExerciseQuery.GetExercisOfLevelWith
Logs(this.level), this.user);
                await context.PostAsync(exercise.Text);
            }catch(Exception e)
            {
                await context.PostAsync(e.StackTrace);
                await context.PostAsync(e.Message);
                await context.PostAsync(e.HelpLink);
            }
            context.Wait(this.ExerciseResponse);
        }



    }
}
```

Dispatchers:

```
using ReadingBot.Services;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace ReadingBot.Dispatchers
{
    public class NextTaskDispatcher
    {
        public static string NextTask(Emotion emotion,int performance, int level, int
number)
        {
            int stop = 0;
            int levelup = 0;
            int leveldown = 0;
            double sadness = emotion.sadness;
            double neutral = emotion.neutral;
            double happiness = emotion.happyness;
            double contempt = emotion.contempt;
            if((number > 6)||(number >2 && neutral >0.84)||(number >3 && neutral
>0.74)||(number > 4 && neutral > 0.64)||(number >5 && neutral > 0.54))
            {
                stop = 1;
            }
            if(sadness > 0.24 && level > 1)
            {
                leveldown = 1;
            }
            if((happiness > 0.70||contempt>0.90)  && number > 2)
            {
                levelup = 1;
            }
            if(performance < 50 && level > 1)
            {
                leveldown = 1;
            }

            if(stop == 1)
            {
                return "stop";
            }
            else if(leveldown == 1)
            {
                level = level - 1;
                return level.ToString();
            }
            else if(levelup == 1)
            {
                level = level + 1;
                return level.ToString();
            }
            else
            {
```

```
            {
                return level.ToString();
            }
        }
    }
}
```

Exercise Decider:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using ReadingBot.Models;
namespace ReadingBot.ExerciseDecider
{
    public class ExerciseDecider
    {
        private static List<Exercis> FilterUsersFromExercises(List<Exercis> exercises,
User user)
        {
            foreach (Exercis exercise in exercises.ToList())
            {
                foreach(Logs_Excercises log in exercise.Logs_Excercises.ToList())
                {
                    if(log.UserID != user.UserID)
                    {
                        exercise.Logs_Excercises.Remove(log);
                    }
                }
            }

            return exercises;

        }
        private static Exercis FindExerciseWithoutLog(List<Exercis> exercises)
        {
            foreach(Exercis exercise in exercises)
            {
                if (exercise.Logs_Excercises.Count == 0)
                {
                    return exercise;
                }

            }
            return null;
        }
        private static List<Exercis> FilterLogsLatestInclusion(List<Exercis> exercises)
        {
            foreach(Exercis exercise in exercises.ToList())
            {
                Logs_Excercises toreturnlog = exercise.Logs_Excercises.First();
                foreach(Logs_Excercises log in exercise.Logs_Excercises.ToList())
                {
                    if(DateTime.Compare(toreturnlog.Timestamp, log.Timestamp) < 0)
                    {
                        toreturnlog = log;
                    }
                }
```

```
                foreach (Logs_Excercises log in exercise.Logs_Excercises.ToList())
                {
                    if (log != toreturnlog)
                    {
                        exercise.Logs_Excercises.Remove(log);
                    }
                }
            }
            return exercises;
        }
        private static Exercis FindExerciseWithLatestLog(List<Exercis> exercises)
        {
            Exercis toreturn = exercises.First();
            foreach(Exercis exercise in exercises)
            {
                if (DateTime.Compare(toreturn.Logs_Excercises.First().Timestamp,
exercise.Logs_Excercises.First().Timestamp) > 0)
                {
                    toreturn = exercise;
                }
            }
            return toreturn;
        }
        public static Exercis DecideExerciseForUser(List<Exercis> exercises,User user)
        {
            List<Exercis> filteredbyuser =
ExerciseDecider.FilterUsersFromExercises(exercises, user);
            Exercis toreturn = ExerciseDecider.FindExerciseWithoutLog(filteredbyuser);
            if(toreturn == null)
            {
                List<Exercis> latestlogexercises =
ExerciseDecider.FilterLogsLatestInclusion(filteredbyuser);
                toreturn = ExerciseDecider.FindExerciseWithLatestLog(latestlogexercises);
            }
            return toreturn;
        }
        public static bool LevelUpDecider(List<Exercis> exercises,User user)
        {
            List<Exercis> filteredbyuser =
ExerciseDecider.FilterUsersFromExercises(exercises, user);
            Exercis toreturn = ExerciseDecider.FindExerciseWithoutLog(filteredbyuser);
            if (toreturn == null)
            {
                List<Exercis> latestlogexercises =
ExerciseDecider.FilterLogsLatestInclusion(filteredbyuser);
                foreach(Exercis exercise in latestlogexercises)
                {
                    if (exercise.Logs_Excercises.First().Performance < 85)
                        return false;
                }
                return true;
            }
            else
                return false;
        }
    }
}
```

Models:

```
namespace ReadingBot.Models
{
    using System;
    using System.Collections.Generic;
    [Serializable]
    public partial class Dificulty
    {
        [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage",
"CA2214:DoNotCallOverridableMethodsInConstructors")]
        public Dificulty()
        {
            this.Exercises = new HashSet<Exercis>();
            this.Users = new HashSet<User>();
        }
        public Dificulty(int DificultyID)
        {
            this.DificultyID = DificultyID;
        }
        public Dificulty(int DificultyID,string text)
        {
            this.DificultyID = DificultyID;
            this.Text = text;
        }


        public int DificultyID { get; set; }
        public string Text { get; set; }

        [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage",
"CA2227:CollectionPropertiesShouldBeReadOnly")]
        public virtual ICollection<Exercis> Exercises { get; set; }
        [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage",
"CA2227:CollectionPropertiesShouldBeReadOnly")]
        public virtual ICollection<User> Users { get; set; }
    }
}
```

```
namespace ReadingBot.Models
{
    using System;
    using System.Collections.Generic;
    [Serializable]
    public partial class Exercis
    {
        [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage",
"CA2214:DoNotCallOverridableMethodsInConstructors")]
        public Exercis()
        {
            this.Logs_Excercises = new HashSet<Logs_Excercises>();
        }

        public int ExerciseID { get; set; }
        public string Text { get; set; }
        public int Diculty { get; set; }

        public virtual Dificulty Dificulty { get; set; }
        [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage",
"CA2227:CollectionPropertiesShouldBeReadOnly")]
        public virtual ICollection<Logs_Excercises> Logs_Excercises { get; set; }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace ReadingBot.Models
{
    [Serializable]
    public class Performance
    {
        public int rating;
        public List<WordError> errors;
        public Performance(int rating, List<WordError> errors)
        {
            this.rating = rating;
            this.errors = errors;
        }
    }
    [Serializable]
    public class WordError
    {
        public WordError(string originalWord, string errorWord)
        {
            this.originalWord = originalWord;
            this.errorWord = errorWord;
        }
        public string originalWord;
        public string errorWord;
    }
}
```

```
namespace ReadingBot.Models
{
    using System;
    using System.Collections.Generic;
    [Serializable]
    public partial class User
    {
        [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage",
"CA2214:DoNotCallOverridableMethodsInConstructors")]
        public User()
        {
            this.Logs_Excercises = new HashSet<Logs_Excercises>();
        }
        public User(int userID)
        {
            this.UserID = userID;
        }
        public User(string Name,string Surname)
        {
            this.Name = Name;
            this.Surname = Surname;
        }

        public int UserID { get; set; }
        public string Name { get; set; }
        public string Surname { get; set; }
        public int Progress { get; set; }
        public string FaceID { get; set; }

        public virtual Dificulty Dificulty { get; set; }
        [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage",
"CA2227:CollectionPropertiesShouldBeReadOnly")]
        public virtual ICollection<Logs_Excercises> Logs_Excercises { get; set; }
    }
}
```

```
namespace ReadingBot.Models
{
    using System;
    using System.Collections.Generic;
    [Serializable]
    public partial class Logs_Excercises
    {
        public Logs_Excercises()
        {

        }
        public Logs_Excercises(int UserID,int ExercisesID,int performance)
        {
            this.UserID = UserID;
            this.ExercisesID = ExercisesID;
            this.Performance = performance;
            this.Timestamp = System.DateTime.Now;
        }
        public int LogID { get; set; }
        public int UserID { get; set; }
        public int ExercisesID { get; set; }
        public int Performance { get; set; }
        public System.DateTime Timestamp { get; set; }

        public virtual Exercis Exercis { get; set; }
        public virtual User User { get; set; }
    }
}
```

Performance rater:

```
using ReadingBot.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text.RegularExpressions;
using System.Web;

namespace ReadingBot.PerformanceRater
{
    public class PerformanceRater
    {
        public static Performance RatePerformance(string original, string exercise)
        {
            int rate = 100;
            int desfase = 0;
            string exerciseText = exercise.Trim();
            List<WordError> errors = new List<WordError>();
            string[] originalWords = original.Split(' ');
            string[] exerciseWords = exercise.Split(' ');
            for (int i = 0;i < exerciseWords.Length;i++)
            {

                if (Char.IsDigit(exerciseWords[i][0]))
                {
                    exerciseWords[i] =
Services.NumeroALetra.enletras(exerciseWords[i]).ToLower();

                }
                if (i != 0)
                    exerciseWords[i] = exerciseWords[i].ToLower();
            }
            for (int i = 0; i < originalWords.Length; i++)
            {
                if(i != originalWords.Length-1)
                    originalWords[i] = originalWords[i].Trim(',', '.');
                if (i != 0)
                    originalWords[i] = originalWords[i].ToLower();
            }
            if (originalWords.Length < exerciseWords.Length)
            {
                for (int i = 0; i < originalWords.Length; i++)
                {
                    if (originalWords[i] != exerciseWords[i+desfase])
                    {
                        if (i + 1 < originalWords.Length)
                        {
```

```
                            if(originalWords[i+1] == exerciseWords[i+ desfase + 2])
                            {
                                errors.Add(new WordError(originalWords[i],
String.Concat(exerciseWords[i + desfase], " ", exerciseWords[i + desfase + 1])));
                                desfase = desfase + 1;
                            }
                            else
                                errors.Add(new WordError(originalWords[i],
exerciseWords[i + desfase]));

                        }else
                            errors.Add(new WordError(originalWords[i], exerciseWords[i +
desfase]));
                        if (rate > 15)
                            rate = rate - 15;

                    }

                }
            }
            else if (originalWords.Length > exerciseWords.Length)
            {
                for (int i = 0; i < exerciseWords.Length; i++)
                {
                    if (originalWords[i + desfase] != exerciseWords[i])
                    {
                        if (i + 1 < exerciseWords.Length)
                        {
                            if (exerciseWords[i + 1] ==originalWords[i + desfase + 2])
                            {
                                errors.Add(new WordError(String.Concat(originalWords[i +
desfase], " ", originalWords[i + desfase + 1]), exerciseWords[i]));
                                desfase = desfase + 1;
                            }
                            else
                                errors.Add(new WordError(originalWords[i+ desfase],
exerciseWords[i]));

                        }
                        else
                            errors.Add(new WordError(originalWords[i + desfase],
exerciseWords[i]));
                        if (rate > 15)
                            rate = rate - 15;

                    }

                }
            }
            else
            {
                for (int i = 0; i < exerciseWords.Length; i++)
                {
```

```
                    if (originalWords[i] != exerciseWords[i])
                    {
                        errors.Add(new WordError(originalWords[i], exerciseWords[i]));
                        if (rate > 15)
                            rate = rate - 15;
                    }

                }
            }
            Performance performance = new Performance(rate, errors);
            return performance;
        }
        public static int RatePerformanceWithLength(string original, string exercise, int
exerciseDuration)
        {
            int rate = 100;

            string exerciseText = exercise.Trim();
            int wordCount = 0, index = 0;

            while (index < exerciseText.Length)
            {
                // check if current char is part of a word
                while (index < exerciseText.Length &&
!char.IsWhiteSpace(exerciseText[index]))
                    index++;

                wordCount++;

                // skip whitespace until next word
                while (index < exerciseText.Length &&
char.IsWhiteSpace(exerciseText[index]))
                    index++;
            }


            int originalLength = wordCount * 750;

            if (originalLength < exerciseDuration)
                rate = rate - 30;



            string[] originalWords = original.Split(' ');
            string[] exerciseWords = exercise.Split(' ');

            for(int i = 0; i <= originalWords.Length; i++)
            {
                if(originalWords[i] != exerciseWords[i])
                {
```

```
                    if(rate>15)
                        rate = rate - 15;
                }

            }
            return rate;
        }
    }
}
```

Queries

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using ReadingBot.Models;
namespace ReadingBot.Queries
{
    public class DificultyQuery
    {
        public static Dificulty GetDificultyByID(Dificulty searchDificulty)
        {
            IEnumerable<Dificulty> dificulties;
            using (LearningAIDBEntities context = new LearningAIDBEntities())
            {
                dificulties =
                    from dificulty in context.Dificulties
                    where dificulty.DificultyID == searchDificulty.DificultyID
                    select dificulty;
                if (dificulties.ToList().Count == 0)
                    return new Dificulty(1, "hello, the querie didnt work");
                else
                    return dificulties.First<Dificulty>();
            }

        }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using ReadingBot.Models;
namespace ReadingBot.Queries
{
    public class ExerciseQuery
    {
        public static List<Exercis> GetExercisByDificulty(Dificulty dificulty)
        {
            IEnumerable<Exercis> exercises;
            using (LearningAIDBEntities context = new LearningAIDBEntities())
            {
                exercises =
                    from exercis in context.Exercises
                    where exercis.Dificulty.DificultyID == dificulty.DificultyID
                    select exercis;
                return exercises.ToList<Exercis>();
            }

        }
        public static List<Exercis> GetExercisOfLevelWithLogs(int level)
        {

            using(LearningAIDBEntities context = new LearningAIDBEntities())
            {
                var exercises = context.Exercises
                    .Include("Logs_Excercises")
                    .Where(b => b.Diculty == level);
                List<Exercis> lista = exercises.ToList();
                return lista;

            }
        }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using ReadingBot.Models;
namespace ReadingBot.Queries
{
    public class LogQuery
    {
        public static Logs_Excercises GetLogByExerciseIDandUserID(Exercis exercis, User
user)
        {
            IEnumerable<Logs_Excercises> logs;
            using (LearningAIDBEntities context = new LearningAIDBEntities())
            {

                logs =
                    from Logs_Excercises in context.Logs_Excercises
                    where Logs_Excercises.UserID == user.UserID &&
Logs_Excercises.ExercisesID == exercis.ExerciseID
                    select Logs_Excercises;
                return logs.First<Logs_Excercises>();
            }

        }
        public static void AddNewLog(Logs_Excercises log)
        {
            using (LearningAIDBEntities context = new LearningAIDBEntities())
            {
                context.Logs_Excercises.Add(log);
                context.SaveChanges();
            }
        }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using ReadingBot.Models;
using ReadingBot.Queries;
namespace ReadingBot.Queries
{
    public class UserQuery
    {
        public static User GetUserByName(User searchUser)
        {

            using (LearningAIDBEntities context = new LearningAIDBEntities())
            {
                var users = context.Users
                        .Include("Dificulty")
                        .Where(b => (b.Name == searchUser.Name && b.Surname ==
searchUser.Surname))
                        .ToList();
                if (users.Count == 0)
                {
                    return null;
                }else
                    return users.First();
            }

        }
        public static void AddUser(User user)
        {
            using (LearningAIDBEntities context = new LearningAIDBEntities())
            {
                user.Progress = 1;
                context.Users.Add(user);
                context.SaveChanges();
            }
        }
        public static void LevelUpUser(User user)
        {
            using (LearningAIDBEntities context = new LearningAIDBEntities())
            {
                var userDB = context.Users
                    .Single(b => b.UserID == user.UserID);
                if (userDB != null)
                {
                    userDB.Progress = user.Progress + 1;
                    context.SaveChanges();
                }
            }

        }
    }
}
```

Services:

```
namespace ReadingBot.Services
{
    using System;
    using System.Net.Http;
    using System.Threading;
    using System.Web;
    using System.Web.Configuration;

    public sealed class Authentication
    {
        // The token has an expiry time of 10 minutes
https://www.microsoft.com/cognitive-services/en-us/Speech-api/documentation/API-
Reference-REST/BingVoiceRecognition
        private const int TokenExpiryInSeconds = 600;

        private static readonly object LockObject;
        private static readonly string ApiKey;
        private string token;
        private Timer timer;

        static Authentication()
        {
            LockObject = new object();
            ApiKey = WebConfigurationManager.AppSettings["MicrosoftSpeechApiKey"];
        }

        private Authentication()
        {
        }

        public static Authentication Instance { get; } = new Authentication();

        /// <summary>
        /// Gets the current access token.
        /// </summary>
        /// <returns>Current access token</returns>
        public string GetAccessToken()
        {
            // Token will be null first time the function is called.
            if (this.token == null)
            {
                lock (LockObject)
                {
                    // This condition will be true only once in the lifetime of the
application
                    if (this.token == null)
                    {
                        this.RefreshToken();
                    }
                }
            }
```

```
        /// <summary>
        /// Issues a new AccessToken from the Speech Api
        /// </summary>
        /// This method couldn't be async because we are calling it inside of a lock.
        /// <returns>AccessToken</returns>
        private static string GetNewToken()
        {
            using (var client = new HttpClient())
            {
                client.DefaultRequestHeaders.Add("Ocp-Apim-Subscription-Key", ApiKey);

                var response =
client.PostAsync("https://westus.api.cognitive.microsoft.com/sts/v1.0/issueToken",
null).Result;
                if (response.StatusCode != System.Net.HttpStatusCode.OK)
                {
                    throw new HttpException((int)response.StatusCode,
$"({response.StatusCode}) {response.ReasonPhrase}");
                }

                return response.Content.ReadAsStringAsync().Result;
            }
        }

        /// <summary>
        /// Refreshes the current token before it expires. This method will refresh the
current access token.
        /// It will also schedule itself to run again before the newly acquired token's
expiry by one minute.
        /// </summary>
        private void RefreshToken()
        {
            this.token = GetNewToken();
            this.timer?.Dispose();
            this.timer = new Timer(
                x => this.RefreshToken(),
                null,

TimeSpan.FromSeconds(TokenExpiryInSeconds).Subtract(TimeSpan.FromMinutes(1)), //
Specifies the delay before RefreshToken is invoked.
                TimeSpan.FromMilliseconds(-1)); // Indicates that this function will only
run once
        }
    }
}
```

```
namespace ReadingBot.Services
{
    using System;
    using System.IO;
    using System.Linq;
    using System.Net.Http;
    using System.Net.Http.Headers;
    using System.Threading.Tasks;
    using System.Web;
    using Microsoft.Bot.Connector;
    using Newtonsoft.Json;
    using static System.Collections.Generic.HashSet<string>;

    public class MicrosoftCognitiveSpeechService
    {

        public async Task<string> ReturnTextFromAudio(Activity activity)
        {
            string message;

            var connector = new ConnectorClient(new Uri(activity.ServiceUrl));
            var audioAttachment = activity.Attachments?.FirstOrDefault(a =>
a.ContentType.Equals("audio/wav") || a.ContentType.Equals("application/octet-stream"));
            if (audioAttachment != null)
            {
                var stream = await GetAudioStream(connector, audioAttachment);
                var text = await this.GetTextFromAudioAsync(stream);
                message = ProcessText(text);
                return message;
            }
            else
            {
                return "Error";
            }

        }
        /// <summary>
        /// Gets text from an audio stream.
        /// </summary>
        /// <param name="audiostream"></param>
        /// <returns>Transcribed text. </returns>
        public async Task<string> GetTextFromAudioAsync(Stream audiostream)
        {
            var requestUri =
@"https://westus.stt.speech.microsoft.com/speech/recognition/conversation/cognitiveservic
es/v1?cid=753eb865-ad38-464e-8745-356546ffda45";

            using (var client = new HttpClient())
            {
                var token = Authentication.Instance.GetAccessToken();
                client.DefaultRequestHeaders.Add("Authorization", "Bearer " + token);

                using (var binaryContent = new
ByteArrayContent(StreamToBytes(audiostream)))
```

```
                    binaryContent.Headers.TryAddWithoutValidation("content-type",
"audio/wav; codec=\"audio/pcm\"; samplerate=16000");

                    var response = await client.PostAsync(requestUri, binaryContent);
                    if (response.StatusCode != System.Net.HttpStatusCode.OK)
                    {
                        throw new HttpException((int)response.StatusCode,
$"({response.StatusCode}) {response.ReasonPhrase}");
                    }

                    var responseString = await response.Content.ReadAsStringAsync();

                    try
                    {
                        dynamic data = JsonConvert.DeserializeObject(responseString);
                        dynamic first = data.DisplayText;




                        return first;
                    }
                    catch (JsonReaderException ex)
                    {
                        throw new Exception(responseString, ex);
                    }
                }
            }
        }

        /// <summary>
        /// Converts Stream into byte[].
        /// </summary>
        /// <param name="input">Input stream</param>
        /// <returns>Output byte[]</returns>
        private static byte[] StreamToBytes(Stream input)
        {
            using (MemoryStream ms = new MemoryStream())
            {
                input.CopyTo(ms);
                return ms.ToArray();
            }
        }
        private static string ProcessText(string text)
        {
            string message = text;


            return message;
        }
        private static async Task<Stream> GetAudioStream(ConnectorClient connector,
Attachment audioAttachment)
        {
            using (var httpClient = new HttpClient())
            {
```

```
                var uri = new Uri(audioAttachment.ContentUrl);
                if (uri.Host.EndsWith("skype.com") && uri.Scheme == "https")
                {
                    httpClient.DefaultRequestHeaders.Authorization = new
AuthenticationHeaderValue("Bearer", await GetTokenAsync(connector));
                    httpClient.DefaultRequestHeaders.Accept.Add(new
MediaTypeWithQualityHeaderValue("application/octet-stream"));
                }

                return await httpClient.GetStreamAsync(uri);
            }
        }
        private static async Task<string> GetTokenAsync(ConnectorClient connector)
        {
            var credentials = connector.Credentials as MicrosoftAppCredentials;
            if (credentials != null)
            {
                return await credentials.GetTokenAsync();
            }

            return null;
        }
    }
```

```
using Microsoft.Bot.Connector;
using Newtonsoft.Json.Linq;
using System;
using System.Collections;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Net.Http;
using System.Net.Http.Headers;
using System.Threading.Tasks;
using System.Web;

namespace ReadingBot.Services
{
    [Serializable]
    public class Emotion
    {
        public Emotion(double anger, double contempt, double disgust, double fear, double
happyness, double neutral, double sadness, double surprise)
        {
            this.anger = anger;
            this.contempt = contempt;
            this.disgust = disgust;
            this.fear = fear;
            this.happyness = happyness;
            this.neutral = neutral;
            this.sadness = sadness;
            this.surprise = surprise;
        }
        public double anger;
        public double contempt;
        public double disgust;
        public double fear;
        public double happyness;
        public double neutral;
        public double sadness;
        public double surprise;

        public string toString()
        {
            string s = $"Your emotions were: anger:{this.anger},
contempt:{this.contempt}" +
                $", disgust:{this.disgust}, fear:{fear}, happiness:{this.happyness}, " +
                $"neutral:{this.neutral}, sadness:{this.sadness},
surprise:{this.surprise}";
            return s;
        }
    }
    public class MicrosoftFaceApiService
    {
        public static async Task<Emotion> getEmotions(Activity activity)
        {
            string imageURL = activity.Attachments[0].ContentUrl;
```

```
      //Peticion de las emociones de la foto
            var client = new HttpClient();

            //Header con la clave de suscripción
            client.DefaultRequestHeaders.Add("Ocp-Apim-Subscription-Key",
"1805264c3b0e4c0889e94fe846003d97");

            string requestParameters = "returnFaceId=true&returnFaceLandmarks=false" +
                "&returnFaceAttributes=age,gender,headPose,smile,facialHair,glasses," +
                "emotion,hair,makeup,occlusion,accessories,blur,exposure,noise";

            string uri = "https://westus.api.cognitive.microsoft.com/face/v1.0/detect" +
"?" + requestParameters;

            HttpResponseMessage responseEmotionRequest;
            string responseEmotionRequestContent;


            var httpClient = new HttpClient();

            Stream imgStream = await httpClient.GetStreamAsync(imageURL);
            byte[] imageBytes;
            byte[] buffer = new byte[16 * 1024];
            using (MemoryStream ms = new MemoryStream())
            {
                imgStream.CopyTo(ms);
                imageBytes = ms.ToArray();
            }
            using (var content = new ByteArrayContent(imageBytes))
            {
                content.Headers.ContentType = new
MediaTypeHeaderValue("application/octet-stream");
                responseEmotionRequest = await client.PostAsync(uri, content);
                IList lista = await responseEmotionRequest.Content.ReadAsAsync<IList>();
                Emotion emotion = null;
                foreach (JObject objec in lista)
                {
                    try
                    {
                        var anger = (double)objec["faceAttributes"]["emotion"]["anger"];
                        var happiness =
(double)objec["faceAttributes"]["emotion"]["happiness"];
                        var contempt =
(double)objec["faceAttributes"]["emotion"]["contempt"];
                        var disgust =
(double)objec["faceAttributes"]["emotion"]["disgust"];
                        var fear = (double)objec["faceAttributes"]["emotion"]["fear"];
                        var neutral =
(double)objec["faceAttributes"]["emotion"]["neutral"];
                        var sadness =
(double)objec["faceAttributes"]["emotion"]["sadness"];
                        var surprise =
(double)objec["faceAttributes"]["emotion"]["surprise"];
                        emotion = new Emotion(anger, happiness, contempt, disgust, fear,
neutral, sadness, surprise);
```

```
                    }catch(Exception e){ }
                }
                return emotion;
            }
        }
    }

}
```

```
using System;
using System.Collections.Generic;
using System.Text;
namespace ReadingBot.Services
{
    class NumeroALetra
    {
        public static string enletras(string num)
        {
            string res, dec = "";
            Int64 entero;
            int decimales;
            double nro;

            try

            {
                nro = Convert.ToDouble(num);
            }
            catch
            {
                return "";
            }

            entero = Convert.ToInt64(Math.Truncate(nro));
            decimales = Convert.ToInt32(Math.Round((nro - entero) * 100, 2));
            if (decimales > 0)
            {
                dec = " CON " + decimales.ToString() + "/100";
            }

            res = toText(Convert.ToDouble(entero)) + dec;
            return res;
        }

        private static string toText(double value)
        {
            string Num2Text = "";
            value = Math.Truncate(value);
            if (value == 0) Num2Text = "CERO";
            else if (value == 1) Num2Text = "UNO";
            else if (value == 2) Num2Text = "DOS";
            else if (value == 3) Num2Text = "TRES";
            else if (value == 4) Num2Text = "CUATRO";
            else if (value == 5) Num2Text = "CINCO";
            else if (value == 6) Num2Text = "SEIS";
            else if (value == 7) Num2Text = "SIETE";
            else if (value == 8) Num2Text = "OCHO";
            else if (value == 9) Num2Text = "NUEVE";
            else if (value == 10) Num2Text = "DIEZ";
            else if (value == 11) Num2Text = "ONCE";
            else if (value == 12) Num2Text = "DOCE";
            else if (value == 13) Num2Text = "TRECE";
            else if (value == 14) Num2Text = "CATORCE";
            else if (value == 15) Num2Text = "QUINCE";
```

```
            else if (value == 20) Num2Text = "VEINTE";
            else if (value < 30) Num2Text = "VEINTI" + toText(value - 20);
            else if (value == 30) Num2Text = "TREINTA";
            else if (value == 40) Num2Text = "CUARENTA";
            else if (value == 50) Num2Text = "CINCUENTA";
            else if (value == 60) Num2Text = "SESENTA";
            else if (value == 70) Num2Text = "SETENTA";
            else if (value == 80) Num2Text = "OCHENTA";
            else if (value == 90) Num2Text = "NOVENTA";
            else if (value < 100) Num2Text = toText(Math.Truncate(value / 10) * 10) + " Y
" + toText(value % 10);
            else if (value == 100) Num2Text = "CIEN";
            else if (value < 200) Num2Text = "CIENTO " + toText(value - 100);
            else if ((value == 200) || (value == 300) || (value == 400) || (value == 600)
|| (value == 800)) Num2Text = toText(Math.Truncate(value / 100)) + "CIENTOS";
            else if (value == 500) Num2Text = "QUINIENTOS";
            else if (value == 700) Num2Text = "SETECIENTOS";
            else if (value == 900) Num2Text = "NOVECIENTOS";
            else if (value < 1000) Num2Text = toText(Math.Truncate(value / 100) * 100) +
" " + toText(value % 100);
            else if (value == 1000) Num2Text = "MIL";
            else if (value < 2000) Num2Text = "MIL " + toText(value % 1000);
            else if (value < 1000000)
            {
                Num2Text = toText(Math.Truncate(value / 1000)) + " MIL";
                if ((value % 1000) > 0) Num2Text = Num2Text + " " + toText(value % 1000);
            }

            else if (value == 1000000) Num2Text = "UN MILLON";
            else if (value < 2000000) Num2Text = "UN MILLON " + toText(value % 1000000);
            else if (value < 1000000000000)
            {
                Num2Text = toText(Math.Truncate(value / 1000000)) + " MILLONES ";
                if ((value - Math.Truncate(value / 1000000) * 1000000) > 0) Num2Text =
Num2Text + " " + toText(value - Math.Truncate(value / 1000000) * 1000000);
            }

            else if (value == 1000000000000) Num2Text = "UN BILLON";
            else if (value < 2000000000000) Num2Text = "UN BILLON " + toText(value -
Math.Truncate(value / 1000000000000) * 1000000000000);

            else
            {
                Num2Text = toText(Math.Truncate(value / 1000000000000)) + " BILLONES";
                if ((value - Math.Truncate(value / 1000000000000) * 1000000000000) > 0)
Num2Text = Num2Text + " " + toText(value - Math.Truncate(value / 1000000000000) *
1000000000000);
            }
            return Num2Text;

        }

    }

}
```

Webconfig:

```
<?xml version="1.0" encoding="utf-8"?>
<!--
  For more information on how to configure your ASP.NET application, please visit
  http://go.microsoft.com/fwlink/?LinkId=301879
  -->
<configuration>
  <configSections>
    <!-- For more information on Entity Framework configuration, visit
http://go.microsoft.com/fwlink/?LinkID=237468 -->
    <section name="entityFramework"
type="System.Data.Entity.Internal.ConfigFile.EntityFrameworkSection, EntityFramework,
Version=6.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089"
requirePermission="false" />
  </configSections>
  <appSettings>
    <!-- update these with your BotId, Microsoft App Id and your Microsoft App Password--
>
    <add key="BotId" value="YourBotId" />
    <add key="MicrosoftAppId" value="" />
    <add key="MicrosoftAppPassword" value="" />
    <add key="MicrosoftSpeechApiKey" value="6d4beb4dbecb44aaaef9ca3fea831af9" />
  </appSettings>
  <!--
    For a description of web.config changes see
http://go.microsoft.com/fwlink/?LinkId=235367.

    The following attributes can be set on the <httpRuntime> tag.
      <system.Web>
        <httpRuntime targetFramework="4.6" />
      </system.Web>
  -->
  <system.web>
    <customErrors mode="Off" />
    <compilation debug="true" targetFramework="4.6" />
    <httpRuntime targetFramework="4.6" />
  </system.web>
  <system.webServer>
    <defaultDocument>
      <files>
        <clear />
        <add value="default.htm" />
      </files>
    </defaultDocument>
    <handlers>
      <remove name="ExtensionlessUrlHandler-Integrated-4.0" />
      <remove name="OPTIONSVerbHandler" />
      <remove name="TRACEVerbHandler" />
      <add name="ExtensionlessUrlHandler-Integrated-4.0" path="*." verb="*"
type="System.Web.Handlers.TransferRequestHandler"
preCondition="integratedMode,runtimeVersionv4.0" />
    </handlers>
  </system.webServer>
```

```
  <runtime>
    <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
      <dependentAssembly>
        <assemblyIdentity name="Autofac" publicKeyToken="17863AF14B0044DA"
culture="neutral" />
        <bindingRedirect oldVersion="0.0.0.0-4.6.0.0" newVersion="4.6.0.0" />
      </dependentAssembly>
      <dependentAssembly>
        <assemblyIdentity name="System.Web.Helpers" publicKeyToken="31bf3856ad364e35" />
        <bindingRedirect oldVersion="1.0.0.0-3.0.0.0" newVersion="3.0.0.0" />
      </dependentAssembly>
      <dependentAssembly>
        <assemblyIdentity name="System.Web.Mvc" publicKeyToken="31bf3856ad364e35" />
        <bindingRedirect oldVersion="1.0.0.0-5.2.3.0" newVersion="5.2.3.0" />
      </dependentAssembly>
      <dependentAssembly>
        <assemblyIdentity name="System.Web.WebPages" publicKeyToken="31bf3856ad364e35" />
        <bindingRedirect oldVersion="1.0.0.0-3.0.0.0" newVersion="3.0.0.0" />
      </dependentAssembly>
      <dependentAssembly>
        <assemblyIdentity name="Newtonsoft.Json" publicKeyToken="30ad4fe6b2a6aeed"
culture="neutral" />
        <bindingRedirect oldVersion="0.0.0.0-10.0.0.0" newVersion="10.0.0.0" />
      </dependentAssembly>
      <dependentAssembly>
        <assemblyIdentity name="System.Net.Http.Primitives"
publicKeyToken="b03f5f7f11d50a3a" culture="neutral" />
        <bindingRedirect oldVersion="0.0.0.0-4.2.29.0" newVersion="4.2.29.0" />
      </dependentAssembly>
      <dependentAssembly>
        <assemblyIdentity name="System.Net.Http.Formatting"
publicKeyToken="31bf3856ad364e35" culture="neutral" />
        <bindingRedirect oldVersion="0.0.0.0-5.2.3.0" newVersion="5.2.3.0" />
      </dependentAssembly>
      <dependentAssembly>
        <assemblyIdentity name="System.IdentityModel.Tokens.Jwt"
publicKeyToken="31bf3856ad364e35" culture="neutral" />
        <bindingRedirect oldVersion="0.0.0.0-5.1.4.0" newVersion="5.1.4.0" />
      </dependentAssembly>
      <dependentAssembly>
        <assemblyIdentity name="Autofac" publicKeyToken="17863af14b0044da"
culture="neutral" />
        <bindingRedirect oldVersion="0.0.0.0-4.6.0.0" newVersion="4.6.0.0" />
      </dependentAssembly>
      <dependentAssembly>
        <assemblyIdentity name="Microsoft.Bot.Connector"
publicKeyToken="31bf3856ad364e35" culture="neutral" />
        <bindingRedirect oldVersion="0.0.0.0-3.15.3.0" newVersion="3.15.3.0" />
      </dependentAssembly>
      <dependentAssembly>
        <assemblyIdentity name="Microsoft.Bot.Builder" publicKeyToken="31bf3856ad364e35"
culture="neutral" />
        <bindingRedirect oldVersion="0.0.0.0-3.15.3.0" newVersion="3.15.3.0" />
      </dependentAssembly>
```

```
<dependentAssembly>
        <assemblyIdentity name="Microsoft.Azure.Documents.Client"
publicKeyToken="31bf3856ad364e35" culture="neutral" />
        <bindingRedirect oldVersion="0.0.0.0-1.11.0.0" newVersion="1.11.0.0" />
      </dependentAssembly>
      <dependentAssembly>
        <assemblyIdentity name="Microsoft.Bot.Builder.Autofac"
publicKeyToken="31bf3856ad364e35" culture="neutral" />
        <bindingRedirect oldVersion="0.0.0.0-3.15.3.0" newVersion="3.15.3.0" />
      </dependentAssembly>
      <dependentAssembly>
        <assemblyIdentity name="Microsoft.Bot.Builder.History"
publicKeyToken="31bf3856ad364e35" culture="neutral" />
        <bindingRedirect oldVersion="0.0.0.0-3.15.3.0" newVersion="3.15.3.0" />
      </dependentAssembly>
      <dependentAssembly>
        <assemblyIdentity name="Microsoft.Data.Services.Client"
publicKeyToken="31bf3856ad364e35" culture="neutral" />
        <bindingRedirect oldVersion="0.0.0.0-5.7.0.0" newVersion="5.7.0.0" />
      </dependentAssembly>
      <dependentAssembly>
        <assemblyIdentity name="Microsoft.Data.OData" publicKeyToken="31bf3856ad364e35"
culture="neutral" />
        <bindingRedirect oldVersion="0.0.0.0-5.7.0.0" newVersion="5.7.0.0" />
      </dependentAssembly>
      <dependentAssembly>
        <assemblyIdentity name="Microsoft.Data.Edm" publicKeyToken="31bf3856ad364e35"
culture="neutral" />
        <bindingRedirect oldVersion="0.0.0.0-5.7.0.0" newVersion="5.7.0.0" />
      </dependentAssembly>
    </assemblyBinding>
  </runtime>
  <entityFramework>
    <defaultConnectionFactory
type="System.Data.Entity.Infrastructure.LocalDbConnectionFactory, EntityFramework">
      <parameters>
        <parameter value="mssqllocaldb" />
      </parameters>
    </defaultConnectionFactory>
    <providers>
      <provider invariantName="System.Data.SqlClient"
type="System.Data.Entity.SqlServer.SqlProviderServices, EntityFramework.SqlServer" />
    </providers>
  </entityFramework>
  <connectionStrings>
    <add name="LearningAIDBEntities"
connectionString="metadata=res://*/Models.BotData.csdl|res://*/Models.BotData.ssdl|res://
*/Models.BotData.msl;provider=System.Data.SqlClient;provider connection string=&quot;data
source=learningaiserver.database.windows.net;initial catalog=LearningAIDB;user
id=learningai;password=Nonsomus1;MultipleActiveResultSets=True;App=EntityFramework&quot;"
providerName="System.Data.EntityClient" />
  </connectionStrings>
</configuration>
```

ANEXO B

En este anexo se puede encontrar el Código de creación de la base de datos.

```
CREATE TABLE Dificulties (
	DificultyID int NOT NULL IDENTITY(1,1) PRIMARY KEY,
	Text varchar(255) NOT NULL
);
CREATE TABLE Exercises (
	ExerciseID int NOT NULL IDENTITY(1,1) PRIMARY KEY,
	Text varchar(1023) NOT NULL,
	Diculty int NOT NULL FOREIGN KEY REFERENCES Dificulties(DificultyID)
);
CREATE TABLE Users (
	UserID int NOT NULL IDENTITY(1,1) PRIMARY KEY,
	Name varchar(255) NOT NULL,
	Surname varchar(255) NOT NULL,
	Progress int NOT NULL FOREIGN KEY REFERENCES Dificulties(DificultyID),
	FaceID varchar(255)
);
CREATE TABLE Logs_Excercises (
	LogID int NOT NULL IDENTITY(1,1) PRIMARY KEY,
	UserID int NOT NULL FOREIGN KEY REFERENCES Users(UserID),
	ExercisesID int NOT NULL FOREIGN KEY REFERENCES Exercises(ExerciseID),
	Performance int NOT NULL,
	Timestamp datetime NOT NULL
);
```