Centro de Investigación y de Estudios Avanzados
del Instituto Politécnico Nacional

Unidad Zacatenco

Departamento de Computación

Soporte de Interacciones Profesor/Alumno Mediante un Chatbot

Tesis que presenta

Víctor Hugo Espinoza Sixtos

para obtener el Grado de

Maestro en Ciencias en Computación

Directora de la Tesis

Dra. Sonia Guadalupe Mendoza Chapa

Co-Director de la Tesis

Dr. José Guadalupe Rodríguez García

Ciudad de México, México **Noviembre 2021**

Resumen

A lo largo de la historia se han presentado diversos cambios, los cuales afectan directa o indirectamente a la sociedad, aportando y modificando diversos aspectos en ella. Un ejemplo de ello es la forma en la que el uso de la tecnología ha influenciado la vida del ser humano: desde una forma primitiva, que ayudó a sobrevivir al hombre, hasta las tecnologías actuales que son primordiales incluso para actividades recreativas. De ahí que la frase «Las tecnologías más profundas son las que desaparecen. Se entretejen en la trama de la vida cotidiana hasta que no se distinguen de ella» de Mark Weiser, en su multicitado artículo "La Computadora del siglo XXI", toma una relevancia importante, brindando un panorama de cómo la tecnología ha cambiado el desarrollo de la sociedad, al punto que es parte fundamental en la vida cotidiana de sus individuos, aun cuando esta pueda ser invisible a los ojos de los mismos.

Un área en específico que está cambiando con la ayuda de la tecnología es la educación. Esto ha quedado particularmente demostrado en la actual crisis sanitaria producto de la pandemia del COVID 19, forzando a trasladar gran parte de los modelos tradicionales a formatos a distancia. No obstante, también ha quedado al descubierto la falta de herramientas que permitan a los alumnos no sólo memorizar, sino entender problemas complejos y crear reflexiones que posibiliten solucionarlos. Ante esta carencia, la tecnología debe jugar un papel vital, servir como puente entre los diferentes actores inmersos en la educación, facilitando la colaboración y vínculo entre ellos, liberando de ciertas cargas de trabajo y ayudando con el control de actividades.

Entre las nuevas tecnologías, encontramos a los chatbots, los cuales se pueden definir como agentes conversacionales previamente programados, que sirven para mantener conversaciones, comúnmente, en formato de texto y voz. Su aplicación y valía en la educación han sido probados en diversos trabajos, sin embargo, la mayoría de estas investigaciones han pasado por alto la inclusión de los diversos actores que componen el proceso de enseñanza/aprendizaje, centrándose sólo en los alumnos. Esto ha llevado a que se creen soluciones parciales, lo que dificulta la integración de los chatbots a las actividades diarias en las aulas, viéndose reducidos a un simple mecanismo para tareas limitadas.

Es por ello que en el presente trabajo de tesis se propone un chatbot que asista al proceso de enseñanza/aprendizaje dentro de secundarias mexicanas. Este sistema toma en cuenta las necesidades de los tres entes principales en la educación: alumnos, maestros y personal administrativo. No se pretende sustituir a ninguna parte, sino aportarles herramientas que permitan hacer más sencillas sus tareas, incentivar y facilitar la comunicación, así como crear puentes que ayuden a la transición de modelos educativos tradicionales a aquellos soportados por tecnología.

Abstract

Throughout history, there have been various changes that directly or indirectly affect society, contributing and modifying various aspects. An example of this is how technology has influenced human life: from a primitive form, which helped man survive, to current technologies that are essential even for recreational activities. Hence, the phrase "The deepest technologies are those that disappear. They are woven into the fabric of everyday life until they are indistinguishable from it" by Mark Weiser, in his much-quoted article "The Computer of the 21st Century", takes on a crucial relevance, providing an overview of how technology has changed the development of society, to the point that it is a fundamental part of the daily life of its individuals, even though it may be invisible to their eyes.

One specific area that is changing with the help of technology is education. This has been particularly demonstrated in the current health crisis resulting from the COVID 19 pandemic, forcing a large part of the traditional models to be transferred to remote formats. However, the lack of tools that allow students to understand complex problems and create reflections to solve them instead of just memorizing things has also been exposed. To face this need, technology must play a vital role, serve as a bridge between the different actors involved in education, facilitating collaboration and bonding between them, freeing up certain workloads and helping to control activities.

Among the new technologies, we find chatbots, defined as previously programmed conversational agents, which commonly serve to hold conversations in text and voice formats. Its application and value in education have been proven in various works. However, most of this research has overlooked the various actors that make up the teaching/learning process, focusing only on students. This has led to partial solutions, which makes it challenging to integrate chatbots into daily activities in the classroom, being reduced to a simple mechanism for limited tasks.

In this thesis work, a chatbot is proposed to assist Mexican middle schools teaching/learning process. This system considers the needs of the three primary entities needs: students, teachers and administrative personnel. It is not intended to replace any part but to provide them with tools that make their tasks easier, encourage and facilitate communication, and create bridges that help the transition from traditional educational models to those supported by technology.

Agradecimientos

A mi familia que me brindo las bases como ser humano para sobrellevar retos a pesar de obstáculos que nos han aquejado, gracias por aguantar en algunos de mis desvelos, caídas y logros, siendo que todos sus integrantes son parte fundamental de mi motor para dar lo mejor, pensando en poder contribuir en un mejor futuro por mínimo que sea.

A CONACyT y al CINVESTAV IPN por brindar apoyo económico y el entorno para poder realizar mis estudios de maestría.

A los directores de la presente tesis, la Dra. Sonia G. Mendoza Chapa y al Dr. José G. Rodríguez García, siendo que sin su ayuda, consejos y revisiones no hubiese sido posible sacar a flote dicha tesis.

Al personal administrativo y profesores del departamento de Computación del CINVESTAV quienes pudieron apoyarme en distintas labores y brindar experiencias que sirvieron en mi formación.

A amigos y personas que estimo con mi corazón, siendo que si bien no puedo mencionar a todas por lo corto del papel y lo agradecido que estoy, les puedo decir gracias por sacarme risas, levantar el ánimo, soportarme, dar consejos y palabras en diferentes momentos, por esos amigos que pude encontrar en el trabajo, la escuela, talleres, incluso en lugares de ocio, viajes, redes sociales y deportivas, gracias por esas vueltas a la pista, comidas ocasionales, platicas en coche o bosques, salidas a diferentes lugares, caminatas o sucesos importantes incluso en lugares menos pensados como estacionamientos.

A alumnos con los que tuve el honor de compartir clases y los pocos conocimientos que tengo.

A los profesores que mencionare y a muchos otros que omito por espacio, los cuales marcaron mi camino. Gracias al Dr. Félix Mata y al Dr. Roberto Zagal, Dr. Casimiro Linares, entre otros, por impulsarme a querer indagar e investigar temas por mínimo que fuesen. A profesores que cuando no me quedaban fuerzas y pensé en dejar de lado mis estudios a temprana edad me dieron las palabras y conocimientos útiles como el Dr. Ángel Hernández. A maestras como Carmen Jacobo que me dieron el apoyo para formarme no solo en la parte de ingeniería, si no ver en el arte algo muy grato y una ventana a mundos mejores.

Incluso gracias a quienes que me cerraron en su momento las puertas, que con palabras duras y experiencias agotadoras me pusieron obstáculos, porque a base de esto pude crecer

como persona y ver que si caemos podemos levantarnos, seguir y aprender.

Gracias a todo lo que omito por olvido y conveniencia, pero que de alguna forma me han contribuido en poco o mucho.

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

1.1 Antecedentes y motivación

Las TICs, Tecnologías de la Información y Comunicaciones por sus siglas, según Graells (2000), son un conjunto de avances tecnológicos, posibilitados por la informática, las telecomunicaciones y las tecnologías audiovisuales, las cuales proporcionan herramientas para el tratamiento y difusión de la información por diversos canales de comunicación, siendo el elemento más poderoso que integra las TICs el Internet, que ha llevado a la configuración de la llamada Sociedad de la Información. El mismo autor vincula las TICs con la educación, mencionando que su importancia tiene que ver con ser un medio de expresión para la creación, canal de comunicación, instrumento para procesar información, fuente de información, organización y gestión de tutorías, así como recursos interactivos para el aprendizaje (Graells, 2013).

Entendiendo este vínculo entre tecnología y educación, nace el *e-learning*, que Lozano (2005) define como: «La formación que se imparte mediante el uso de las nuevas tecnologías, por tanto, su distinción respecto con la educación tradicional se centra justamente en la enorme potencialidad y oportunidades que nos ofrecen las TIC para ser usadas como medio excelente para formar a las personas». Otra definición es la de Garcia-Peñalvo (2014), que lo define como: «proceso formativo, de naturaleza intencional o no intencional, orientado a la adquisición de una serie de competencias y destrezas en un contexto social, que se desarrolla en un ecosistema tecnológico en el que interactúan diferentes perfiles de usuarios que comparten contenidos, actividades y experiencias, que en situaciones de aprendizaje formal, debe ser tutelado por actores docentes cuya actividad contribuya a garantizar la calidad de todos los factores involucrados». Mientras, para nosotros el *e-learning* es el proceso de formación educativa, enfocado en que, por medio del uso de las TIC, los diferentes actores en el proceso educativo se vean beneficiados al adquirir conocimiento, compartir información, realizar y controlar actividades.

Debido a la importancia del proceso de enseñanza-aprendizaje y la verificada efectividad en el apoyo de la educación por medio de tecnologías, se puede abordar una de ellas, los *chatbots*, definidos por (Rozga, 2018) como: «Un software que puede tomar una entrada de un usuario en lenguaje natural y devolver una respuesta en texto o en un formato multimedia a dicho usuario». Una definición propia de *chatbot* es la siguiente: «Los *chatbots* son sistemas de software, previamente programados bajo determinada arquitectura y lenguajes, especializados para dar respuestas en determinadas áreas. Son capaces de recibir y procesar entradas comúnmente en texto y voz, mantener conversaciones y ofrecer respuestas predefinidas, regularmente por canales de texto y voz. Hacen uso de procesamiento de lenguaje natural, coincidencia de patrones y entendimiento del contexto.».

Su implementación en entornos educativos toma gran relevancia en el panorama de la crisis actual, que ha dejado expuesta la necesidad de una faz más tecnológica para la educación y los involucrados en ella (Brown & Salmi, 2020). De ello que el *e-learning*, ayude como puente y conexión entre la tecnología y el sector educativo. La tecnología aplicada a la educación permite una mayor comunicación y participación entre los actores, así como un mejor control y gestión del material y los cursos. Se crea y busca entonces, un entorno donde la tecnología juegue un papel relevante en la educación, no sustituyendo roles actuales, sino sirviendo como una herramienta de soporte para alcanzar los objetivos de un curso (Basogain et al., 2017).

Para ello, se plantea que el *chatbot*, propuesto en la presente tesis, pueda ser parte de un sistema Web que integre varias funcionalidades que den soporte al proceso de enseñanza-aprendizaje en escuelas secundarias mexicanas. Se tomará como base un sistema ya creado y reportado en Mendoza et al. (2020), se corregirán errores, se mejorarán las funcionalidades ofrecidas y se crearán nuevas herramientas.

Respecto al *chatbot* (Mendoza et al., 2020) resultante de este proyecto de tesis, cabe mencionar que se tienen evaluaciones heurísticas realizadas con anterioridad (L. Sánchez, 2020), las cuales sugieren la factibilidad del mismo y sirven como punto de partida para explorar posibles mejoras de este sistema. Además, otros aspectos se agregan como resultado de las pruebas de funcionalidad [1] realizadas sobre el sistema, de manera que se pudiera vislumbrar el estado actual del mismo y corroborar las áreas de oportunidad, junto con los problemas

[1] La ISO (Organización Internacional para la Estandarización), en la *ISO/IEC 9126*, define la funcionalidad de software como la capacidad de un producto de software de proveer un conjunto de funciones que satisfagan las necesidades tanto implícitas como explicitas de los usuarios. Basandose en: apropiabilidad (proveer un conjunto de funciones que cumplan las tareas y objetivos del usuario), exactitud (brindar resultados correctos y con alto grado de precisión), interoperabilidad (interactuarcción del software con otros sistemas específicos), seguridad (capacidad del software de proteger datos) y conformidad en la funcionalidad (capacidad del software para ajustarse a estándares y regulaciones).

que se deben solucionar.

Consecuentemente, en el presente trabajo se realiza, en primer lugar, la familiarización del estado actual tanto del sistema como los módulos que lo conforman, e.g., el propio *chatbot*, así como las adecuaciones necesarias que den solución a los problemas y limitaciones actuales de este sistema. De esta manera, se obtendrá una propuesta que brinde soporte y sirva de enlace entre los actores implicados, favoreciendo el proceso enseñanza-aprendizaje.

1.2 Planteamiento del problema

Actualmente prevalece, al menos en países como México y en gran parte de América Latina, una tendencia a llevar la educación por caminos muy tradicionales, favoreciendo estrategias que, si bien parecen ser efectivas, quedan rezagadas a una época donde las tecnologías no tenían el impacto de hoy, en gran medida debido al rezago económico y social Miklos & Arroyo (2008). Un ejemplo de que aún se opta por métodos tradicionales (desde cierto punto de vista antiguos) se aborda en el trabajo de Basogain et al. (2017) donde se analizan las pruebas actuales en educación como PISA (*Programme for International Student Assessmen*) y la manera de enseñar usando problemas deterministas, los cuales sólo dan un panorama parcial de la educación, perjudicando al alumno en su aprendizaje real y favoreciendo únicamente la memorización de información. Además, se menciona que, con la ayuda de la tecnología aplicada a la educación, se pueden abordar paulatinamente problemas no deterministas, en los cuales radica la educación futura.

Además del ya mencionado rezago tecnológico, se tienen crisis que impactan de manera global y en diferentes ámbitos, como la pandemia del COVID-19, que afecta gravemente al área educativa, dado que ahora casi todas las clases se realizan a distancia. Consecuentemente, las herramientas para videoconferencias y para la gestión de clases en línea han tomado mayor relevancia. Sin embargo, estos sistemas no brindan todas las herramientas necesarias para que exista una comunicación adecuada entre los actores implicados en el proceso de enseñanza-aprendizaje. Cabe resaltar también que los estudiantes buscan satisfacer sus necesidades personales y requieren un grado de individualización en su aprendizaje. Sin embargo, este aspecto no ha sido satisfecho completamente por los medios tradicionales (Bollweg et al., 2019).

Los *chatbots* pueden ser una herramienta útil ante tal escenario, pues ayudan con parte de la carga de trabajo de los profesores, liberándoles de tareas tediosas con el fin de permitirles enforcarse en tareas relevantes. Además, los profesores se convierten en facilitadores del aprendizaje y los alumnos en elementos activos en su propio aprendizaje. Un ejemplo es

el *chatbot* de nuestro grupo de trabajo (Mendoza et al., 2020), el cual podría ayudar en el proceso enseñanza-aprendizaje en alumnos de nivel secundaria en México.

Sin embargo, dicho *chatbot* presenta problemas puntuales como:

- Una pobre interacción y limitado flujo de la conversación entre un usuario y el *chatbot*, a causa de su reducida base de conocimientos.
- La interfaz de usuario del sistema es, hasta cierto punto, pobre. Por ejemplo, no se adapta al tamaño de las pantallas.
- Hay acciones que se realizan a nivel interfaz de usuario, pero que no se ven reflejadas a nivel base de datos y por ende no surten efecto en el sistema. Por ejemplo, la recomendación y envío de trabajos. Estos se muestran en la interfaz de usuario e incluso al compartir o subir material, se notifica que las acciones se realizaron, pero no se ven reflejadas a nivel base de datos; incluso, si se accede por medio de otros roles, tampoco se ve efecto alguno.
- Existen errores al dar alta a un alumno en algún grupo.
- Hay errores al cruzar alumnos con sus respectivas materias.
- Se tiene un limitado conjunto de funciones para los tres roles definidos para el sistema, e.g., el profesor no puede ver claramente el progreso tanto individual como grupal de sus alumnos.
- Las respuestas que brinda el *chatbot* no permiten una retroalimentación, lo cual puede generar incertidumbre (e.g., el *chatbot* brinda una respuesta, pero no es útil para los estudiantes).
- Existen problemas al validar las tareas que puede realizar un usuario, dependiendo del estatus del mismo.
- Se presenta una violación de seguridad para usuarios inactivos y nuevos, debido a que, al momento de registrar un nuevo usuario en el sistema, se le dirige automáticamente a su menú de inicio; sin embargo, las opciones que se le presentan se encuentran habilitadas, lo cual rompe con las siguientes premisas de diseño: 1) un usuario se encuentra desactivado al momento de iniciar su registro y 2) hasta que un usuario administrador lo active, el usuario puede usar las herramientas disponibles para su rol en el sistema.
- Se usa una versión antigua de Dialogflow, pues el *chatbot* está diseñado con la versión 1 de *Dialogflow*, por lo que su conexión con el sistema se da con el uso de un token,

el cual ha quedado en desuso en la versión 2; por lo tanto, no se pueden efectuar actualizaciones ni mejoras, mientras no se migre a la segunda versión y se use el nuevo método de conexión que se ofrece.

- Existe poca información sobre el uso del sistema, debido a la carencia de documentación.

Un aspecto que se ha estudiado en trabajos como el de Li et al. (2017), es la influencia que tiene el hecho de dotar de cierta «personalidad» humana al *chatbot*, ya que esto puede influir, de manera positiva, en la interacción y resultados de los usuarios que lo usen. Sin embargo, actualmente el sistema no cuenta con dicha característica.

1.3 Objetivos

General

Implementar las características faltantes, al sistema de *chatbot*, así como mejorar las ya existentes, para que evolucione en una herramienta usable que apoye el proceso enseñanza-aprendizaje.

Especificos

1. Identificar y corregir errores del sistema anterior.
2. Incrementar la utilidad e interacción del sistema y el módulo de *chatbot*, mediante el reforzamiento de la base de conocimientos y el diseño de una interfaz más intuitiva y amigable.
3. Ofrecer un seguimiento estadístico del rendimiento de los estudiantes, para proporcionar sugerencias y alertas con base en sus calificaciones y trabajos.
4. Desarrollar las características que permitan al *chatbot* ofrecer un díalogo con mayor fluidez.
5. Realizar pruebas de funcionalidad para obtener la validación del sistema y del módulo de *chatbot*.

1.4 Organización del documento

El presente documento de tesis se encuentra dividido en 7 capítulos (ver Figura 1.1). En el segundo capítulo, «Marco Teórico», se presentan tanto conceptos como definiciones y tecnologías que dan fundamento teórico a este trabajo y que se usan a lo largo del documento

de tesis.

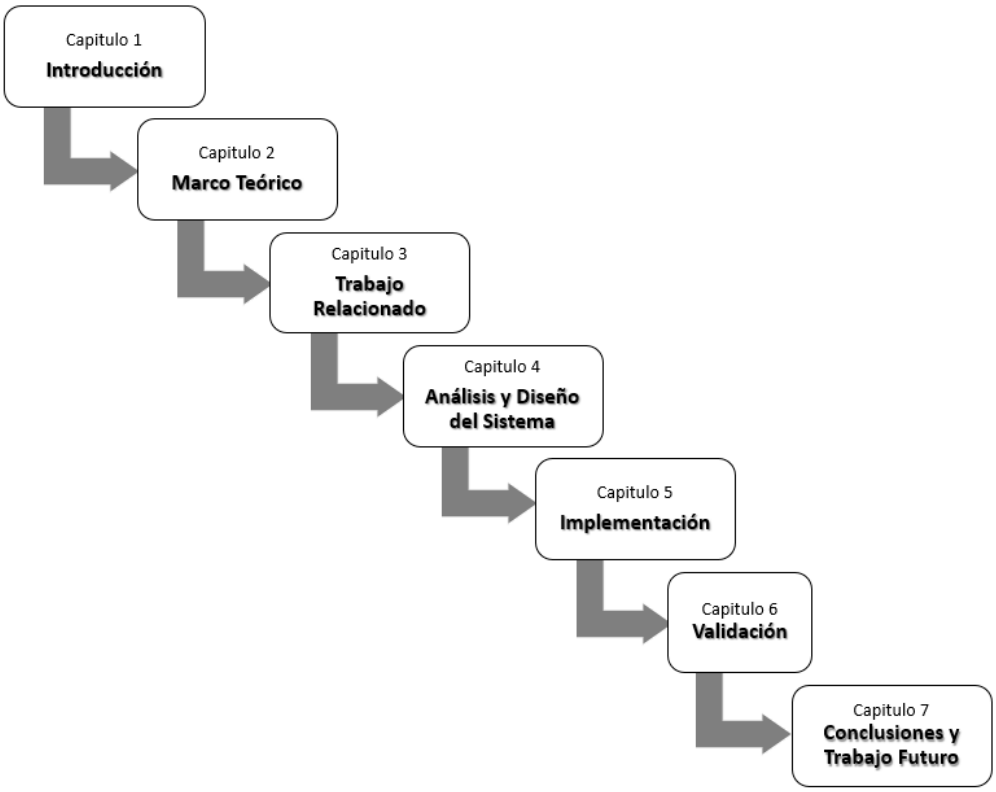

Figura 1.1: Diagrama de la organización del documento.

En el tercer capítulo, «Trabajo Relacionado», se exponen trabajos que sirven de base para el desarrollo de la tesis, tanto en ideas que se pudieron tomar de ellos, como en errores que se evitaron repetir. Para esto, parte del núcleo del capítulo se centra en resumir los trabajos relacionados, exponer sus ventajas y desventajas para, por último, mostrar una tabla comparativa de los trabajos relacionados que se abordaron.

En el cuarto capítulo, «Análisis y Diseño del Sistema», se muestra el análisis y diseño que se realizó para la sentar las bases del nuevo sistema y su posterior desarrollo. En particular, se presenta la arquitectura de la anterior y de la nueva versión del sistema, así como diagramas de diseño de ambas versiones. Adicionalmente, se presentan tablas comparativas del sistema anterior y del nuevo sistema que se desarrolló como parte de esta tesis.

A lo largo del quinto capítulo, «Implementación», se brindan detalles técnicos que se vie-

ron implicados durante el desarrollo del sistema y se exponen características del mismo.

Para el sexto capítulo, «Validación», se detallan las validaciones que se realizaron en el sistema desarrollado. Centrándose en validaciones de funcionalidad para todos los módulos del sistema, dividiendo la parte de validaciones, en el agente de *chatbot* y en los demás módulos del proyecto de *Angular*. Para ello, se explican en qué consisten las validaciones realizadas, los escenarios posibles, el registro y seguimiento de errores, y las medidas para contrarrestar los errores

Por último, en el séptimo capítulo, «Conclusiones y Trabajo Futuro», se recapitula el problema que se prentende solucionar con esta tesis y se da un panorama general de cómo se resolvió y de los resultados que se obtuvieron. Además, se presentan posibles áreas de mejora para con el sistema, mismas que se pueden realizar en trabajos posteriores.

Capítulo 2

Trabajo Relacionado

En el presente capítulo, se exponen diferentes trabajos relacionados a esta propuesta de tesis, los cuales han sido estructurados según su categoría. Primero se describen trabajos relacionados con *chatbots* en entornos educativos (cf. Sección 2.1). Posteriormente se presenta una sección que muestra trabajos relacionados a estudios generales de *chatbots* (cf. Sección 2.2).

2.1 *Chatbots* en educación

Un sector social de suma relevancia es la educación, misma que ha quedado expuesta a necesidades cambiantes, e.g., adoptar nuevas estrategias que faciliten el ingreso a los procesos inmersos en ella. Basogain et al. (2017) mencionan la necesidad de una transición de la educación tradicional a una educación más eficiente, por medio de tecnologías de *e-learning*. La crisis generada por el *COVID-19* ha puesto en evidencia las consecuencias del enfoque tradicional en la educación, e.g., el sector educativo no ha sacado provecho de los avances tecnológicos. Según UNESCO (2020), al 30 de marzo de 2020, 166 países habían cerrado sus escuelas y universidades. Esto significa que, a nivel mundial, el 87 % de la población estudiantil, i.e., 1,520 millones de alumnos, se vio afectado por estas medidas. Además, alrededor del mundo, 63 millones de maestros dejaron de laborar en las aulas.

Brown & Salmi (2020) dan cuenta del panorama, a nivel internacional, sobre las reacciones de ciertas instituciones educativas frente a la transición de la educación en aula a la educación en línea, mencionando algunas carencias en materia de infraestructura y de formación del personal académico para manejar satisfactoriamente una educación en línea. Estas carencias impactan más, y se vuelven más notorias, en panoramas como la actual pandemia de *COVID-19*, que ha forzado a llevar parte de la educación por modalidades a distancia, donde la tecnología es indispensable. Santuario et al. (2020) mencionan que las instituciones y sus

integrantes tienen que desarrollar soluciones innovadoras y eficaces, que mejoren el aprendizaje de sus estudiantes y aprovechen, de mejor forma, los medios digitales y presenciales.

Para hacer frente a situaciones como las descritas arriba, aparecen los *chatbots*, los cuales son sistemas de *software* que permiten una conversación humano-computadora, según la cual el *chatbot* puede entender, responder y resolver algunas necesidades y problemas en un contexto específico, haciendo uso de lenguaje natural. Además, los *chatbots* pueden ser usados en una gran variedad de servicios y algunos de ellos pueden integrarse con diversas plataformas de mensajería.

Basogain et al. (2017) hacen mención del panorama en la educación con base en la replicación del plan de estudios y métodos entre países. PISA (Programa para la Evaluación Internacional de Estudiantes) (OECD, 2016) evalúa la calidad de los sistemas educativos de países miembros de la OCDE (Organización para la Cooperación y el Desarrollo Económico). PISA usa problemas Tipo A (Olabe et al., 2014), que son ideales en la enseñanza estandarizada, al ser deterministas en su solución y en su proceso, sin embargo, no es posible resolver problemas más complejos. De ahí surgen los problemas Tipo B, con procesos mentales de alto nivel, que no son deterministas en su solución ni en sus procesos.

Este estudio se realizó porque no se enseñan temas avanzados a los estudiantes, pues la educación sigue un camino tradicional, alejado de la tecnología. Sin embargo, hay ejemplos de que, gracias al uso de tecnología, los alumnos pueden resolver problemas sin que se den cuenta de la complejidad adyacente. Para este estudio se analizaron trabajos existentes, que implican entornos educativos tradicionales y están orientados a problemas Tipo A. Para luego analizar cómo, con el uso de tecnología, nuevos tipos de problemas y propuestas se pueden solucionar. Además, en este trabajo se da un panorama general de la educación y la necesidad de que evolucione, proponiendo problemas y el uso de primitivas basadas en objetos, que son recursos existentes en la mente humana. Además, se propone aplicar escenarios experimentales durante periodos de tiempo, que sirvan para sentar bases a futuras implementaciones, junto a una transición gradual de la educación con el apoyo de *e-learning*.

Las ventajas son que brinda un panorama de cómo se evalúa e imparte la educación actualmente, además de proponer ideas sobre cómo los sistemas educativos actuales podrían mejorar en el futuro. Mientras las limitaciones radican en que no muestra claramente cómo se pueden evaluar los sistemas educativos futuros y no entra en detalles de las diferentes culturas, formas de pensar y calidad de vida de diferentes países.

Un campus inteligente es aquel en donde las TICs crean un ecosistema de recursos tecnológicos, que satisfacen las necesidades de los miembros. Villegas-Ch et al. (2020) proponen

una arquitectura de campus inteligente compuesta por cuatro capas: (1) de adquisición de datos, (2) de almacenamiento de datos, (3) de análisis de datos y (4) de presentación e interacción con el usuario.

El proyecto se inscribe en un contexto educativo y fue desarrollado con el fin de realizar la integración de las TICs en un campus universitario y analizar cómo se puede crear o implementar un campus inteligente. Se utilizaron dispositivos *IoT* (*Internet of Things*) como sensores y actuadores para recolectar datos, que se almacenaron en bases de datos locales y en la nube; además, se tienen datos de la interacción de alumnos con un sistema *LMS* (*Learning management system*), que incluye un módulo de *chatbot*. Para el análisis de los datos y el aprendizaje del *chatbot*, se usaron *Hadoop* (Lam, 2010) y redes neuronales (Lawrence, 1993).

La arquitectura creada se probó en dos cursos de 24 alumnos cada uno, mostrando que los alumnos tuvieron mayor interés en sus tareas durante los primeros 30 días de actividades. La contribución principal del trabajo radica en describir una arquitectura que se puede aplicar para la creación e implementación de campus inteligentes, detallando las diferentes capas que lo conforman, así como su funcionamiento y las tecnologías implicadas. Con el trabajo, se pudieron detectar problemas comunes para los estudiantes en sus cursos. Se pudo implementar un sistema de recomendación de actividades para los alumnos con la ayuda de un *chatbot*, mismo que es capaz de informar de las actividades de mayor peso a los alumnos y profesores, además de aprender constantemente por retroalimentación.

Las ventajas del trabajo son que muestra tanto los beneficios e implementación de la recomendación de actividades como la aplicabilidad del análisis de datos con *Hadoop*. Las limitaciones son que no muestra aspectos de seguridad, ni algún trabajo a futuro bien definido.

Recientemente han surgido los cursos denominados *MOOC* (*Massive Open Online Course*) (ver Figura 2.1). Bollweg et al. (2019) construyeron un *chatbot* que brinda entrevistas individualizadas para un *MOOC* llamado *ERP Lab Excercises*. Con la ayuda del *chatbot* los estudiantes entrevistan a una empresa real y recaban resultados. El motivo del proyecto es que sólo el 13 % de los estudiantes finaliza cursos de este tipo. Lo que implica una falta de compromiso y falta de individualización de la educación presentada a sus alumnos en un *MOOC*. Para la realización del proyecto, se desarrolló un *chatbot* multi-entidad, basado en el enfoque *CAR* (*Canonical Action Research*) (Davison et al., 2004). El enfoque consiste de cinco etapas: diagnóstico, planificación, intervención, evaluación y reflexión. En el desarrollo del *chatbot*, se uso *DialogFlow* de *Google* y posteriormente el *chatbot* se agregó en un sitio desarrollado en *Wordpress*. Como metodología de desarrollo se uso un enfoque basado en *SCRUM* [1]. El

[1]Marco de trabajo iterativo e incremental para la gestión y desplegado de proyectos principalmente en el desarrollo ágil de *software*

chatbot se probó con un grupo de 28 estudiantes, evaluando su aceptación y rendimiento con el «Inventario de Evaluación de la Capacitación» (Ritzmann et al., 2014) y la «Medida de Rendimiento del *Chatbot*» (Kuligowska, 2015).

La contribución del trabajo radica en poder observar la aceptación y rendimiento de los estudiantes para con el *chatbot*. En partícular el *chatbot* contribuyó con el curso en el que fue implementado (respondiendo más del 50 % de preguntas). Sin embargo, aún la mayoría de estudiantes prefiere entrevistar empresas reales (77.2 %).

Las ventajas del trabajo son que usa tecnologías similares a las propuestas en esta tesis, valida tanto el rendimiento como la aceptación del sistema y usa una metodología de trabajo similar a *Scrum*, lo cual agiliza el proceso de desarrollo en equipos pequeños. Las limitaciones son que no muestra aspectos de seguridad y el desarrollo se limita al contenido de *Wordpress*.

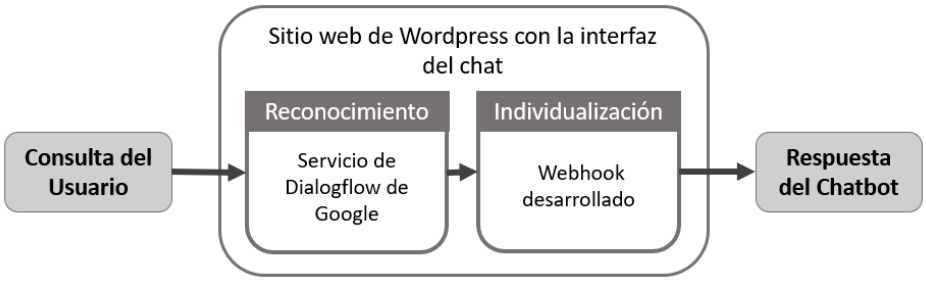

Figura 2.1: Arquitectura de *ERP Lab Excercises*.

En un contexto educativo, se desarrolló un *chatbot* Clarizia et al. (2018), que fue implementado en dos cursos de computación de la Universidad de Salermo. El *chatbot*, tiene una arquitectura compuesta por la interfaz de usuario (para la representación gráfica y entrada/salida de datos de los usuarios), *Back-Office* (maneja la lógica empresarial y el almacenamiento de datos), módulo de base de conocimientos (para la gestión del conocimiento) y un módulo de aprendizaje electrónico (supervisa el diálogo e interacción humano-computadora usando una ontología).

El *chatbot* se implementó para apoyar el proceso de aprendizaje, inferir las intenciones de los alumnos, así como para poder reutilizar materiales que sirvieran a los estudiantes. Aunque no se mencionan a detalle las tecnologías utilizadas, se sabe que se uso el algoritmo *LDA* (*Latent Dirichlet Allocation*) (Blei et al., 2003), que se basa en: procesar la consulta del usuario, eliminar palabras vacías o derivaciones y extraer palabras clave; para crear una ontología que pueda relacionar temas de estudio, usuarios, cursos y lecciones.

Los resultados del estudio, ayudan a reconocer las solicitudes de los estudiantes y observar cuando un *chatbot*, puede dar sugerencias correctas, sugerencias correctas fuera de contexto y sugerencias incorrectas.

Las ventajas del trabajo son que brinda explicación sobre cómo el uso de ontologías puede ayudar en la base de conocimiento y que detalla brevemente, pero de forma concisa, el modelo de ontología usado. Mientras las limitaciones son que no muestra tanto un trabajo a futuro, como detalles técnicos de las tecnologías usadas en el *chatbot* desarrollado.

También en el contexto educativo, se desarrolló un proyecto (Benotti et al., 2014) [2] entre la fundación *Sadosky* y la Universidad Nacional de Córdoba. El motivo de la realización del proyecto fue que existe una baja demanda para ingresar a carreras informáticas, así como bajas tasas de participación femenina y retención, (Carter, 2006) y (Wilson et al., 2010). Se pretende que el uso de *chatbots*, pueda influenciar y motivar a alumnos a estudiar informática (Shaw, 2012).

Para el proyecto, se comparó el impacto de *Chatbot* (propuesta donde los alumnos creaban su propio *chatbot* y aprendían conceptos de informática) y *Alice* (propuesta que permitía a los alumnos crear animaciones y aprender conceptos de informática), en escuelas públicas de Córdoba. Se uso un enfoque basado en cuatro orientaciones: (1) Motivación, crea la necesidad de usar conceptos de informática, (2) Conferencia Breve, introducción a un concepto para resolver el problema, (3) Exploración y Producción, se explora la plataforma y se experimenta para ganar comprensión y (4) Mostrar y Evaluar, los alumnos comparten su progreso. Sin embargo, no se dan detalles técnicos del proyecto, ni cómo son las tecnologías utilizadas.

La contribución del proyecto radica en ver las diferencias de impacto entre las dos propuestas para enseñar informática. Es decir, se pudo observar mayor participación de general hacia *Alice*, pero las mujeres prefieren *Chatbot* (probablemente se debe a que crear animaciones es más popular que saber qué es un *chatbot*), se obtuvieron indicadores de compromiso con el uso de encuestas, se observaron conceptos de la informática que se dificultan a los alumnos. Además, se llegó a la conclusión de que el uso de *chatbots* puede ayudar a motivar a los alumnos a estudiar informática.

Las ventajas encontradas son que el trabajo presentado otorga la capacidad de conectarse con redes sociales como *Facebook*, muestra análisis completos de resultados de las pruebas de campo realizadas y está disponible para descargar y probar desde la Web. Sin embargo,

[2] Link: $bit.ly/1iglAf6$

las limitaciones que se encontraron son que no muestra aspectos de seguridad y no detalla la arquitectura del proyecto.

En Fonte et al. (2016), se presenta *NLAST* (ver Figura 2.2), un sistema que sirve como asistente de estudiantes en su proceso de aprendizaje. El sistema se implementó para brindar apoyo en el contexto educativo. El desarrollo consta de una aplicación en *Android* que funciona como un *chatbot* desarrollado en lenguaje *AIML* que actúa como intermediario entre el alumno y la plataforma servidor. Es un sistema integral del proceso de evaluación de los alumnos. Se utilizan tres repositorios: tarjetas de objetos educativos (recomendación de contenido), preguntas de examen (recursos de exámenes pasados), objetos explicativos (recursos que el profesor subió como apoyo).

La arquitectura del *chatbot* NLAST se compone de una *BUI* (interfaz de usuario del *bot*), un intérprete y una base de datos *AIML*. La (*BUI*) hace uso de texto y voz para la comunicación. La base de conocimiento almacena información del *chatbot*, y permite reaccionar según el estímulo recibido.

El sistema permite observar la importancia en el manejo de repositorios, incluso de material pasado y como éste puede servir para apoyar a los alumnos. Además, el sistema procesa entradas por voz y texto, lo cual supone ventajas sobre otros *chatbots*, por ejemplo los trabajos de Roca et al. (2020), Bollweg et al. (2019) y Li et al. (2017). Sin embargo, no se da prueba de la efectividad del sistema, solamente se presenta el desarrollo.

La ventaja del trabajo es que hace uso eficiente de repositorios que permiten un control y escalabilidad notable. Mientras que las limitaciones encontradas son que no muestra aspectos de seguridad, no da pie a pruebas de campo realizadas que validen el éxito del sistema y no da detalles específicos sobre las herramientas y tecnologías utilizadas.

El artículo de Cunningham-Nelson et al. (2019), resume cuestiones a considerar en el desarrollo de *chatbots* en educación. El estudio se realizó debido a que los grandes grupos de clase presentan mayores problemas (Lee, 2009). En línea los estudiantes esperan una respuesta de apoyo, dan importancia a sus necesidades de comunicación, consideran ser tratados como individuos lo que ayuda al rendimiento académico y satisfacción (Dennen et al., 2007). Frente a ello, los *chatbots* pueden ser una solución.

Entre los ejemplos de *chatbots* creados y mencionados en el artículo están *Siri* de *Apple*, *Cortana* de *Microsoft*, *Bixby* de *Samsung*, *Alexa* de *Amazon*. También, se menciona el uso de *frameworks* que facilitan el desarrollo de *chatbots* y sus características (Techlabs, 2018), entre los *frameworks* mencionados estan: *DialogFlow*, *QnA Maker*, *Wit AI* y *Rasa NLU & Core*.

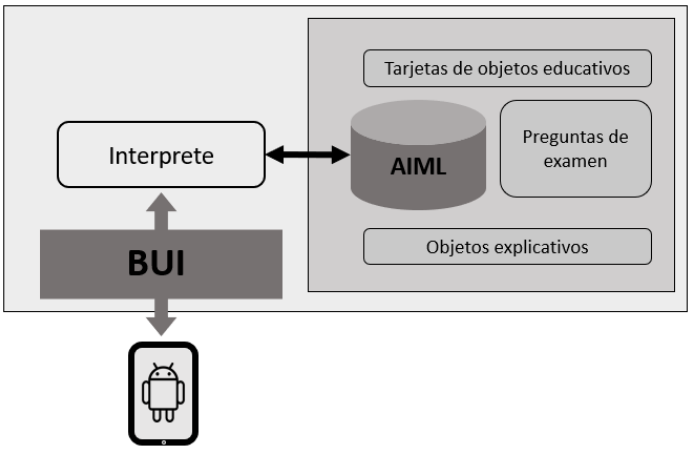

Figura 2.2: Arquitectura de NLAST.

Se hace mención de la importancia del uso de *chatbots* como apoyo a preguntas frecuentes y como prueba de respuesta corta (para confirmar el aprendizaje real de un alumno). El estudio se hizo en base a la observación de trabajos pasados y los resultados de la implementación de *chatbots*, sobre todo en entornos educativos. E. g., el de Brandtzaeg & Folstad (2017).

La contribución del artículo es que brinda conceptos importantes para el manejo de *chatbots*, permite conocer *frameworks* que facilitan su desarrollo y brinda un panorama general de cómo pueden ser aplicados los *chatbots* en la educación.

En esta propuesta encontramos como ventajas el que muestra conceptos, ejemplos y uso de *chatbots* en el contexto educativo, brinda una comparativa entre algunos *frameworks* para *chatbots*, además de ejemplos de los mismos. Sin embargo las limitaciones son que no muestra aspectos de seguridad y no se muestra aspectos que prueben el éxito en la implementación de *chatbots*.

En Verleger & Pembridge (2018), se desarrolló un *chatbot* llamado *EduBot*. Parte de su funcionamiento consiste en: cuando los usuarios preguntan a *EduBot*, éste busca una respuesta en la base de datos de preguntas y respuestas, si no encuentra respuesta, responde que no se ha encontrado tal respuesta y notifica al profesor. El profesor puede responder simultáneamente al estudiante y actualizar la base de datos de preguntas y respuestas con la nueva respuesta. Para combatir respuestas falsas positivas (cuando *EduBot* responde, pero la respuesta realmente no sirve para los estudiantes), se incluye un botón para marcar «Esta respuesta no es útil». *EduBot* usa *IMS-Global LTI* para interactuar con la plataforma de *e-learning Canvas LMS* y *Microsoft QnA Maker* para manejar el procesamiento de IA, otras interacciones se

manejan por una interfaz *PHP* personalizada para el flujo de trabajo de *EduBot*.

EduBot se aplicó a un curso de Introducción a la Computación. Ingresaron 67 estudiantes, la mayoría de los cuales prefirió usar métodos tradicionales. Las respuestas que el sistema recibió fueron clasificadas según la precisión percibida. No se encontraron respuestas en 33.3 % de las ocasiones. Se observó en el estudio que los estudiantes sugieren que *EduBot* estaría mejor respondiendo preguntas específicas, proporcionando ejemplos y orientando a recursos específicos del curso. En trabajo futuro los autores indican que buscarán contar con una base de datos más completa, para poder alentar a los estudiantes a hacer preguntas más significativas.

La ventaja del trabajo es que muestra un método para medir respuestas falsas positivas del *chatbot* y las limitaciones son que no muestra aspectos de seguridad y que no aborda detalles técnicos del desarrollo.

Por último a manera de tabla comparativa (2.1), se resumen los trabajos relacionados a *chatbots* en educación, anteriormente presentados y que fueron de utilidad para el desarrollo de la tesis.

TRABAJO	NIVEL	TECNOLOGÍAS/ ESTANDARES/ LENGUAJES	ARQUITECTURA	ENTRADA	PRUEBAS REALIZADAS	TRABAJO FUTURO	PUNTOS DE INTERES A TOMAR
(Verleger & Pembridge, 2018)	Licenciatura.	*IMS-Global LTI*, *Canvas LMS*, *Microsoft QnA Maker*, *PHP*, *MATLAB*.	-	Texto.	*Focus Group*, pruebas de usabilidad con los estudiantes.	Crear una base de datos más completa para alentar a los estudiantes a hacer preguntas más significativas.	Forma para indicar que "Esta respuesta no es útil". Aspectos a medir en las pruebas relacionadas al interés y retención de los estudiantes.
(Villegas-Ch et al., 2020)	Licenciatura.	Dispositivos *IoT* (sensores, actuadores), *Hadoop*.	En capas: (1) capa de adquisición de datos, (2) capa de almacenamiento de datos, (3) capa de análisis, (4) capa de presentación.	Texto y voz.	Informes de profesores sobre la observación del progreso de los alumnos.	-	Medir tiempo efectivo en trabajo de los estudiantes. Aplicar herramientas de *BI* o *Big Data* para el análisis de los datos. Aplicar una recomendación de actividades.
(Bollweg et al., 2019)	Licenciatura.	*Dialogflow*, *JavaScript*, *WordPress*.	Arquitectura multi-entidad. Usando integraciones de *webhook* con *Dialogflow*.	Texto.	Evaluación de la aceptación del *chatbot*. Medida de Rendimiento del *chatbot* para evaluar el rendimiento.	Mejorar el dialogo del *chatbot* y ampliar el *backend* de la base de datos. Integrar videos e imágenes a las respuestas, así como de reconocimiento de voz.	Algunas pruebas de aceptación y rendimiento pueden ser útiles para aplicar. Analizar implementación de tecnologías usadas como *Dialogflow*.
(Cunningham-Nelson et al., 2019)	Revisión de la aplicación de *chatbots* en contextos educativos, descripciones generales, conceptos, antecedentes, herramientas, *Frameworks* y ejemplos de sus implementaciones.						Permite ampliar el *background* de conocimiento de *chatbots* en contextos educativos. Da comparativas de *Frameworks* para *chatbots*.

TRABAJO	NIVEL	TECNOLOGÍAS/ ESTANDARES/ LENGUAJES	ARQUITECTURA	ENTRADA	PRUEBAS REALIZADAS	TRABAJO FUTURO	PUNTOS DE INTERES A TOMAR
(Clarizia et al., 2018)	Licenciatura.	*Facebook Messenger*.	Por módulos: (1) capa de presentación *front-end* al usuario y su interfaz, (2) *back-office* lógica empresarial y almacenamiento de datos, (3) módulo de base de conocimientos con datos procesados para su gestión, (4) módulo *BOT* de aprendizaje como motor principal que monitorea la interacción hombre máquina. Usa enfoque basado en redes *Petri*.	Texto.	Evaluación por parte de usuarios finales del rendimiento del *chatbot* en dar sugerencias correctas.	-	Modelo de Ontología para la representación de información.
(Benotti et al., 2014)	Secundaria.	-	-	Texto.	Evaluación del *chatbot* por pruebas: previa, intermedia y final. Observación de indicadores de compromiso en los estudiantes.	Realizar más pruebas bajo diferentes enfoques.	Diseño y pruebas realizadas (previa, intermedia y final) que da un buen respaldo y muestra indicadores interesantes sobre el compromiso de los estudiantes.

Cuadro 2.1: Tabla de trabajos relacionados a *chatbots* en educación.

2.2 Estudios generales en *chatbots*

Roca et al. (2020) (ver Figura 2.3), presentó un *chatbot* en el contexto de *eHealth*, que se desarrolló para ayudar a los pacientes con enfermedades crónicas para tener un control en la evolución de su enfermedad. En el artículo, se puede verificar la ayuda que brindan los *chatbots* en el campo de la salud, resumiendo trabajos como el de Chaix et al. (2019). El *chatbot* está basado en tres pilares: escalabilidad (flexible, modular a base de microservicios), modelos de datos estándar (para el almacenaje y transferencia de datos) y modelo conversacional estándar (para procesar las entradas y definir el comportamiento del *chatbot*). Los microservicios del *chatbot*, basan su estructura interna en la comunicación con otros microservicios usando *JSON* y *HTTPS*, la interacción con las bases de datos es através de *HTTPS* y *FHIR*, la interacción con el usuario con *AIML*. Para validar la arquitectura, se estudiaron doce enfermedades crónicas, dando una lista de enfermedades con las preguntas acorde a cada una, con el fin garantizar que el paciente tiene esa enfermedad, si responde dichas preguntas.

La contribución del proyecto es la creación de un prototipo de *chatbot*, que guarda imágenes de los pacientes, para tener un historial y ver su evolución. Además, el artículo permite observar detalles de seguridad en el *chatbot* propuesto, e. g., los datos de los pacientes se cifran, se usan listas de control de acceso y las comunicaciones se protegen con *HTTPS*. También, se observa la aplicabilidad de los *chatbots* en el campo de la salud.

Las ventajas del trabajo son que muestra y justifica ampliamente el por qué utilizar microservicios, muestra explícitamente la tecnología y metodología utilizadas durante el desarrollo del proyecto, además de que brinda consideraciones de seguridad y su implementación durante el desarrollo. Sin embargo, las limitaciones obtenidas son que no muestra trabajo a futuro bien definido y no muestra a detalle pruebas de campo realizadas, que validen mejor el desarrollo realizado.

Encontramos el trabajo de Xiao et al. (2020), que analiza el uso de un *chatbot* con tecnología IA vs una encuesta en línea típica. El estudio se llevo a cabo debido a que las encuestas son un método para recopilar datos, pero si su longitud crece, el tiempo y calidad por pregunta disminuye.

Posteriormente, se construyeron dos encuestas, una en forma típica usando *Qualtrics* y otra por medio de un *chatbot*, usando *Juji*. Los participantes eran jugadores de videojuegos, que fueron reclutados por una empresa de entretenimiento. Se les hicieron preguntas relacionadas con sus datos, su opinión respecto a los avances de videojuegos mostrados, la aceptación a los avances mostrados y retroalimentación. Una mitad fue encuestada con *Qualtrics* y la otra con *Juji*. Para el *chatbot*, al usar *Juji*, se facilitó el uso de diferentes funciones que ya tiene

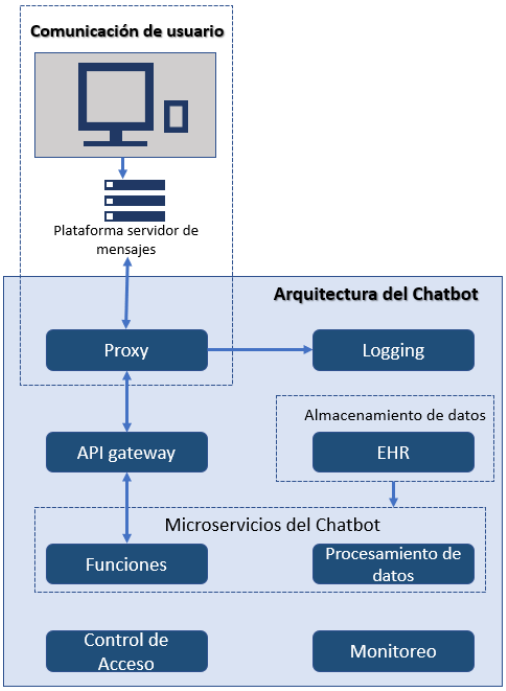

Figura 2.3: Arquitectura basada en microservicios de Roca et al. (2020).

cargadas por omisión, como son reconocer las entradas, transmitir comprensión y emociones, manejar respuestas nulas o preguntas reciprocas e indagar las respuestas. Los resultados se almacenaron en archivos *CSV*. Para la evaluación se siguió un conjunto de métricas basadas en *Gricean Maxims* (Grice, 1975). Se evaluó manualmente cada respuesta para medir su especificidad, relevancia y claridad. Además, de medir el nivel de participación, longitud de respuesta y autorrevelación. El *chatbot* fue capaz de inferir las características de los usuarios, misma capacidad que fue evaluada por los usuarios.

La contribución del trabajo, recae en que compara métodos tradicionales de encuestas *vs* el uso de *chatbots*, observando diferencias y presentando, con base a las pruebas realizadas, porcentajes que permiten ver cómo un *chatbot* puede ser una herramienta muy útil para realizar encuestas (el *chatbot* recopila sólo 30 % más de información). El trabajo también hace mención de riesgos en seguridad que se deben tratar al hacer encuestas con *chatbots* y la posible adicción a los mismos.

Las ventajas del estudio son que compara, usando diferentes métricas, los resultados entre un *chatbot* y un método común para encuestas, da un panorama general de cómo se pueden validar las interacciones usuario - *chatbot* y aborda temas de seguridad o adicción a *chatbots*

sin entrar en detalles técnicos. Por otro lado las limitaciones son que no muestra a detalle la arquitectura con la que se creó el *chatbot* o bien su contraparte de encuestas típicas, no se abordan detalles técnicos y no se sabe con total certeza si el estudio es aplicable a otras líneas de investigación y en otros rangos de edades.

En el contexto general de entrevistas de trabajo, encontramos el estudio de Li et al. (2017), que creó dos entrevistadores virtuales *REP* (*Kaya* de personalidad cálida y alegre y *Albert* reservado y asertivo), para preseleccionar candidatos de empleo a través de una conversación de solo texto. Lo anterior debido a que estudios indican que equipar a los agentes con formas humanas como caras, los hace más agradables y atractivos, (Koda & Maes, 1996) y (Sproull et al., 1996), además del interés de ver cómo afecta la personalidad de un agente en el comportamiento de los usuarios.

El *REP*, realiza entrevistas por medio de una interfaz de chat basada en texto. El proyecto cuenta con un analizador para observar la interacción de un usuario e inferir sus características. También, se cuenta con un motor de conversación que decide qué temas y cuándo activarlos en una conversación, en base a patrones y la conversación misma. Para los dos tipos de entrevistadores, se crearon dos imágenes estáticas de perfil. La personalidad del entrevistador virtual se basa en el tipo de cortesía y de lenguaje. Se usa una arquitectura cliente-servidor basada en Web. Los componentes del lado del servidor están escritos en *Clojure* y del cliente en *HTML* y *JavaScript*. Se mide la disposición de compartir información de los usuarios con la escala de gestión de impresiones y tres acciones de intercambio de información, comparando las debilidades inferidas por el sistema contra las que piensa tener el usuario mismo.

La contribución del trabajo permite verificar cómo la personalidad de un agente influye en las respuestas de los usuarios, e. g., el estudio arrojó que los usuarios emocionales, preocupados y vulnerables confían menos en *Kaya*, contrario a los más agradables y confiados. Se obtuvieron porcentajes de la participación de los usuarios y se plantea a futuro verificar cómo se pueden aplicar diferentes personalidades de agentes en otros contextos.

Las ventajas observadas del trabajo son que explica bien cómo la personalidad de un agente influye en la confiabilidad y respuesta del usuario al sistema, muestra aspectos básicos, pero completos del diseño de la personalidad de *chatbots* y realiza varias pruebas para validar el sistema. Las limitaciones obtenidas son que no muestra aspectos de seguridad del *chatbot* y que no se sabe cómo el enfoque de personalidad pueda afectar en otros campos.

A continuación, a manera de tabla comparativa (2.2), se resumen los trabajos anteriormente presentados, los cuales estan relacionados a estudios generales en *chatbots*.

TRABAJO	ENFOQUE	TECNOLOGÍAS/ ESTANDARES/ LENGUAJES	ARQUITECTURA	ENTRADA	PRUEBAS REALIZADAS	TRABAJO FUTURO	PUNTOS DE INTERES A TOMAR
(Roca et al., 2020)	Salud: *chatbot* para seguimiento de pacientes con enfermedades crónicas.	*AIML* 2.0, *HL7 FHIR*, *Java* 8 *Update* 121, *Python* 3.7, *Docker*.	Basado en microservicios, con dos de ellos centrales: "*proxy*" que traduce el mensaje recibido del usuario a la arquitectura y "puerta de enlace *API*" que decide que microservicio del *chatbot* es el adecuado para procesar la solicitud del usuario.	Texto.	Validando únicamente el funcionamiento de la aplicación.	Dotar con reconocimiento de voz y agregar servicios de asesoramiento basados en el procesamiento de imágenes.	Lógica de microservicios para escalabilidad.
(Xiao et al., 2020)	General: Estudio comparativo del uso de *chatbots*.	*Qualtrics*, *Juji*.	-	Texto.	Métricas basadas en *Gricean Maxims*, midiendo calidad de respuestas de acuerdo a su especificidad, relevancia y claridad.	Probar el estudio en diferentes rangos de edades y líneas de investigación.	Observar cómo encuestas tradicionales pueden ser reemplazadas por *chatbots*.
(Li et al., 2017)	General: Estudio sobre la personalidad de *chatbots*.	*Clojure*, *HTML*, *JavaScript*.	Cliente servidor basada en web.	Texto.	Escala de gestión de impresiones, y la disposición de compartir información.	Aplicar la personalidad de agentes en otros contextos.	Considerar la personalidad de los agentes y sus tipos de respuestas para motivar a los usuarios.

Cuadro 2.2: Tabla de trabajos relacionados a estudios generales en *chatbots*.

Capítulo 3

Análisis y diseño del sistema

En el capítulo actual, se aborda el análisis y diseño del sistema. Se comienza analizando la situación del sistema en su primera versión Mendoza et al. (2020), el cual sirve de base para el proyecto, detallando sus características, arquitectura y funcionalidades (cf. Sección 3.1). Seguido, se analizan las áreas de oportunidad y errores que se detectaron en el análisis del sistema en su primera versión (cf. Sección 3.2). Por último, se analizan y diseñan soluciones que cubren las deficiencias y errores descubiertos en la versión anterior, estableciendo así la pauta para el desarrollo de la segunda versión, que es la que se desarrolla en esta tesis (cf. Sección 3.3).

3.1 Análisis de la primera versión del sistema

El sistema en su primera versión (Mendoza et al., 2020), mismo que sirve como base para la segunda versión que da sustento a la tesis, cuenta con tres proyectos (ver Figura 3.1):

- API-CHATBOT-BASEDEDATOS: emplea en su tecnología *MySQL* y *AdonisJS*, dando pie a un *ORM* [1] que permite mapear las tablas de la base de datos relacional, a objetos para su posterior manipulación en el sistema.

- CINVESTAVCHATBOT: utiliza *Angular* como su tecnología principal. Contiene el código del sistema que se muestra al usuario final, desde todas las interfaces del sistema,

[1] *ORM*: permite el mapeo de una estructura de base de datos relacional a una estructura orientada a objetos, generando una base de datos virtual orientada a objetos de forma, que las acciones CRUD (Crear, leer, modificar y borrar) se realicen por medio de la base de datos virtual de objetos y afecten a la base de datos relacional.

hasta operaciones de gestión y control. También, cuenta con conexión a *Adonis* para obtener los objetos mapeados por el proyecto de API-CHATBOT-BASEDEDATOS, además, tiene la conexión tanto a la base de datos *NoSQL* de *Firebase*, como al agente de *chatbot* utilizado para el proyecto y que se encuentra alojado en *Dialogflow*.

- FIREBASEFUNCTIONS: define dentro de una función el comportamiento individual de cada *intent*, una vez que ha sido elegido por *Dialogflow*, según la expresión del usuario capturada mediante el *chatbot*. Además dentro con dicha función, se establecen las operaciones correspondientes con la base de datos de *Firebase*, siendo indirectamente un puente entre *Dialogflow* y *Firebase*.

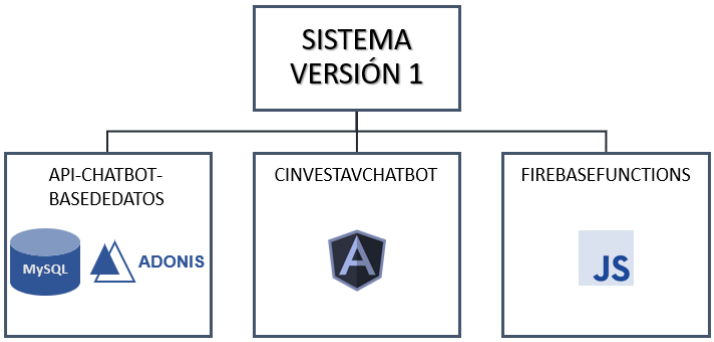

Figura 3.1: Proyectos que conforman el sistema de Mendoza et al. (2020), que sirve como base al proyecto de tesis.

Teniendo en mente los proyectos que conforman la primera versión del sistema, se verifico a detalle el código fuente y se obtuvo que las herramientas implicadas son:

- *Angular*: *framework* principal para el desarrollo del sistema, desde la programación de sus interfaces, el control y gestión de operaciones y datos.
- *MySQL*: sistema gestor de base de datos que sirve para mantener y gestionar la base de datos relacional del sistema base.
- *AdonisJS*: permite mantener un *ORM* que mapea los datos y estructura de la base de datos relacional a objetos virtuales, para su uso en el sistema creado en *Angular*.
- *Firebase*: base de datos *NoSQL*, que almacena información relevante para el *chatbot*, ya que es la base de conocimientos.

- *Dialogflow*: servicio de *Google*, que contiene el agente de *chatbot* usado para el sistema.

Para los servicios de *Google* (*Dialogflow* y *Firebase*), se proporcionó acceso directo a la liga en donde se encuentran ubicados los servicios originales, con el fin de observar la estructura de la base de datos y del agente de *chatbot*. Al indagar entre carpetas y estructuras, se obtuvieron los diagramas del sistema, las bases de datos y el agente.

3.1.1 Panorama general previo

El sistema base, plantea tres roles de usuario:

- Alumno: toma clases de distintas materias y está inscrito en determinado grado y grupo.
- Profesor: imparte clases en diferentes materias de diferentes grados y grupos.
- Administrativo: administra usuarios del sistema, vincula materias con grados y grupos, administra profesores y alumnos.

El sistema base toma algunas reglas y características del sistema de educación a nivel secundaria en México (Federal, 2018), además de ciertas reglas que se definieron para su diseño y desarrollo. Algunas son:

- Los alumnos sólo pueden estar inscritos en un sólo grado y grupo.
- Los alumnos sólo pueden tomar materias que les corresponden a su grado y grupo.
- Sólo se tienen tres grados escolares (primero, segundo y tercero).
- Se definieron hasta diez grupos (A, B, C, D, E, F, G, H, I, J).
- Un profesor puede subir material a determinada clase.
- Una clase que se imparte para determinado grado, automáticamente se imparte en todos los grupos de ese grado.
- Un profesor asignado a una clase, se les asigna a todos los grupos del grado de esa la clase.
- Un material extra-clase subido por el profesor, contiene nombre, descripción y un link de apoyo.
- Un administrativo, puede activar y desactivar usuarios de tipo profesor y alumno.
- Un administrativo, inscribe a un grado y grupo ha determinado alumno.

- Un administrativo, es capaz de dar de alta nuevas materias.
- Al registrarse, cualquier usuario (a excepción del administrativo inicial) tiene su cuenta inactiva.
- Un usuario inactivo puede entrar al sistema, pero no puede realizar acción alguna hasta que su cuenta sea activada.
- Un profesor puede registrar exámenes y tareas para determinadas materias de ciertos grados y grupos.

3.1.2 Proyecto CINVESTAVCHATBOT

Proyecto donde *Angular* toma relevancia. El proyecto (ver Figura 3.2) utiliza el patrón Modelo Vista Controlador, por lo que tiene separados los datos, su gestión y su visualización. *Angular* hace uso de componentes con el fin de descomponer una aplicación compleja en partes más sencillas, otorgando mayor escalabilidad. El proyecto se divide en Componentes, que definen la interfaz gráfica y operaciones de cada módulo que conforma el sistema. En Modelos que sirven para la definición de clases de los diferentes tipos de objetos que se usan en el sistema. Y de Servicios que definen las diferentes operaciones y sus llamados a los servicios externos y del Ambiente (*Environment*) para establecer la conexión a servicios externos y la creación de variables globales de estos servicios. Se encontró que el proyecto originalmente cuenta con los siguientes componentes:

- *Chatbot*: contiene el módulo del *chatbot*, al cual pueden acceder profesores y alumnos, para interactuar con el agente de *chatbot* previamente programado y pueden preguntar diferentes cuestiones.
- *Header*: para controlar acciones y la interfaz de la barra del menú superior del sistema.
- *Lessons*: permite agregar nuevas materias a determinados grados y grupos.
- *Login*: permite que los usuarios accedan al sistema o vayan a la opción de registrarse.
- *Menu*: controla y permite la visualización de diferentes herramientas según el rol del usuario actual en el sistema.
- *Register*: para registrar nuevos usuarios.
- *User-lessons*: permite inscribir alumnos a cierto grado y grupo.

- *Validate-users*: para gestionar usuarios del sistema.

Modelos:

- *File*: para crear objetos de tipo archivo, que sirven al profesor para añadir material de apoyo.
- *Lessons*: permite crear objetos de tipo lecciones, para vincular alumnos con grados y grupos.
- *Subject*: permite crear objetos de tipo asignatura, para vincular materias con algún grado.
- *User*: para crear objetos de tipo usuario, que sirven para administrar usuarios en el sistema.

Servicios:

- *Dialogflow*: para realizar llamadas a operaciones alojadas en servicio de *Dialogflow*.
- *Firebase*: permite realizar llamadas y operaciones al servicio de *Firebase*.
- *Api*: permite tomar los objetos mapeados de los datos de la base de datos de *MySQL*.

Environment:

- *Dialogflow*: establece la conexión con el servicio de *Dialogflow* y crea una variable global que permite las llamadas al servicio de *Dialogflow* desde cualquier parte del código.
- *Firebase*: Establece la conexión con el servicio de *Firebase* y crea una variable global que permite las llamadas al servicio de *Firebase* desde cualquier parte del código.

Se dieron por obvias, al menos para el documento de tesis, algunas partes del proyecto como las que involucran ventanas emergentes, rutas y controladores.

3.1.3 Proyecto API-CHATBOT-BASEDEDATOS

Se usa *ORM* para manipular datos de la base de datos relacional y asi poder usarlos en CINVESTAVCHATBOT. Hace uso de controladores que permiten definir las operaciones sobre los distintos tipos de objetos mapeados de la base de datos, cada clase en los modelos está vinculada con una tabla en la base de datos. Mientras las rutas vinculan por métodos *POST, GET, DELETE, PUT*, a los diferentes controladores definidos. La arquitectura que se pudo definir es parecida a la mostrada en la Figura 3.3:

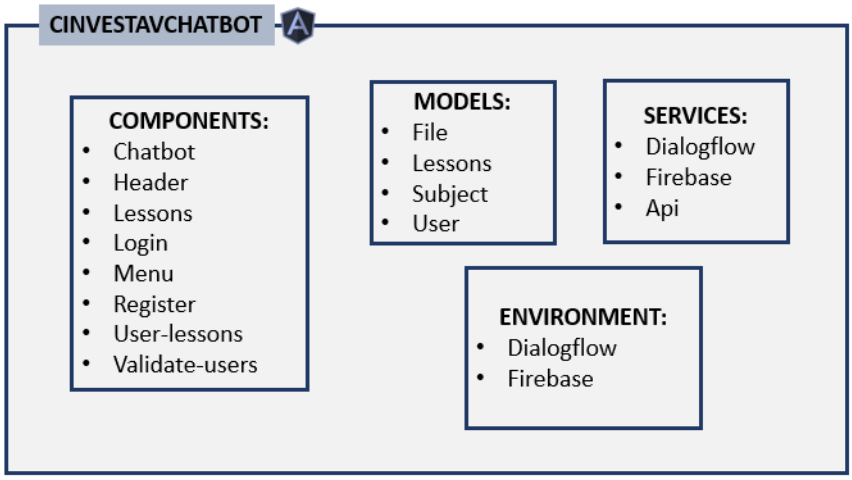

Figura 3.2: Proyecto CINVESTAVCHATBOT original.

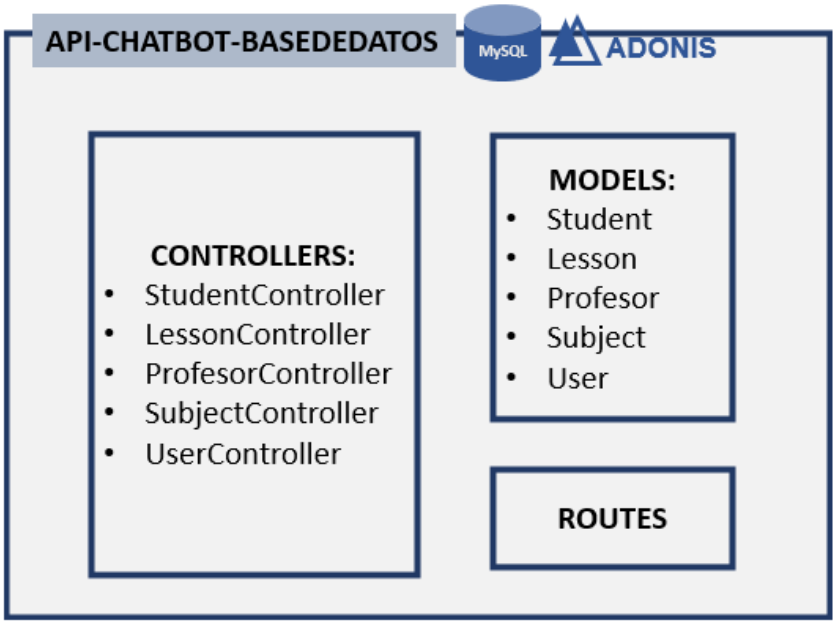

Figura 3.3: Proyecto API-CHATBOT-BASEDEDATOS original.

3.1.4 Base de datos relacional

La base de datos relacional del proyecto, sirve para almacenar todo tipo de datos que se genera del sistema, con excepción de datos vinculados al manejo del *chatbot* de *Dialogflow* como exámenes, tareas y material extra-clase. Para su mejor compresión, se creo un diagrama

entidad relación, ver Figura 3.4, obteniendo tres tablas:

- *Users*: contiene a los diferentes usuarios registrados en el sistema, junto a sus datos personales, su rol e información de autentificación.
- *Lessons*: para vincular un usuario a determinado grado y grupo.
- *Subjects*: para almacenar las materias del sistema y vincularlas a cierto grado.

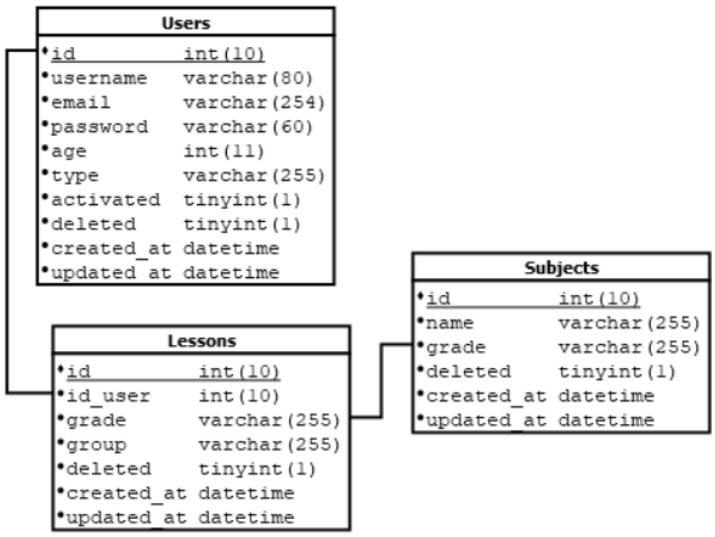

Figura 3.4: Diagrama entidad relación del sistema base.

3.1.5 Agente de *Dialogflow*

El proyecto FIREBASEFUNCTIONS, permite establecer el vínculo entre un *intent* con su correspondiente función, dentro de la cual se definen las operaciones de llamadas a la base de datos *NoSQL* de *firebase* y las salidas/respuestas para el usuario final por medio del *chatbot*. El agente de *chatbot* alojado en *Dialogflow* contiene los siguientes elementos (ver Figura 3.5):

Intents:

- Añadir-examen: permite agregar nuevos exámenes de determinada materia a cierto grado y grupo.
- Consultar-examen: permite la consulta de los exámenes registrados.

- Añadir-material-clase: permite añadir una tarea para determinada materia a cierto grado y grupo.
- Consultar-tarea: permite consultar las tareas registradas.
- Consultar-material-clase: permite consultar los materiales que han sido añadidos en determinada materia en cierto grado y grupo.

Contextos:
De momento, el agente no cuenta con algún contexto creado.

Entidades:

- Grado: para los diferentes grados y sus variaciones que pueden existir.
- Clases: para las diferentes materias y sus nombres que pueden darse.
- Grupo: para los distintos grupos y sus variaciones que pueden existir.

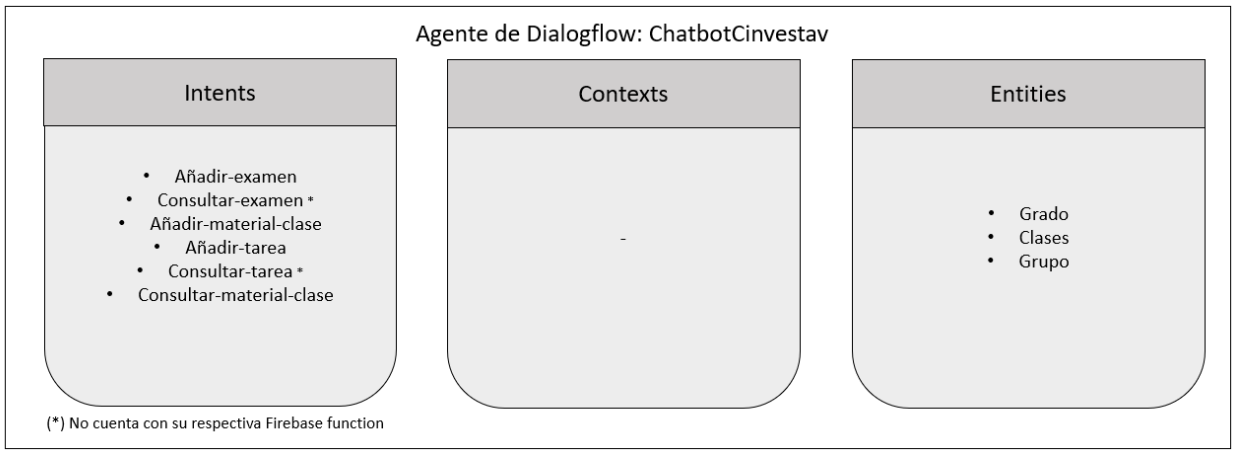

Figura 3.5: Partes del agente de *chatbot* para el sistema base.

3.1.6 Base de datos *NoSQL*

La estructura de la base de datos que se aloja en *Firebase*, véase Figura 3.6, tiene la estructura de un árbol *JSON*, dividido en:

- Exámenes: almacena los exámenes registrados por los profesores.

- Tareas: almacena las tareas registradas por los profesores.

- Material Extra-Clase: contiene el material extra-clase que han subido los profesores.

Cada objeto tiene datos del día y fecha de subida, además del grado, grupo y clase a los que pertenece. Las tareas tienen un tema el cual se refiere a su título. El material extra-clase tiene una descripción del material y una *url* para almacenar algún link a materiales externos.

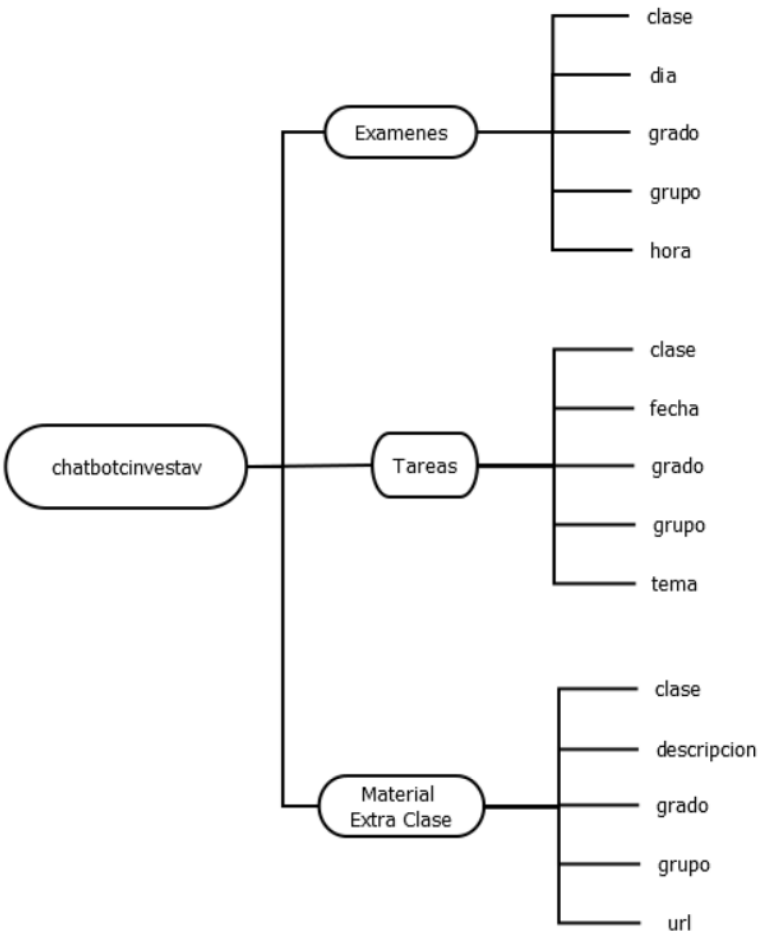

Figura 3.6: Base de datos *NoSQL* de *Firebase* usada para el sistema base.

3.1.7 Arquitectura del sistema base

El proyecto API-CHATBOT-BASEDEDATOS (ver Figura 3.7), sirve como puente entre los diferentes componentes definidos en CINVESTAVCHATBOT y la base de datos relacional de *MySQL* del sistema. Esta base de datos contiene usuarios, materias y grupos. API-CHATBOT-BASEDEDATOS sirve como un *ORM* que realiza el mapeo entre entidades relacionales sobre objetos.

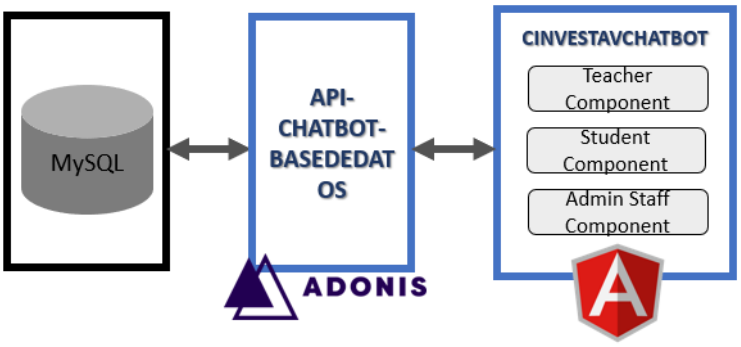

Figura 3.7: Proyecto API-CHATBOT-BASEDEDATOS.

El proyecto CinvestavCHATBOT (ver Figura 3.8), contiene los componentes en *Angular* definidos para cada rol, y la conexión entre los servicios de *Google* con el sistema (*Dialogflow v1*, y la base de datos *NoSQL* que nutre la de conocimiento del *chatbot*, alojada en *Firebase*). La relación entre el *chatbot* y su base de datos de *Firebase*, se realiza por medio de *Firebase Functions*.

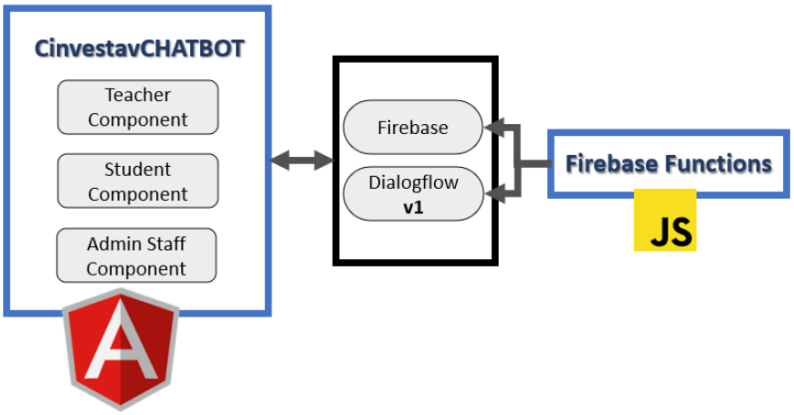

Figura 3.8: Proyecto CinvestavCHATBOT.

Finalmente la arquitectura del CINVESTAVCHATBOT se presenta en la Figura 3.9.

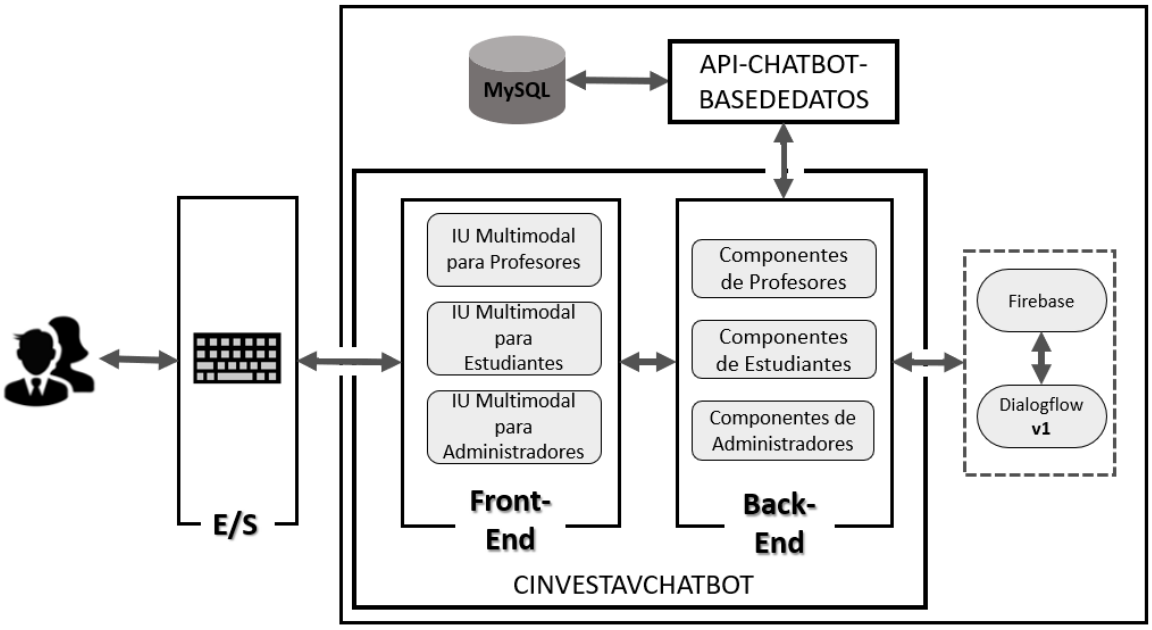

Figura 3.9: Arquitectura del sistema de Mendoza et al. (2020), que sirve como base al proyecto de tesis.

3.2 Problemáticas de la primera versión del sistema

Una vez analizado el sistema en su primera versión, se procedió a validarlo, por lo que el sistema se probó su funcionalidad por medio de casos de uso por rol del sistema, de lo cual se obtuvierón las siguientes tablas:

Sistema V1	
Rol: Administrador	
Módulo	***Detalles***
Administración de los usuarios.	• Permite la visualización y búsqueda de usuarios registrados en el sistema. • Permite cambiar de rol a un usuario. • Permite desactivar y activar cuentas de usuarios.

Administración de materias-grados-grupos-alumnos.	• Permite visualizar y buscar materias registradas en cierto grado del sistema. • Da la capacidad de activar o desactivar una materia del sistema. • Da la capacidad de registrar una nueva materia para un grado. • Da la posibilidad de visualizar posibles alumnos a agregar en cierto grado y grupo.

Cuadro 3.1: Tabla de características del usuario administrador en la primera versión del sistema.

Sistema V1	
Rol: Profesor	
Módulo	**Detalles**
Chatbot.	• Permite interactuar con el agente de Chatbot mediante la entrada y salida de texto. • Permite añadir materiales extra-clase a determinada materia de cierto grado y grupo. • Da la posibilidad de adjuntar un link al material extra-clase. • Permite añadir exámenes para determinadas materias, grados y grupos. • Permite añadir tareas para determinadas materias, grados y grupos. • Permite consultar material extra-clase.

Cuadro 3.2: Tabla de características del usuario profesor en la primera versión del sistema.

Sistema V1	
Rol: Alumno	
Módulo	**Detalles**
Chatbot.	• Permite una interacción con el agente de Chatbot mediante la entrada y salida de texto. • Capacidad de consultar los materiales extra-clase que ha subido un profesor para cierta materia, grado y grupo.

Cuadro 3.3: Tabla de características del usuario alumno en la primera versión del sistema.

Observadas las problemáticas presentadas en el sistema base, se generó una tabla para llevar su registro. La columna “Problema” es el problema detectado, “Descripción” menciona de qué trata o en dónde radica la problemática y “Severidad” se refiere a la medida en que la problemática puede afectar al sistema, es decir, el grado en el que puede llegar a afectar la

funcionalidad del sistema.

Problemáticas en el Sistema V1		
Problemática	**Descripción**	**Severidad**
Falta de validación al agregar materias.	Se pueden añadir materias repetidas para cierto grado.	Media.
No se permite el registro de un alumno a determinado grado y grupo.	No se registran alumnos en determinados grados y grupos, aún cuando todos los parámetros anteriores han sido fijados en el sistema.	Crítica.
Falta de validaciones al añadir alumnos.	Falta de validación para verificar que sólo se muestren usuarios con rol de alumnos al momento de querer añadir un alumno a determinado grado y grupo.	Media.
Pobre interacción entre un usuario y el *chatbot*.	No se puede mantener una conversación fluida entre el agente de *chatbot* y un usuario, además de que el agente rara vez entiende las entradas del usuario.	Alta.
La solución no retroalimenta al usuario cuando éste interactúa con el sistema.	Al realizar una tarea no se muestra algún mensaje de que se ha efectuado o no la tarea.	Media.
Duplicación innecesaria de datos entre las bases de datos.	La base de datos relacional contiene todos los datos que tiene la base de datos no relacional, aún cuando la primera se orienta más a la autentificación de usuarios, mientras la segunda se orienta a reforzar el conocimiento del agente de *chatbot* y de las diversas herramientas del sistema.	Crítica.
Uso de versión antigua de *Dialogflow*.	El *chatbot* está diseñado con la versión 1 de *Dialogflow*, donde su conexión con el sistema se da con el uso de un *token*, el cual ha quedado en desuso en la versión 2, por lo que no se pueden efectuar actualizaciones ni mejoras, hasta no migrar a la segunda versión y usar el nuevo método de conexión que se ofrece.	Alta.

Problemáticas en el Sistema V1		
La interfaz del sistema no es responsiva.	La interfaz del sistema no es responsiva ante los diferentes tamaños de pantalla que se pudiésen presentar. Las imágenes que se utilizan en la interfaz del sistema son de terceros, además, sobresaturan en algunos casos la interfaz.	Alta.
Errores de ortografía en interfaces del sistema.	Parte del texto mostrado cuenta con errores ortográficos.	Baja.
Violación de seguridad para usuarios inactivos y nuevos.	Al momento de registrar un nuevo usuario en el sistema, se le dirige automáticamente a su menú de inicio, sin embargo, las opciones que se le presentan se encuentran habilitadas, lo cual rompe con el punto de que un usuario se encuentra desactivado al momento de su creación, hasta que un usuario administrador lo active y con ello pueda usar las herramientas disponibles para su rol en el sistema.	Crítica.
Falta de validación al registrar un nuevo usuario en el sistema.	En la ventana de registro de usuarios, se permite el registro de correos inválidos, lo cual da cabida a posibles inyecciones *SQL*.	Alta.
Opciones y herramientas limitadas para los usuarios.	Permite una interacción con el agente de *chatbot* mediante la entrada y salida de texto. Las herramientas que se encuentran disponibles en el sistema se encuentran limitadas. Ejemplos son: • Un material de apoyo sólo puede incluir un único link de apoyo. • Un profesor no puede consultar los exámenes que ha programado previamente. • Un alumno no puede consultar los exámenes que ha programado un profesor.	Alta.

Problemáticas en el Sistema V1		
Ausencia de herramientas en el sistema.	Hay aspectos básicos que se encuentran presentes en la escuela, sin embargo, no han sido desarrollados para el sistema, ejemplos son: • Gestión de tareas. • Gestión de calificaciones. • Control del progreso de los estudiantes. • Gestión de avisos.	Crítica.

Cuadro 3.4: Tabla de problemáticas detectadas en la primera versión del sistema.

Para solventar las problemáticas, se hizo necesaria la construcción de una nueva versión del sistema, versión que da sustento a la presente tesis.

3.3 Segunda versión del sistema

La segunda versión del sistema que es el producto central de la actual tesis, surgió como respuesta ante las problemáticas detectadas de la primera versión y la necesidad de actualizarlo. Para el análisis y diseño de esta nueva versión, se estudiaron distintas soluciones y propuestas a cada problemática/deficiencia detectada en el análisis de la primera versión del sistema, las soluciones fueron concentradas en la tabla 3.5.

Soluciones Propuestas a Problemáticas detectadas en el Sistema V1		
Problemática	**Causa**	**Solución Propuesta**
Falta de validación al agregar materias.	No se realizan las validaciones que aseguren que no se repiten datos.	Consultar primero si ya hay una materia registrada en el grado y grupo seleccionado, con el mismo nombre de la materia que se está insertando, devolviendo una bandera de *true* o *false* para la inserción o no de la nueva materia a la base de datos.

Soluciones Propuestas a Problemáticas detectadas en el Sistema V1		
No se permite el registro de un alumno a determinado grado y grupo.	Error en la función para añadir alumnos alojada en el proyecto de API-CHATBOT-BASEDE DATOS.	Volver a reescribir la función para añadir alumnos del proyecto API-CHATBOT-BASEDEDATOS, debido a que su modificación tomaría más tiempo.
Falta de validaciones al añadir alumnos.	Falta de validación para verificar que el rol de los usuarios es el de alumnos.	Añadir la verificación en la consulta de usuarios, para que sólo se muestren los que tienen el rol de alumnos, para su posible inscripción a tomar clases en cierto grado y grupo.
Pobre interacción entre un usuario y el *chatbot*.	Falta de entrenamiento, poca cantidad de *intents*, poca cantidad de entidades y nulo uso de contextos.	Añadir mayor cantidad de *intents*, entidades y posiblemente usar contextos. Agregar más frases de entrenamientos en los *intents* diseñados.
La solución no retroalimenta al usuario cuando éste interactúa con el sistema.	No se programó la retroalimentación del sistema.	Agregar ventanas emergentes que contengan la retroalimentación, con pocas palabras, resultado de las acciones de los usuarios en el sistema.

Soluciones Propuestas a Problemáticas detectadas en el Sistema V1		
Duplicación innecesaria de datos entre las bases de datos.	Los datos que se generan, se almacenan a la par en ambas bases de datos. Además, en la base de datos *NoSQL*, no se hace uso ni se saca provecho de la estructura de árbol *JSON*.	Modificar la estructura de la base de datos relacional, haciendo que sólo se almacene información del registro y autentificación de los usuarios. Modificar la estructura de la base de datos *NoSQL*, eliminando en lo posible duplicación de datos, haciendo uso de la estructura de árbol *JSON*.
Uso de versión antigua de *Dialogflow*.	El *chatbot* está diseñado con la versión 1 de *Dialogflow*, el cual usa un *token* de acceso para la integración de *Dialogflow* en plataformas webs de terceros.	Utilizar la nueva versión de *Dialogflow* (V2). Migrando parte de la estructura inicial en la V1 a la V2 y buscando un nuevo método de conexión al servicio de *Dialogflow*, desde webs externas.
La interfaz del sistema no es responsiva.	La interfaz del sistema, se programo en sus *CSS*, usando medidas en pixeles.	Considerar los diferentes tipos de tamaño de pantalla posibles, utilizando en los *CSS* del proyecto medidas porcentuales en lugar de pixeles.

Soluciones Propuestas a Problemáticas detectadas en el Sistema V1		
Errores de ortografía en interfaces del sistema.	No se verificó que el texto en pantalla careciera de faltas de ortografía.	Verificar los errores de ortografía en las vistas del sistema y corregir los errores detectados.
Violación de seguridad para usuarios inactivos y nuevos.	Cuando un usuario está dentro del sistema, una vez pasando el *login*, no se verifica su estatus.	Verificar el estatus de un usuario al acceder a cada componente, deshabilitando con el uso de banderas, las funcionalidades en caso de que el estatus sea inactivo.
Falta de validación al registrar un nuevo usuario en el sistema.	No se realizo ninguna verificación en el campo de correo, al registrar nuevos usuarios.	Verificar que para el campo de correo, al registrar un nuevo usuario, el texto insertado tenga la estructura de un correo electrónico.
Opciones y herramientas limitadas para los usuarios.	No se programaron o están limitadas algunas funcionalidades por usuarios del sistema.	Añadir funcionalidades a las herramientas ya programadas en el sistema, como: añadir funciones para que el profesor y alumno consulten exámenes, tareas, avisos, etc.

Soluciones Propuestas a Problemáticas detectadas en el Sistema V1		
Ausencia de herramientas en el sistema.	No se programaron o consideró añadir, algunas herramientas por usuarios del sistema.	Programar nuevas herramientas por rol en el sistema.

Cuadro 3.5: soluciones propuestas ante las problemáticas detectadas en la primera versión del sistema.

3.3.1 Nueva versión de la base de datos relacional

La nueva base de datos relacional (ver Figura 3.10), busca resolver el problema del almacenaje innecesario de datos, pensando en que la base de datos sólo tenga datos personales de los usuarios, para su registro y posterior autenticación, sólo fue necesario mantener una única tabla de usuarios. La tabla tiene la misma estructura que su homóloga en la versión base del sistema.

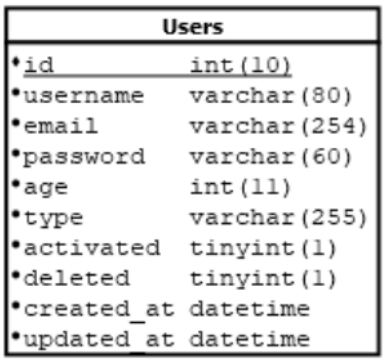

Figura 3.10: Estructura de la Base de datos relacional de la nueva versión del sistema.

3.3.2 Nueva versión de la base de datos *NoSQL*

Para la base de datos *NoSQL* alojada en *Firebase* se utilizan, dadas las posibles soluciones a problemáticas y carencias del sistema base, dos servicios de *Firebase*:

- *Realtime Database*: para alojar los datos simples del sistema, como datos manejados en

texto plano, e.g., avisos, información de exámenes, tareas, recordatorios, dudas, material extra clase, calificaciones, etc.

- *Cloud Storage*: para almacenar datos en archivos. Algunos de los datos son archivos en diferentes formatos, que pueden contener tareas, mensajes, etc.

La base de datos de *Realtime*, se diseñó aprovechando la estructura de árbol *JSON*, dividida en niveles. El primer nivel (Figura 3.11) divide la estructura en tres grupos: `avisosGenerales` (para contener los avisos generales creados por los administrativos), `clases` (para todo lo relacionado a las clases, grados y grupos del sistema) y `usuarios` (para almacenar los usuarios registrados en el sistema).

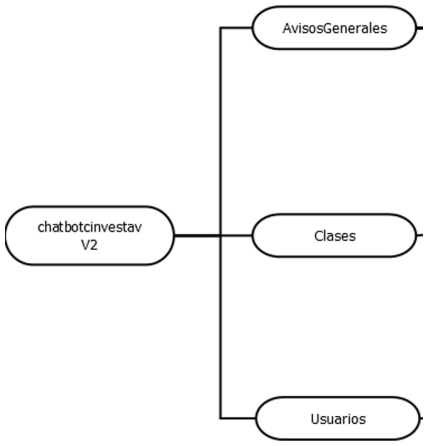

Figura 3.11: Primer nivel en la estructura de árbol de la base *NoSQL* de la segunda versión del sistema.

Para los `avisosGenerales` (Figura 3.12), se añadieron parámetros como `contenido` (texto contenido en el aviso), `fechaExp` (fecha en que expira el aviso), `horaExp` (hora en que expira el aviso), `fechaPub` (fecha de publicación del aviso), `profesores` (indica con un 1 si el aviso es visible a los profesores y con un 0 si no), `estatus` (indica el estatus de un aviso: 0 inactivo y 1 activo), `titulo` (título del aviso), `Destinatarios` (grados y grupos que pueden ver el aviso).

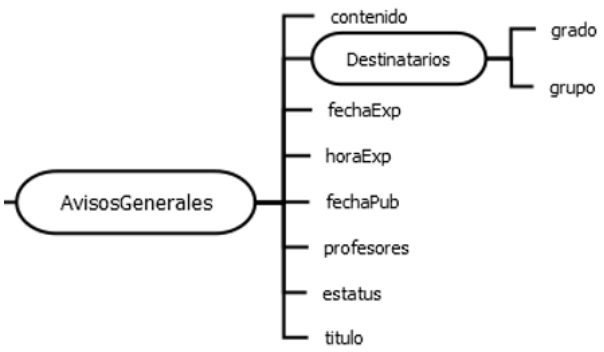

Figura 3.12: Avisos en la base de datos *NoSQL* de la segunda versión del sistema.

Para las `clases` (Figura 3.13), se busca evitar la duplicación de datos, agilizar consultas y operaciones sobre la base de datos. Por ello, desde la estructura de los niveles del árbol, se coloca el `grado`, la `materia` y el `grupo` a los que pertenece cierta clase, además del `estatus` (indica el estado de una clase: 0 inactiva y 1 activa), `nombreMateria` (para indicar el nombre de la materia/clase) y `profesor` (indica el nombre del profesor que imparte dicha clase).

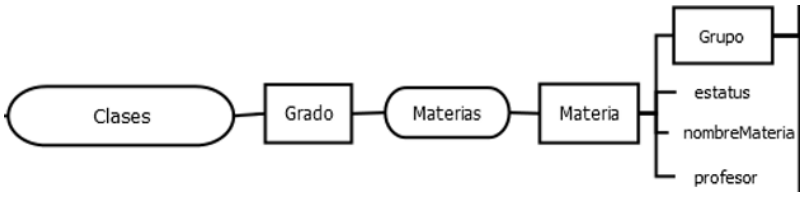

Figura 3.13: Clases en la base de datos *NoSQL* de la segunda versión del sistema.

Para los `usuarios` (Figura 3.14), se tiene la división entre los roles de usuarios que interactúan para con el *chatbot*: `alumnos` y `profesores` (recordando que la base de datos *NoSQL* de *Realtime Database*, sirve primordialmente como la base de conocimientos del agente de *chatbot*).

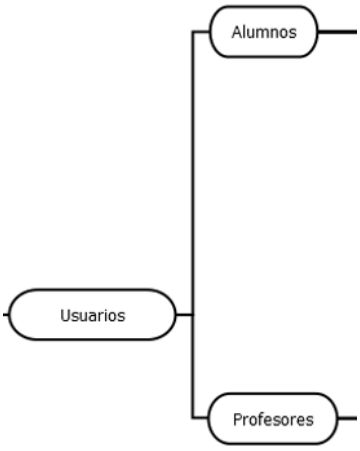

Figura 3.14: Usuarios en la base de datos *NoSQL* de la segunda versión del sistema.

Dentro de `Clases` se tienen: `Calificaciones`, `DudasAlumnos`, `Examenes`, `Materiales`, `RecordatoriosClase` y `Tareas`. Las `Calificaciones` (Figura 3.15), cotienen las calificaciones/notas de los alumnos, y tienen los elementos de `bimestreReportado` (indica en número el bimestre hasta el cual se ha calificado) y `Estudiantes` (contiene todos los alumnos que están inscritos a determinada clase y las calificaciones reportadas para dichos alumnos). Para el diseño se consideran cinco bimestres, tal como lo marca la secretaria de educación en México en las escuelas secundarias.

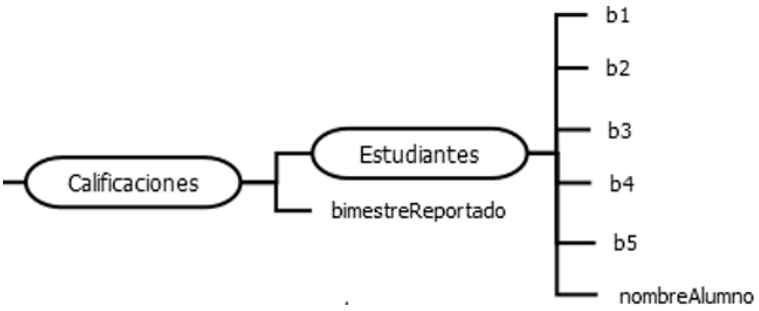

Figura 3.15: Calificaciones de clases en la base de datos *NoSQL* de la segunda versión del sistema.

`DudasAlumnos` (ver 3.16), contiene las dudas que se registran de los alumnos de determinada clase, siendo `alumno` (nombre del alumno que tuvo la duda), `correo` (correo del alumno de la duda), `estatus` (estatus de la duda: 0 no resuelta, 1 resuelta y 2 desconocida), `fecha` (fecha en la que se genero la duda), `hora` (hora en que se genero la duda) y

`MaterialRecomendados` (el o los identificadores del material recomendado para resolver la duda).

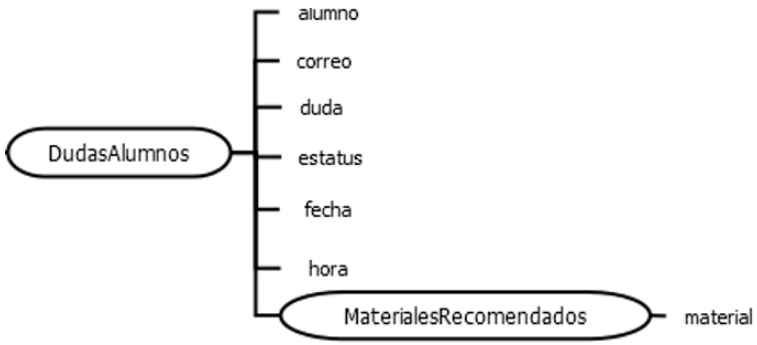

Figura 3.16: Dudas de los alumnos en las clases en la base de datos *NoSQL* de la segunda versión del sistema.

Los `Examenes` (ver 3.17), contienen los exámenes que se han registrado para determinada clase. Cuentan con los campos: `descripcion` (descripción del examen), `dia` (día de la aplicación del examen), `estatus` (estado del examen 0 inactivo y 1 activo), `hora` (hora de la aplicación del examen), `numeroConsultas` (número de consultas que ha tenido el examen), `Dudas` (dudas de los alumnos para con ese examen y los datos de dicha duda).

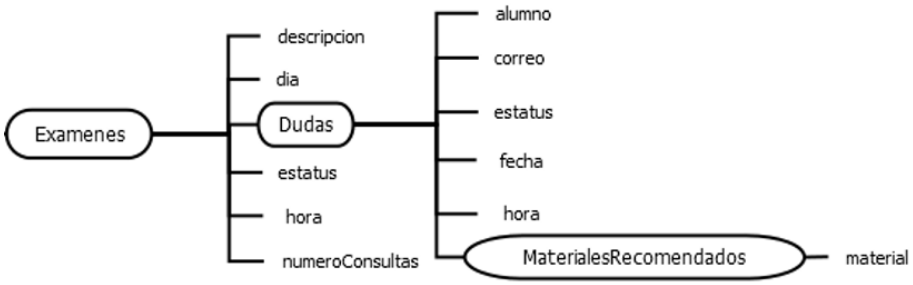

Figura 3.17: Exámenes de las clases en la base de datos *NoSQL* de la segunda versión del sistema.

Los `materiales` (ver 3.18), contienen los materiales extra-clase que se han subido a determinada clase, siendo sus campos: `descripcion` (resumen del material), `estatus` (estatus del material: 0 inactivo y 1 activo), `fecha` (fecha de subida del material), `numeroConsultas` (número de veces que ha sido consultado o recomendado el material), `Links` (links a fuentes externas de apoyo al material), `Recomendados` (dividido en: `Examenes` para indicar con el identificador del examen a qué exámenes ha sido recomendado y con el `puntaje` indicando

qué tan buena ha sido la recepción de los alumnos ante su recomendación para dicho examen. `Tareas` para indicar a qué tareas ha sido recomendado y con el `puntaje` qué tan buena ha sido la recepción de los alumnos ante su recomendación. `General` que indica las veces que ha sido la recomendación del material acertada, fallada y desconocida para resolver las dudas generales de alumnos).

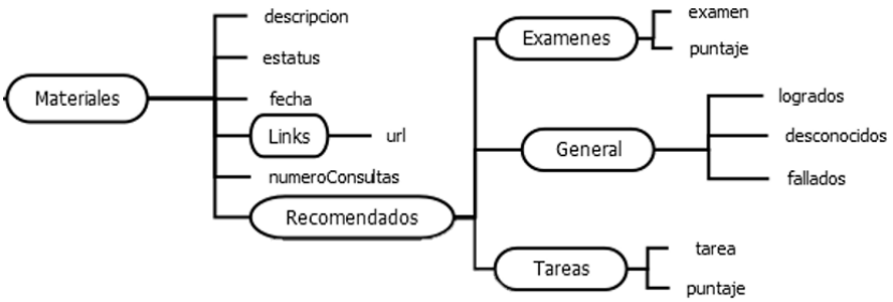

Figura 3.18: Materiales de las clases en la base de datos *NoSQL* de la segunda versión del sistema.

Los `RecordatoriosClase` (ver 3.19), contienen los recordatorios hechos por los profesores para sus clases. Contiene: el `contenido` (texto relativo al recordatorio de clase), `estatus` (estatus del recordatorio: 0 inactivo y 1 activo), `fechaLimite` (fecha en que expira el recordatorio), `horaLimite` (hora en que expira el recordatorio), `fechaPublicacion` (fecha en que fue publicado el recordatorio), `horaPublicacion` (hora en que fue publicado el recordatorio).

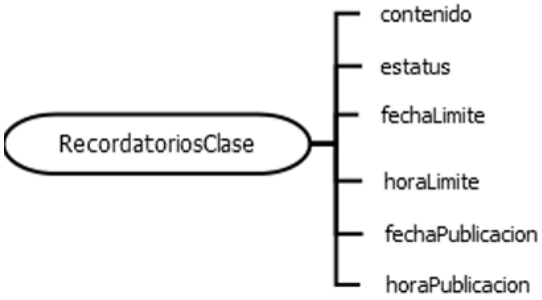

Figura 3.19: Recordatorios de las clases en la base de datos *NoSQL* de la segunda versión del sistema.

Las `Tareas` (ver 3.20), contienen las tareas asignadas por los profesores para sus clases, con los campos: `descripcion` (resumen de lo que trata la tarea), `fechaLimite` (fecha

límite en que se puede entregar la tarea), `horaLimite` (hora límite en que se puede entregar la tarea), `estatus` (estatus de la tarea: 0 inactiva y 1 activa), `tema` (título de la tarea), `Dudas` (dudas que se han generado de la tarea a los alumnos) y `Entregados` (las tareas que han sido entregadas por los alumnos), su `estatus` (2 entregada fuera de tiempo, 1 entregada a tiempo y 0 no entregada), además contiene la retroalimentación a la tarea por parte del profesor.

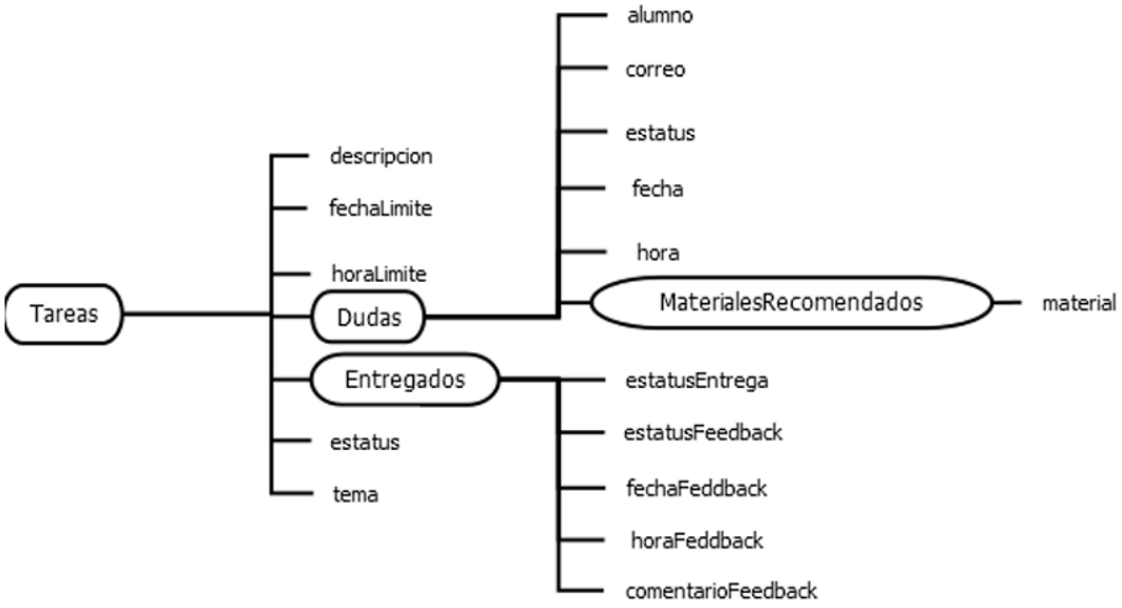

Figura 3.20: Tareas en la base de datos *NoSQL* de la segunda versión del sistema.

Para los `Alumnos` (Figura 3.21), se tienen sus datos personales y de autenticación como `clave` (contraseña), `correo` (correo del usuario), `estatus` (estatus de la cuenta de usuario: 0 inactiva y 1 activa), `grado` (grado del alumno), `grupo` (grupo del alumno), `nombre` (nombre del alumno), `EstadosAnimo` (los estados de ánimo detectados y registrados del alumno, con su fecha de registro, estatus, así como algún consejo/retroalimentación que haya sido proporcionado por profesores ante ese estado de ánimo). También, se tienen `Notificaciones` (que indican el estado de algunas notificaciones o pendientes por checar en diferentes rubros del sistema, siendo un valor numérico el que indica su estatus: 0 sin notificación activa, 1 con notificación activa y 2 no hay nada registrado). Por último, se tiene `RetroalimentacionDudas`, para las dudas que ha tenido el alumno y que ya han sido retroalimentadas por profesores.

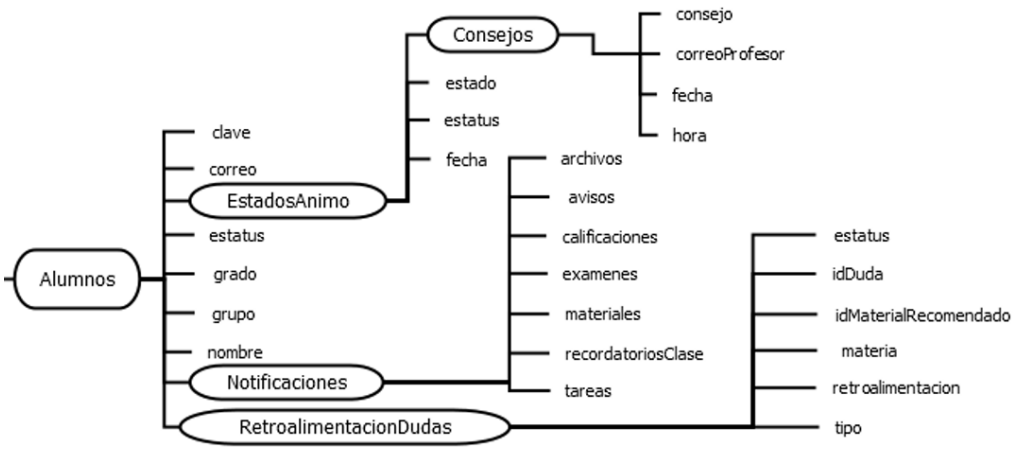

Figura 3.21: Alumnos en la base de datos *NoSQL* de la segunda versión del sistema.

Para los `Profesores` (Figura 3.22), se cuenta con sus datos personales y de autenticación como: `clave` (contraseña de usuario), `correo` (correo del usuario), `estatus` (estatus de la cuenta de usuario: 0 inactiva y 1 activa), `estatusRecepcion` (indica si el profesor puede recibir mensajes de alumnos a los que no les imparte clases), `nombre` (nombre del profesor), `AlertasAnimo` (registro de las diferentes alertas de ánimo generadas por los alumnos) y `Notificaciones` (indican el estado de algunas notificaciones o pendientes por checar en diferentes rubros del sistema, siendo un valor numérico el que indica su estatus: 0 sin notificación activa, 1 con notificación activa y 2 no hay nada registrado).

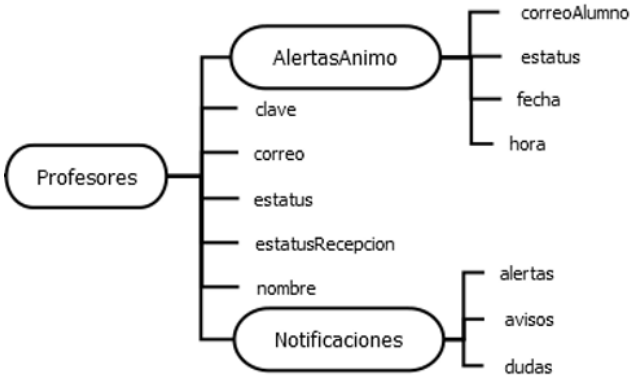

Figura 3.22: Profesores en la base de datos *NoSQL* de la segunda versión del sistema.

3.3.3 Nueva versión del agente de *chatbot*

El agente para la nueva versión plantea hacer uso de más *intents* y entidades, así como empezar a usar contextos, ver Figura 3.23.

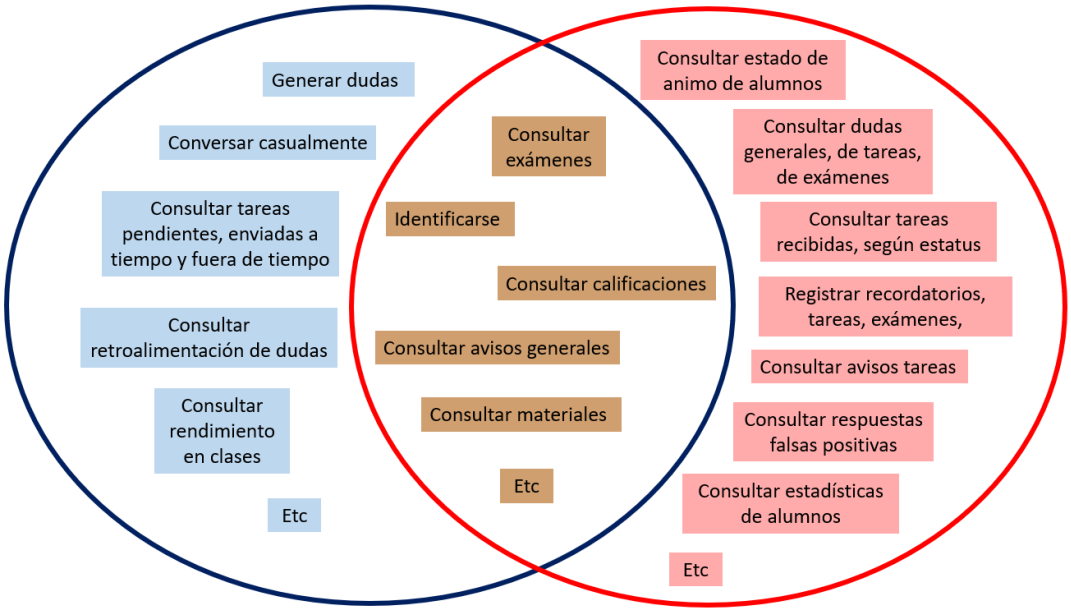

Figura 3.23: Diagrama de *Venn* de las *entities* de los profesores y alumnos.

Los *intents* deben de garantizar lo siguiente:

Para un alumno:
Identificarse, consultar avisos generales, consultar calificaciones, consultar su rendimiento en clases, consultar tareas pendientes, consultar tareas enviadas, consultar tareas enviadas fuera de tiempo, consultar recordatorios de clase, consultar materiales, consultar retroalimentaciones de sus dudas, consultar dudas, consultar exámenes, generar dudas de tareas, generar dudas generales, generar dudas de exámenes, conversar casualmente, consultar consejos de los profesores, retroalimentar las respuestas del agente.

Para un profesor:
Identificarse, consultar avisos generales, consultar/registrar calificaciones, consultar el rendimiento de sus alumnos, consultar dudas generadas de exámenes, consultar dudas generadas de tareas, consultar dudas generales generadas, consultar estado de ánimo de los alumnos, consultar tareas recibidas a tiempo, consultar tareas recibidas fuera de tiempo, consultar tareas pendientes de recibir, consultar/registrar exámenes, consultar/registrar tareas, con-

sultar/registrar recordatorios de clase, consultar/registrar materiales extra clase, consultar estadísticas de sus clases, consultar respuestas falsas positivas.

Las nuevas entidades usadas, fueron: `InfoInicioFinCursos` (para la detección de palabras clave en consultas sobre el inicio o fin de cursos), `Inscripcion` (para la detección de palabras clave en consultas sobre inscripciones o reinscripciones), `Sentimientos` (para la detección de palabras clave en conversaciones donde se tuvieran sentimientos y el reconocimiento de los mismos, ver Figura 3.24).

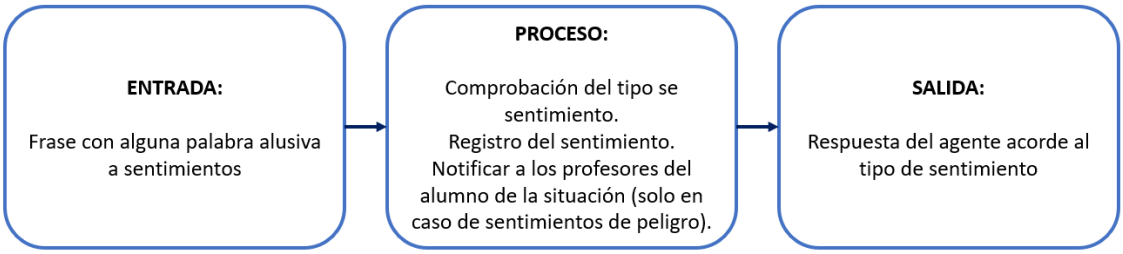

Figura 3.24: Diagrama EPS para la detección de sentimientos.

Para ello, los sentimientos definidos se clasifican en básicos y de peligro (ver Figura 3.25).

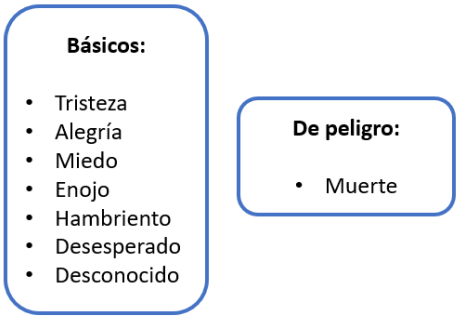

Figura 3.25: Tipos de sentimientos usados para la entidad de sentimientos.

Se analizaron y diseñaron contextos, que sirven para identificar información valiosa y almacenarla, se diseñaron dos grupos:

Contextos para la detección de usuarios:
Permite la autenticación de un usuario con el agente de *chatbot*. Se diseñó un contexto de salida en el *Default Welcome Intent*, de manera que, el primer *intent* de bienvenida sirva para

identificar al usuario por medio de su correo y contraseña, datos que se le preguntan mediante el chat, y se almacenan en el contexto de salida llamado `nombreU` de dicho *intent*. El contexto es definido en todos los *intents* como contexto de entrada y salida, con el fin de que los datos de la autenticación de usuario sigan fluyendo y se recuerden durante toda la conversación. Con el contexto `nombreU`, se identifica a un usuario, evitando con ello, preguntar datos personales con cada *intent* y automáticamente dirigir con la información de dicho usuario agilizando las tareas. E.g, si se autentifica un usuario alumno, cuando él quiera consultar algo pertinente a su rol, como son sus tareas, no sea necesario preguntarle su grado y grupo, o bien, su nombre y contraseña, debido a que ya se identificó desde un inicio, por lo que, ya se sabe en qué grado, grupo y materias está inscrito. Para guiarse puede ver la Figura del ejemplo 3.26.

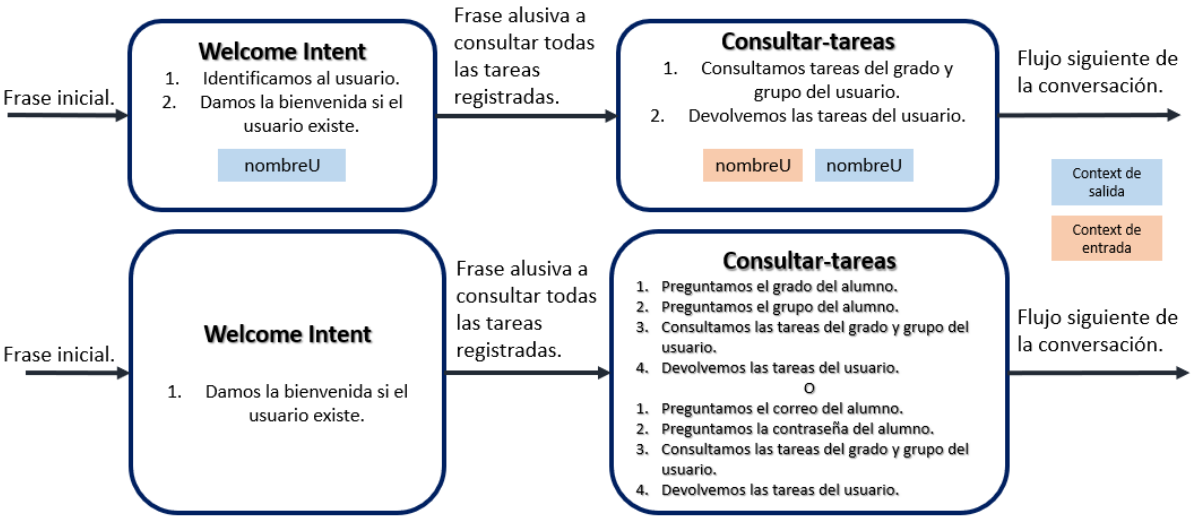

Figura 3.26: Ejemplo del uso del contexto para identificar un usuario.

Contextos para la retención de dudas:

Las dudas de un alumno, pueden ser: Generales, de Tareas y de Exámenes. Se diseñó un contexto de salida para cada tipo de duda. El contexto sirve para retener información sobre la duda que acababa de ser generada, con el fin de que al alumno se le pregunte si su duda ha sido resuelta, el alumno puede responder si o no, y por último con la duda identificada y con el resultado de saber si la duda ha sido resuelta en verdad, definir la duda como resuelta o no. Lo anterior sirve para llevar un registro de las respuestas falsas positivas del sistema (Respuestas que se le dan a un usuario, pero realmente no aclaran sus dudas). Su diseño es similar al mostrado en la Figura 3.27.

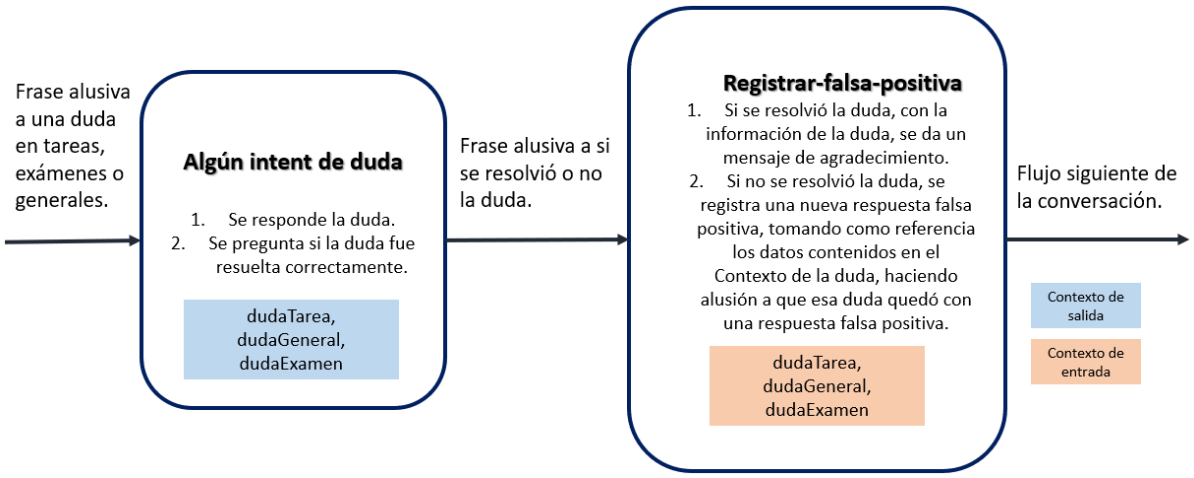

Figura 3.27: Ejemplo del uso del *context* para registrar respuestas falsas positivas.

3.3.4 Arquitectura de la nueva versión del sistema

Tomando como referencia la arquitectura del sistema base, siendo solamente modificadas partes internas de los proyectos y módulos que lo conforman. Las entradas y salidas se dan mediante canales de texto. La arquitectura de CINVESTAVCHATBOT, está dividida en la parte de *Front-End* (con las interfaces de usuario multimodal, para cada rol definido en el sistema), vinculado con las entradas y salidas, y la parte de *Back-End* (que define componentes para cada rol del sistema, en los cuales se realizan las operaciones pertinentes de cada rol).

Se diseñó el proyecto API-CHATBOT-BASEDEDATOS que sirve como *ORM* para mapear la base de datos relacional a objetos y facilituar su uso en CINVESTAVCHATBOT. CINVESTAVCHATBOT, se conecta con servicios externos como *Dialogflow* (para el agente de *chatbot*) en su segunda versión y *Firebase* (para las bases de datos *NoSQL*, tanto de datos en texto plano (*Realtime*), como de archivos complejos (*Cloud Storage*)). El vínculo entre *Dialogflow* y *Firebase*, se da debido a que *Firebase* provee de datos al agente alojado en *Dialogflow*, siendo el agente mismo el que almacena datos en *Firebase*.

Con lo anterior se obtiene la arquitectura final del sistema para la tesis, ver Figura 3.28.

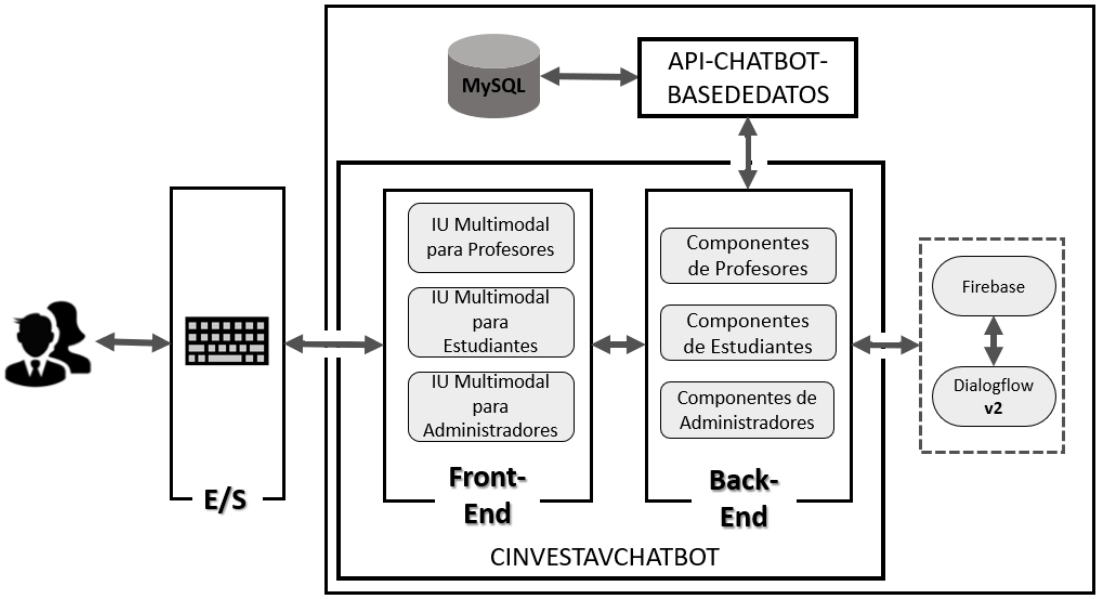

Figura 3.28: Arquitectura del sistema propuesto para la tesis.

Capítulo 4

Implementación

El presente capítulo permite dar un panorama general de la implementación del presente trabajo de tesis. Al inicio, se exponen brevemente las tecnologías y herramientas usadas para el desarrollo del sistema, dando una breve explicación del uso de dicha tecnología, su aplicación en el desarrollo del sistema y la versión usada. En la misma sección se explican detalles teóricos de las tecnologías de Google, *Firebase* y *Dialogflow*, con la finalidad de dar a conocer su arquitectura, funcionalidad y capacidades, ya que fueron herramientas esenciales para el desarrollo del sistema propuesto en la presente tesis. (cf. Sección 4.1). Posteriormente, se describe la implementación de algunas funcionalidades del sistema como resolución de dudas, detección de respuestas falsas positivas y detección de sentimientos, mediante diagramas de flujo (cf. Sección 4.2). El capítulo continúa mostrando los *intents* que se han desarrollado para aumentar las funcionalidades del *chatbot*. Se muestran las características por rol del sistema que fueron implementadas, cerrando con una comparativa entre versiones del sistema (cf. Sección 4.3).

4.1 Tecnologías y herramientas utilizadas

Las tecnologías y herramientas usadas para el desarrollo del sistema, son las mismas que se usaron en la versión previa como son:

- *Angular*: es un *framework* de código abierto para el desarrollo de aplicaciones móviles y web de una sola página. Fue creado con el lenguaje de programación *TypeScript*. La arquitectura que utiliza se basa en el patrón de diseño MVC (Modelo Vista Controlador)(Deacon, 2009), permitiendo dividir la información, su representación y la forma de interacción. Un aspecto importante es que a nivel código, se maneja por medio de componentes, lo cual otorga una gran escalabilidad, haciendo que el desarrollo y las pruebas sean sencillas. Se usó la versión 8.2.5 para el desarrollo del proyecto CINVES-

TAVCHATBOT, el cual contiene los componentes visuales y parte de los internos del sistema, ver figura 3.2 del capítulo anterior.

- *NodeJS*: es un entorno de ejecución multiplataforma para el lenguaje de programación *JavaScript* orientado a eventos asíncronos, que se aloja en la capa del servidor, aunque no se limita a ella. Permite crear aplicaciones web escalables de manera sencilla. Utiliza *npm* como sistema de gestión de paquetes por omisión. Para el proyecto fue usado en su versión 14.15.3.

- *AdonisJS*: es un entorno de trabajo enfocado al desarrollo web. Se basa principalmente en *NodeJS* y utiliza el patrón de diseño MVC. Permitió crear un *ORM* (*Object Relational Mapping*) para el mapeo de la base de datos relacional en objetos y su posterior uso en el sistema creado en *Angular*. ORM es un modelo de programación que permite mapear la estructura de una base de datos relacional a una estructura orientada a objetos, dando como resultado, una base de datos virtual orientada a objetos, de manera que las acciones *CRUD* (Crear, Leer, Modificar y Borrar) se realicen por medio de la base de datos de objetos y afecten a la base de datos relacional. Lo anterior brinda una mayor rapidez, simplicidad y manejo al momento de programar siguiendo un único paradigma.

- *MySQL*: es un sistema de gestión de base de datos relacional. Actualmente pertenece a *Oracle Corporation* y es considerado como uno de los gestores de bases de datos más populares de código abierto, altamente documentado y amigable para trabajar en entornos web. Se uso la versión 10.4.17.

- *PHPMyAdmin*: es una herramienta para el manejo visual de bases de datos relacionales de *MySQL*, permitiendo crear bases de datos, administrarlas y visualizarlas gráficamente. Se utilizo la versión 5.0.4.

- *XAMPP*: permite ejecutar un servidor apache web local, utilizándose para el proyecto en su versión 3.2.4.

- *Postman*: permite el envió de peticiones *HTTP REST*, además de servir para pruebas de peticiones a sitios. Para el proyecto se usó en la creación inicial del usuario administrador del sistema.

- *Git*: sirve para el control de versiones y como repositorio de respaldo en la nube para proyectos, principalmente de software. Se usó en su versión 2.29.2 para mantener respaldado el sistema y contener las diferentes versiones del mismo.

- *Visual Studio Code*: sirve como editor de código permitiendo, por su gran repertorio de paquetes para programación web, desarrollar proyectos en diferentes lenguajes web. Para el proyecto, sirvió como el editor del código principal, usándose su versión 1.59.1.

Para probar y visualizar el sistema, también se utilizaron navegadores como: *Google Chrome*, *Mozilla Firefox* y *Edge*. El sistema operativo usado para el desarrollo y las pruebas, fue principalmente *Windows 10* y secundariamente *Ubuntu 19 LTS*. Otras dos tecnologías usadas para el desarrollo del proyecto fueron *Firebase* y *Dialogflow* en su segunda versión, las cuales se exponen a continuación.

4.1.1 Firebase

Es un proveedor de servicios en la nube y *back-end* como servicio, adquirido por *Google* en 2014, para el desarrollo de aplicaciones móviles y web. Está compuesto por varios servicios, entre ellos una base de datos en tiempo real *NoSQL*[1], almacenada como *JSON*, que proporciona una *API* que otorga a los desarrolladores la capacidad de almacenar y sincronizar los datos a través de múltiples clientes en tiempo real. Otro servicio es el almacenamiento en la nube, siendo un sistema que permite a los desarrolladores guardar los ficheros de sus aplicaciones y vincularlos con referencias a un árbol de ficheros. Algunos detalles de la base de datos en tiempo real (*Realtime Database*) y del almacenamiento en la nube (*Cloud Firestore*) se describen a continuación.

Realtime Database

Es una base de datos *NoSQL* alojada en la nube, que permite sincronizar a los clientes en tiempo real con sus datos. Los datos se almacenan en formato *JSON* y se sincronizan en tiempo real con todos los clientes conectados, de forma que comparten una instancia de *Realtime Database* y reciben actualizaciones automáticamente en milisegundos ante la modificación de los datos. La base de datos de *Realtime Database* puede responder de forma asíncrona cuando no se cuenta con conexión a Internet, gracias a que el *SDK* de *Firebase* hace que los datos persistan en el disco. Cuando se restablece la conexión, el cliente recibe los cambios y los sincroniza con el estado actual del servidor. Otra ventaja es que se puede acceder a *Realtime Database* desde un dispositivo móvil o navegador web, la seguridad y validación de los datos están disponibles por medio de las reglas de seguridad de *Firebase Realtime Database*, estas son reglas basadas en expresiones las cuales indican cuándo se leen y escriben datos.

Los datos alojados en el *Realtime Database* no se almacenan en tablas ni registros, debido a que la base de datos que usa, es una base de datos *NoSQL*, los datos son almacenados como objetos *JSON*, por lo cual la base de datos puede conceptualizarse como un árbol *JSON* alojado en la nube.

[1] Aquel tipo de base de datos que no cumple con el esquema relacional, por lo cual no utiliza las tablas como la estructura principal de su modelo, utiliza documentos, algunas veces compuestos en estructuras de árboles. Con los años han tomado mayor relevancia al permitir el manejo de volúmenes de información mayores a los utilizados en bases de datos relacionales

Cloud Firestore

Es una base de datos *NoSQL* alojada en la nube. Utiliza un modelo en el cual los datos se almacenan en documentos con campos que se asignan a valores, estos valores se organizan en colecciones que se pueden ver como contenedores, ver Figura 4.1. Los documentos, pueden contener además objetos anidados complejos y sub colecciones. También utiliza una sincronización de datos similar a la de *Realtime Database*, lo que permite actualizar datos de forma asíncrona.

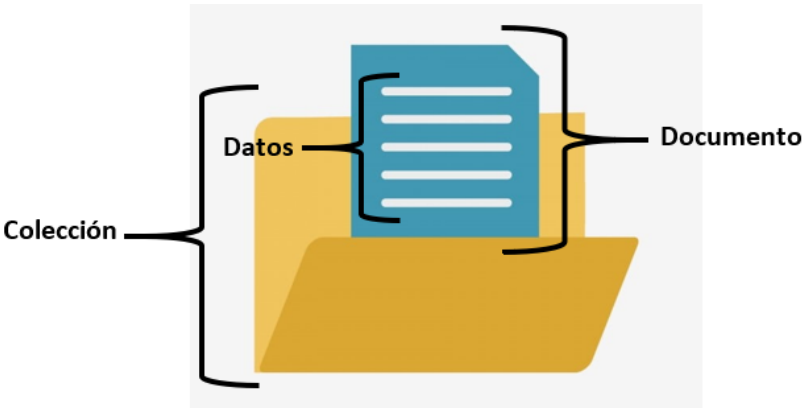

Figura 4.1: Modelo de datos de *Cloud Firestore*.

La diferencia entre *Realtime Database* y *Cloud Firestore*, radica en su modelo de datos, en el primero los datos son almacenados como un gran árbol *JSON* y en el segundo se almacenan los datos como colecciones de documentos. De esta forma, uno se enfoca a datos simples, principalmente de texto plano, mientas otro se enfoca a datos complejos y jerárquicos como archivos que se pueden organizar en documentos.

4.1.2 Dialogflow

Dialogflow es una plataforma para la comprensión de lenguaje natural, proporcionada por *Google*, que facilita el diseño e integración de agentes de *chatbot*. Es capaz de procesar diferentes tipos de entradas en formatos de texto y voz, así como de devolver salidas por los mismos canales.

Dialogflow proporciona agentes virtuales, encargados de entender y manejar las conversaciones con los usuarios finales. Cuando se crea un agente, se crean dos *intents* predeterminados: El *intent* de Bienvenida (*Welcome Intent*) que coincide cuando el usuario inicia una conversación con el agente y que permite al usuario saber que se iniciará la conversación. El otro

intent, es el de retroceso (*Default Fallback Intent*), que se presenta cuando el agente no puede hacer coincidir la expresión del usuario final con la de algún *intent*.

Intents

Clasifica la intención de un usuario dependiendo de una entrada provista. En resumen, se busca saber cuál es la intención del usuario final y su vínculo con el *intent* más coincidente. Además, sirve para extraer información de una expresión de usuario final. Ver Figura 4.2.

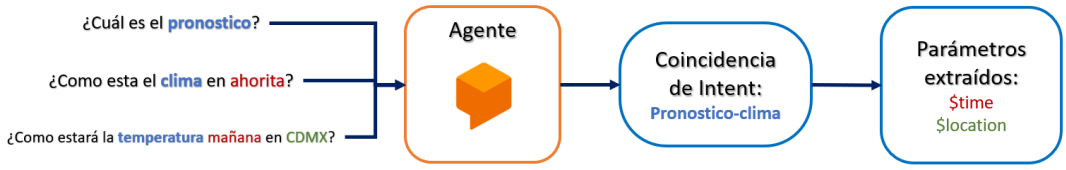

Figura 4.2: Flujo de datos en un *intent*.

Un intent contiene:

- Frases de entrenamiento: Frases de ejemplo de las entradas más comunes que un usuario podría decir. Cuando una entrada de un usuario se parece a una de estas frases, se puede realizar una coincidencia de *intent*. El aprendizaje automático de *Dialogflow* se encarga de entrenar con frases similares, de forma que no es necesario poner una lista infinita de frases de entrenamiento.
- Acción: Usada para activar ciertas acciones definidas en el sistema cuando un *intent* coincide.
- Parámetros: Los valores extraídos de la expresión del usuario final se denominan como parámetros, cada parámetro tiene un tipo (llamado entidad), que representa datos ya estructurados en el sistema.
- Respuestas: Sirven como punto final, es la respuesta que es mostrada como salida una vez que termina el *intent*.

Coincidencia de *intents*

Cuando un usuario final ingresa una frase, denominada como "expresión de usuario final", *Dialogflow* compara la frase entrante contra las frases de entrenamiento, con la finalidad de encontrar una coincidencia para cada uno de los *intents*, a dicha coincidencia de *intents* se le conoce como clasificación de intents (ver Figura 4.3).

Cuando se busca un *intent* que coincida con una frase de entrada, a cada *intent* se le asigna una puntuación conocida como puntuación de confianza o confianza de detección de *intents*, la cual va de 0 a 1, siendo 0 una confianza nula y 1 una confianza completa, e indica qué tan coincidente es el *intent* con la entrada del usuario final. Posteriormente, la puntuación de confianza se verifica contra un umbral de clasificación definido con anterioridad. Las puntuaciones pueden arrojar tres posibles resultados:

- Si el *intent* con la puntuación de confianza más alta, tiene un puntaje mayor o igual a la configuración del umbral de clasificación del algoritmo de aprendizaje (AA), se muestra como una coincidencia con dicho *intent*.
- Si ningún *intent* logra alcanzar el umbral se establece la coincidencia con el *Default Fallback Intent*.
- Si ningún *intent* alcanza el umbral y no hay un *Default Fallback Intent*, no existe ninguna coincidencia.

El umbral de clasificación AA, sirve para filtrar los falsos positivos y controlar la confianza en la detección de *intents* mínima requerida. Cuando dos o más *intents* coinciden con la misma expresión de un usuario final y se tienen puntuaciones de confianza similares, se usa la prioridad del *intent* para ubicar la mejor coincidencia, la cual se define como Muy alta, Alta, Normal, Baja e Ignorada.

En su entrenamiento, *Dialogflow* usa los datos de entrenamiento (frases de entrenamiento de *intents*) con el fin de compilar modelos de aprendizaje automático, específicamente para el agente que se construyó. *Dialogflow* usa las frases de entrenamiento como ejemplos para un modelo de aprendizaje automático, a fin de hacer coincidir las expresiones del usuario con *intents*. Además, actualiza el modelo de aprendizaje automático del agente cada vez que se modifican los *intents* y las entidades. El flujo básico que sigue la coincidencia de *intents* para reponder al usuario final, se puede ver en la Figura 4.3 y sigue los siguientes pasos:

1. Se recibe por el agente la expresión del usuario final.
2. El agente compara la expresión del usuario final con cada *intent*, seleccionando el *intent* que tenga la confianza más alta y que pase el umbral de clasificación. Para ello, toma las frases de entrenamiento, parámetros y prioridades del *intent*.
3. Una vez que se tiene elegido el *intent*, se devuelve una respuesta de las que tiene asociadas el *intent* coincidente.

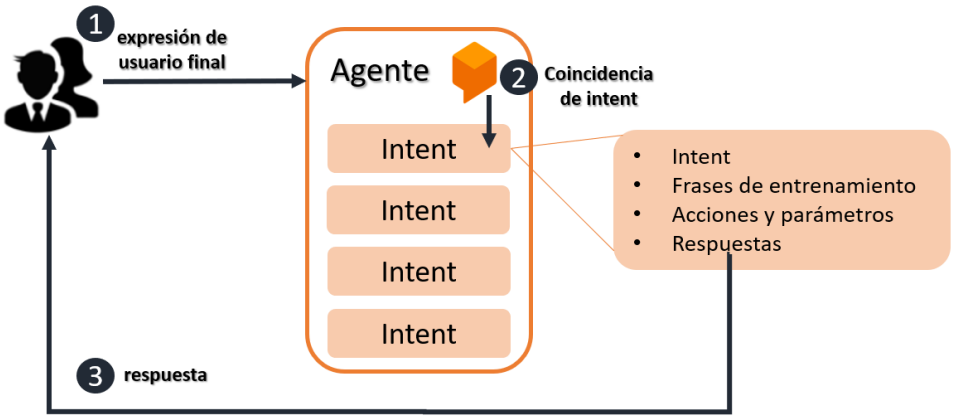

Figura 4.3: Flujo general en la coincidencia de *intent*.

Entidades

Sirven como tipo de dato para cada parámetro de un *intent*, determinan la forma en cómo se extraen los datos de una expresión de usuario final. Pueden ser entidades del sistema, las cuales ya están predefinidas o entidades personalizadas creadas por los usuarios. Los principales aspectos en una entidad son:

- Tipo: define el tipo de información que se desea extraer de la entrada del usuario final.
- Entrada de entidad: cada tipo de entidad puede tener diferentes entradas, cada entrada brinda un conjunto de palabras o frases consideradas equivalentes.
- Valor de referencia de una entidad y sinónimos: El valor de referencia es el valor base de una entidad y los sinónimos son valores de palabras o frases equivalentes al valor de referencia de la entidad.

Para las entidades, *Dialogflow* puede manejar sin problemas variaciones simples como plurales y mayúsculas.

Contextos

El flujo de la conversación se puede controlar por medio de contextos, éstos permiten al agente descubir de qué se está hablando, brindándole más información sobre de qué trata o puede tratar una conversación. Se pueden establecer contextos de entrada y salida para los *intents*, de forma que, si un *intent* es de entrada puede cargar información previa, se trata de datos que anteriormente se han obtenido a lo largo de la conversación. Si se trata de un *intent* de salida, significa que el *intent* producirá nuevos datos para nutrir la información de la

conversación. *Dialogflow* verifica si hay contextos activos, busca que los *intents* configurados con los contextos activos, tengan probabilidad de activarse. Un ejemplo del uso de un contexto en un agente de ventas (ver Figura 4.4), es el siguiente:

1. El usuario solicita información sobre sus pedidos.
2. *Dialogflow* hace coincidir esta expresión con el *intent* `Comprobar-informacion`. Este *intent* tiene un contexto de salida llamado `perfil`, por lo que este contexto se activa.
3. El agente solicita al usuario el tipo de información que desea obtener sobre sus pedidos.
4. El usuario dice: «Mis pedidos cancelados».
5. *Dialogflow* hace coincidir esta expresión con el *intent* `Consultar-pedidos-cancelados` el cual tiene un contexto de entrada llamado `perfil` activo y coincidente.
6. Después de que el sistema realiza las consultas en la BD, el agente responde con información sobre pedidos cancelados.

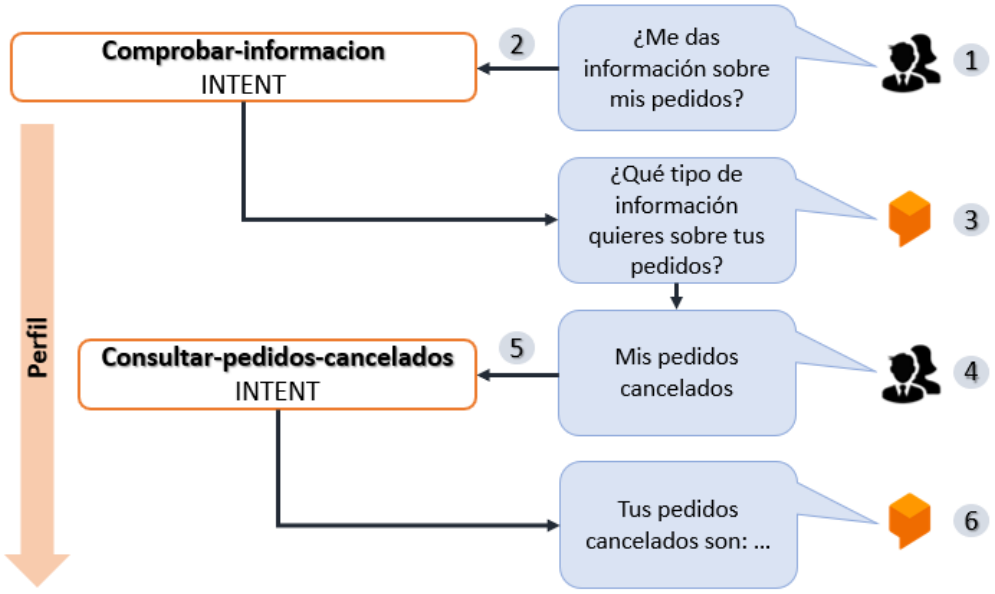

Figura 4.4: Ejemplo del flujo y uso de contextos.

Integraciones

Dialogflow se puede integrar con diferentes servicios, por ejemplo: *Google*, redes sociales conocidas como *Messenger Facebook*, *WhatsApp*, entre otras. De forma predeterminada, el agente responde a un *intent* coincidente por medio de una respuesta estática. Sin embargo, si se usa alguna opción de integración externa, *Dialogflow* responde a ese *intent* con una llamada a un servicio que se haya definido dentro del mismo servicio. De manera que cuando un *intent* tiene una coincidencia, *Dialogflow* envía una solicitud al servicio de *webhook* con información sobre el *intent* coincidente, el servicio externo realiza la acción programada y responde a *Dialogflow* con una respuesta directa desde el *webhook*. A grandes rasgos los pasos que se siguen son, ver Figura 4.5:

1. El usuario ingresa una expresión.

2. *Dialogflow* hace coincidir la expresión con un *intent* y extrae los parámetros.

3. *Dialogflow* envía un mensaje de solicitud al servicio de *webhook*, dicho mensaje contiene información sobre el *intent* coincidente, la acción, parámetros y la respuesta definida para ese *intent*.

4. El servicio realiza las acciones necesarias, por ejemplo, consultas a BD o llamadas externas a *APIs*.

5. El servicio envía un mensaje de respuesta de *webhook* a *Dialogflow*, el cual contiene la respuesta a dar al usuario.

6. *Dialogflow* envía la respuesta al usuario.

7. El usuario recibe la respuesta.

Dialogflow tiene un editor de código integrado, con el cual se puede crear código de entrega e implementar el código de interés en *Cloud Functions*. Sin embargo, de la mano de *Cloud Functions* (funciones que permiten el vínculo y personalización del comportamiento de cada *intent* con las respuestas del usuario), se puede crear un controlador de entregas personalizado, donde su función principal define las funciones asociadas a los *intents*. De esta forma, una de estas funciones se ejecutará solo cuando su *intent* asociado ha sido activado. Se puede, desde el controlador de entregas, acceder a los parámetros de un *intent* por medio de un objeto de tipo *agent* y retornar resultados conforme se hayan programado, desde un servicio externo.

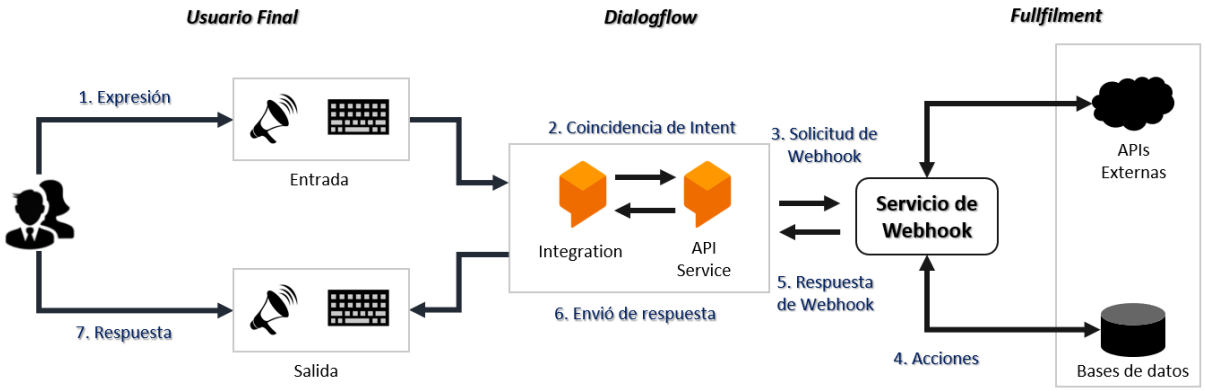

Figura 4.5: Ejemplo del flujo e integraciones con *Dialogflow*.

4.2 Implementación de funcionalidades básicas

En esta sección se muestran algunos algoritmos, presentados por medio de diagramas de flujo, que fueron implementados para cumplir funcionalidades básicas del sistema.

4.2.1 Registro y respuesta a dudas

Permite la detección, registro y posterior respuesta, a dudas generadas por los alumnos. Las dudas pueden ser generales, de tareas o de exámenes. Cuando un usuario ingresa una duda, los pasos que se siguen son los siguientes:

1. El usuario ingresa una duda.
2. El sistema verifica el tipo de duda (general, de examen, de tarea).
3. El sistema verifica el contenido de la duda y lo divide en palabras.
4. Se descartan las palabras que pudieran generar ruido como adverbios y pronombres.
5. El texto de cada material de apoyo es separado en palabras, descartando las palabras que pudieran generar ruido como adverbios y pronombres.
6. Las palabras de la duda son comparadas con el material de apoyo disponible.
7. Los materiales de apoyo que sean mayores al promedio de coincidencias (PC), se añaden a las recomendaciones.

8. Se regresa una respuesta con el material de apoyo recomendado, indicando si el material ha servido en anteriores ocasiones en dudas similares.

9. Se pregunta al alumno si su duda ha sido resuelta.

La puntuación para saber si un material es recomendado o no, es el promedio de coincidencias (PC), que se calcula con la suma de todas las coincidencias de palabras ((C_m), es decir, la cantidad de palabras en el material de apoyo iguales a las palabras de la duda), entre el número de materiales disponibles (n). Es importante que el promedio de coincidencias solo se calcule cuando la cantidad de materiales es mayor a cero, como se refleja en la ecuación 4.1:

$$PC = \frac{C_m}{n} \tag{4.1}$$

Cabe mencionar que si no se cuenta con el material de apoyo, se registra la duda y se notifica al profesor. También, en caso de que no se encuentren coincidencias, es decir, que todos los materiales de apoyo tengan cero en la suma de las coincidencias de palabras (Cm), es registrada la duda y notificado el profesor.

Los algoritmos de las siguientes subsecciones, mostrados en diagramas de flujo, fueron implementados a nivel código para la resolución de dudas. Se implementaron en tres funciones distintas, contenidas en el código de las *Firebase Functions*, cada función va conectada a un *intent* de *Dialogflow*, por lo que, se crearon tres intents para las dudas que se podían generar (de tareas, exámenes y generales):

4.2.2 Resolución de dudas de tareas

Para la detección y resolución de dudas de los estudiantes respecto a sus tareas, se implementó un *intent* en *Dialogflow* con el nombre de `Dudas-tarea`. Este *intent* hace uso del contexto de entrada `nombreU`, el cual contiene la información del usuario actual, lo que permite obtener toda la información en el sistema que está relacionada con el usuario. E.g., en qué grado y grupo estudia, qué materias está cursando y qué profesores están impartiéndolas, etc. Mientras, los contextos de salida son: `nombreU` para seguir identificando al usuario actual y `dudaTarea` que contiene la información de la duda que planteó el usuario y los materiales que le fueron recomendados. Los parámetros definidos en el *intent* y que no son propios del contexto `nombreU`, son título y clase, el primero identifica la tarea en la que el alumno tiene su duda y el segundo sirve para saber a qué clase o materia pertenece su duda.

Posterior a implementar el *intent* se implementó en código el algoritmo para la resolución de dudas de tareas, dicho algoritmo se alojó en las *Firebase functions* desarrolladas en código *JavaScript* y posteriormente desplegadas en la Web. El algoritmo se compone de dos partes, la primera, figura 4.6, sirve para identificar si la entrada de un usuario es una duda relacionada a sus tareas, haciendo uso del *intent* de `Dudas-tarea`, mientras en la segunda parte, ver figura 4.7, se hacen los cálculos y se genera la respuesta final al usuario. El algoritmo está compuesto de los siguientes pasos:

- En caso de no contar con material de apoyo registrado:

  1. Se informa al alumno de que no hay material registrado.
  2. Se registra la duda.
  3. Es notificado el profesor que imparte dicha clase, que hay una duda sin responder relacionada a tareas.

- En caso de contar con material de apoyo registrado:

  1. Se consulta la descripción que tiene asociada la tarea por la cuál se pregunta.
  2. La descripción de la tarea se une a la duda que ingresó el alumno, en una misma cadena de texto, con el fin de contar con más información y con ello se pueda identificar mejor una respuesta.
  3. Se descartan de la cadena de texto resultante, pronombres y ciertas palabras, que pueden generar ruido.
  4. La cadena de texto resultante se divide en palabras.
  5. Se comparan las palabras obtenidas de la duda con las descripciones y el contenido de los materiales de apoyo registrados.
  6. Se calculan las coincidencias de cada material de apoyo con la duda de la tarea.
  7. Se calcula el promedio de ecuación 4.1.
  8. Se verifican y seleccionan los materiales de apoyo que tienen un valor de coincidencia con la duda mayor al promedio de coincidencias.
  9. Se verifica si la tarea seleccionada no tiene dudas anteriores.
  10. En caso negativo:
      a) Se muestran los materiales con mayor valor de coincidencia seleccionados.
  11. En caso positivo:
      a) Se verifica si las coincidencias seleccionadas ya han sido recomendadas antes para esa tarea.

b) Se muestran los materiales con mayor valor de coincidencia seleccionados, agregando un mensaje indicando si antes han sido recomendados o no para resolver dudas en esa tarea.

12. Se inicia la detección de respuestas falsas positivas (ver sección 4.2.5).

Figura 4.6: Primera parte del algoritmo para la resolución de dudas de tareas.

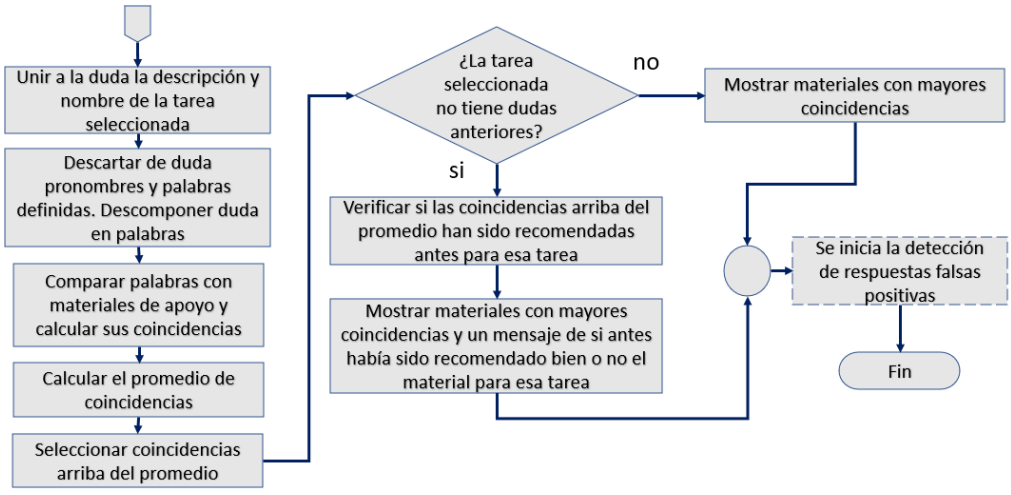

Figura 4.7: Segunda parte del algoritmo para la resolución de dudas de tareas.

4.2.3 Resolución de dudas de exámenes

Por otro lado, para detectar y resolver dudas de los estudiantes respecto a sus próximos exámenes, en *Dialogflow* se implementó el *intent* `Preguntas-acerca-examen`, que hace uso del contexto de entrada `nombreU`, el cual contiene la información del usuario actual, lo que permite obtener toda la información en el sistema que está relacionada con el usuario. E.g., en qué grado y grupo estudia, qué materias está cursando, qué profesores están impartiéndolas, qué exámenes tiene programados, etc. Los contextos de salida son: `nombreU` para seguir manteniendo información del usuario actual y `claseExamen` que contiene el nombre de la clase a la cual pertenece la duda acerca de los exámenes. Se cuenta con el parámetro llamado clase, definido dentro del *intent* y que no es propio del contexto `nombreU`, que sirve para saber a qué clase o materia pertenece la duda relacionada a exámenes.

Una vez implementado el *intent*, se plasmó en código el algoritmo para la resolución de dudas de exámenes. El algoritmo se alojó en las *Firebase functions*, desarrolladas en código *JavaScript* y que posteriormente fueron desplegadas en la Web. El algoritmo está compuesto de dos partes, la primera, Figura 4.8, sirve para identificar si la entrada es una duda relacionada a exámenes, mediante el *intent* de `Preguntas-acerca-examen`. La segunda parte, ver Figura 4.9, consiste en el algoritmo de la función donde se verifica si hay material de apoyo registrado:

- En caso de no contar con material de apoyo registrado:
  1. Se informa al alumno de que no hay material registrado.
  2. Se registra la duda.
  3. Es notificado el profesor que imparte dicha clase, que hay una duda sin responder relacionada a exámenes próximos.
- En caso de contar con material de apoyo registrado:
  1. Se obtienen los exámenes que se encuentran en estado activo y están programados para próximas fechas.
  2. Se consulta la descripción que tiene registrado cada examen programado para próximas fechas.
  3. La descripción de cada examen próximo, se une a la duda que ingresó el alumno en una misma cadena de texto.
  4. Se descartan de cada cadena de texto resultante, pronombres y ciertas palabras, que pudieran generar ruido.
  5. Las cadenas de texto se dividen en palabras.

6. Todas las palabras obtenidas de cada examen, se comparan con las descripciones y el contenido de los materiales de apoyo registrados.
7. Se calculan las coincidencias de cada material de apoyo con las palabras de cada examen próximo activo.
8. Se calcula el promedio de, ecuación 4.1.
9. Se verifican y se seleccionan los materiales de apoyo que tienen un valor de coincidencia, mayor al promedio de coincidencias.
10. Se verifica, para cada examen próximo activo que tuvo una coincidencia mayor al promedio, si no se tienen dudas anteriores.
11. En caso negativo:
    a) Se muestran los materiales con mayor valor de coincidencia seleccionados.
12. En caso positivo:
    a) Se verifica si las coincidencias seleccionadas ya han sido recomendadas antes para dicho examen.
    b) Se muestran los materiales con mayor valor de coincidencia seleccionados, agregando un mensaje en caso de que ya antes hayan sido recomendados para resolver dudas de ese examen.
13. Se inicia la detección de respuestas falsas positivas (ver sección 4.2.5).

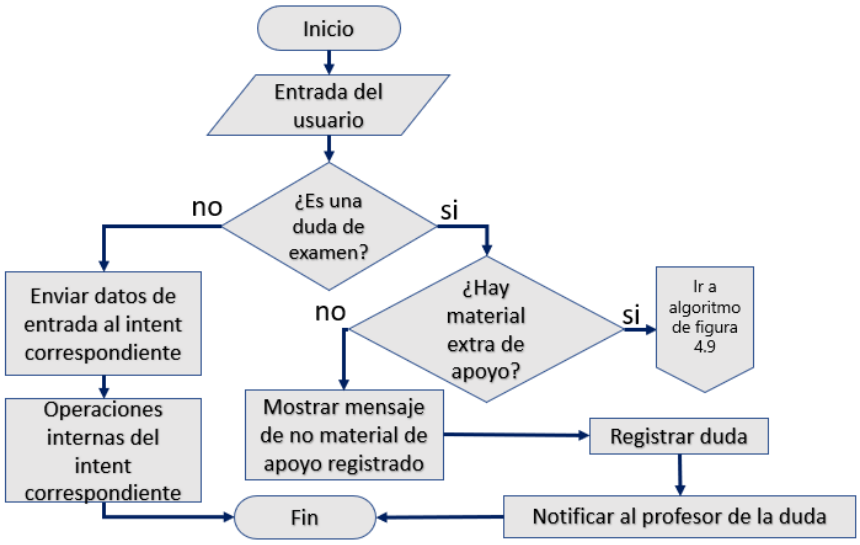

Figura 4.8: Primera parte del algoritmo para la resolución de dudas de exámenes.

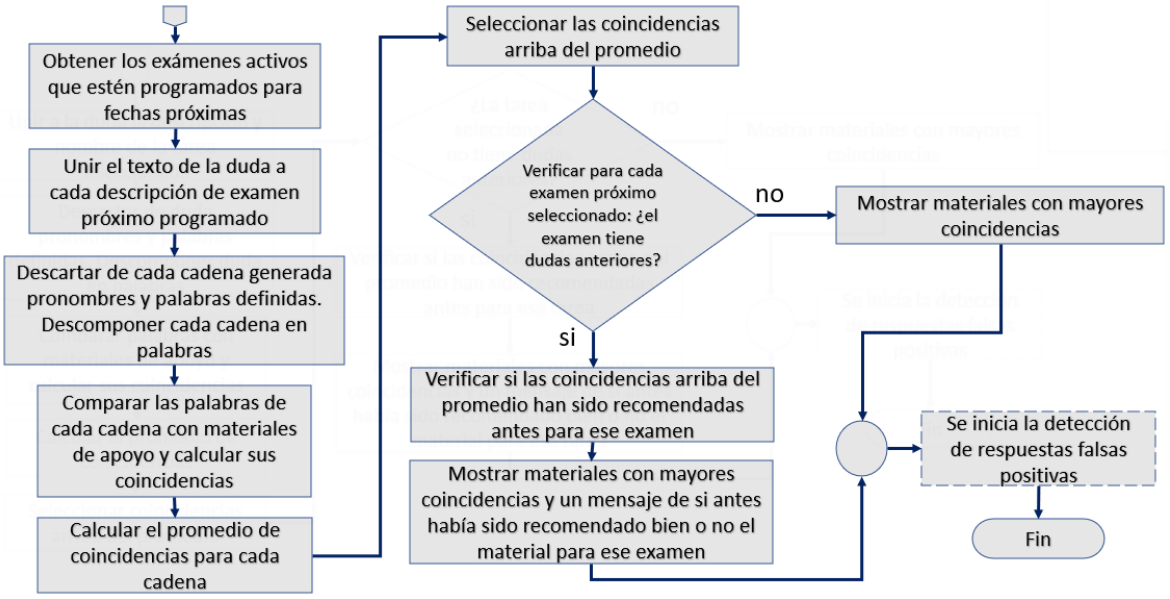

Figura 4.9: Segunda parte del algoritmo para la resolución de dudas de exámenes.

4.2.4 Resolución de dudas en general

Para la detección y resolución de dudas generales de los estudiantes, se implementó en *Dialogflow* el *intent* `Dudas-generales-alumnos`, que utiliza el contexto de entrada `nombreU`, el cual contiene la información del usuario actual, lo que permite obtener toda la información en el sistema que está relacionada con el usuario. E.g., en qué grado y grupo estudia, qué materias está tomando, qué profesores están impartiéndolas, etc. Los contextos de salida son: `nombreU` para mantener la información del usuario actual y `dudaGeneral` que contiene dos parámetros, el primero llamado `duda` que sirve para mantener el contenido de la duda general que fue planteada y el segundo llamado `clase` que permite almacenar el nombre de la clase a la cual pertenece la duda general.

Ya con el *intent* generado, fue implementado en código el algoritmo para la resolución de dudas generales. El algoritmo fue alojado en las *Firebase functions*, desarrolladas en lenguaje *JavaScript* y que posteriormente fueron desplegadas en la Web. Se compone de dos partes, la primera (ver Figura 4.10), permite identificar mediante el *intent* de `Dudas-generales-alumnos` si la entrada de un usuario es una duda de tipo general, mientras la segunda parte (ver Figura 4.11, verifica si hay material de apoyo registrado que pueda resolver la duda general:

- En caso de no tener material de apoyo registrado:

1. Se informa al alumno de que no hay material registrado.
2. Se registra la duda.
3. Se notifica al profesor que imparte dicha clase, que hay una duda general sin responder.

- En caso de si tener material de apoyo registrado:

1. De la cadena de texto de la duda general, se descartan palabras definidas y pronombres que pudieran generar ruido.
2. Se descompone la cadena de la duda en palabras.
3. Cada palabra resultante de la duda es comparada con cada descripción y datos de los materiales de apoyo registrados.
4. Se calculan las coincidencias de cada material de apoyo con la duda general.
5. Se calcula el promedio de, ecuación 4.1.
6. Son seleccionados los materiales de apoyo que tienen un valor de coincidencia mayor al promedio de coincidencias.
7. Se verifica si los materiales de apoyo seleccionados tienen dudas generales anteriores.
8. Se muestran los materiales seleccionados junto con el mensaje indicando si antes han sido recomendados bien o mal en dudas generales.
9. Se inicia la detección de respuestas falsas positivas (ver sección 4.2.5).

4.2.5 Detección de respuestas falsas positivas

Las respuestas falsas positivas, al menos para nuestro sistema, hacen alusión a respuestas que se dan cuando el *chatbot* brinda una respuesta que no resuelve la duda de un alumno. Para evaluar estas respuestas, se implementó un método que se ejecuta al momento de que el *chatbot* responde una duda. Se le pregunta al usuario si su pregunta ha sido respondida, se tienen tres posibles resultados y sus consecuencias:

- El usuario responde que sí se resolvió su duda. Se suma puntuación a los materiales que se recomendaron. Se marca la duda como resuelta.
- El usuario que no se resolvió su duda. Se resta puntuación a los materiales que se recomendaron y se registra una nueva respuesta falsa positiva. Se marca la duda como no resuelta y es notificado el profesor

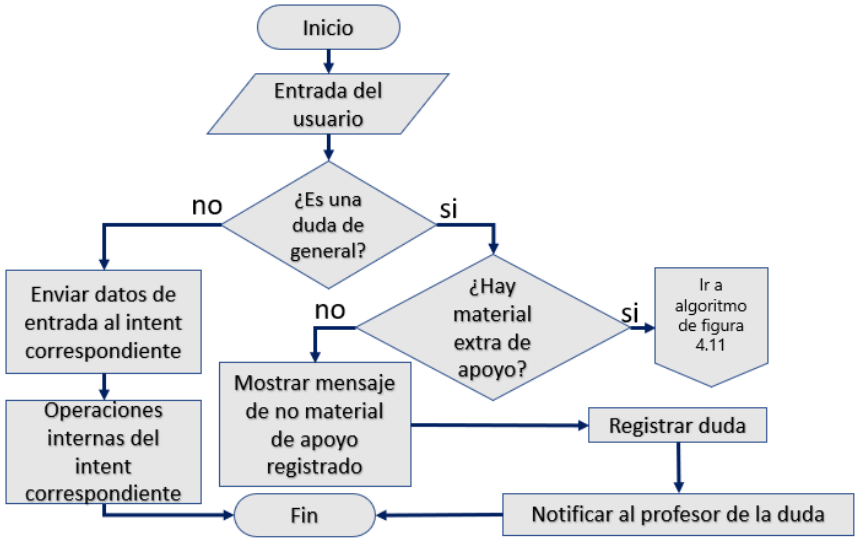

Figura 4.10: Primera parte del algoritmo para la resolución de dudas generales.

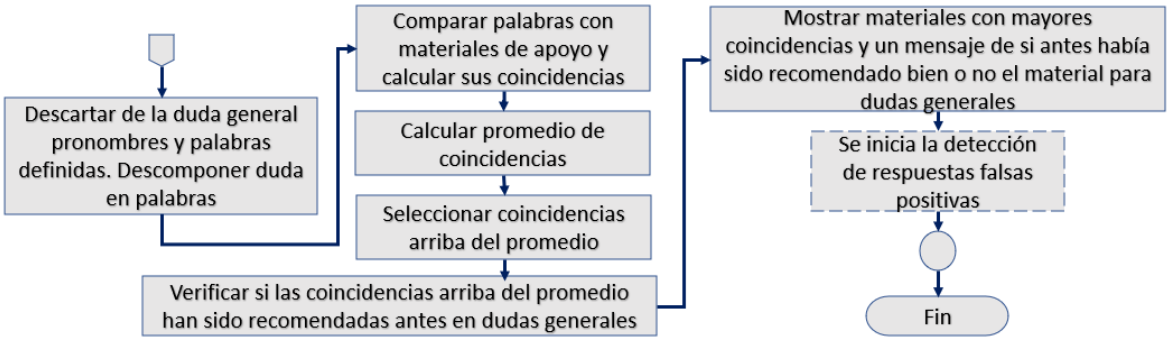

Figura 4.11: Segunda parte del algoritmo para la resolución de dudas generales.

- El usuario no responde nada. No se realiza acción alguna, se marca la duda como desconocida.

Para almacenar temporalmente la información de la duda, entre el paso en que se pregunta si se resolvió la duda y el usuario da una respuesta, se utilizan contextos, ver subsección de Contextos (ver sección 4.1.2). En el contexto se almacenan la duda del usuario y la respuesta del agente a dicha duda, este contexto se mantiene activo, siendo que cuando se le pregunta al usuario si su duda ha sido despejada, el agente por medio del contexto sabe a qué duda y respuestas se refiere el usuario, para con ello poder calificar las respuestas pertinentes. Los contextos que se manejan dependen del tipo de duda que se genera (tarea, general, examen). La Figura 4.12, muestra el algoritmo para detectar respuestas falsas positivas.

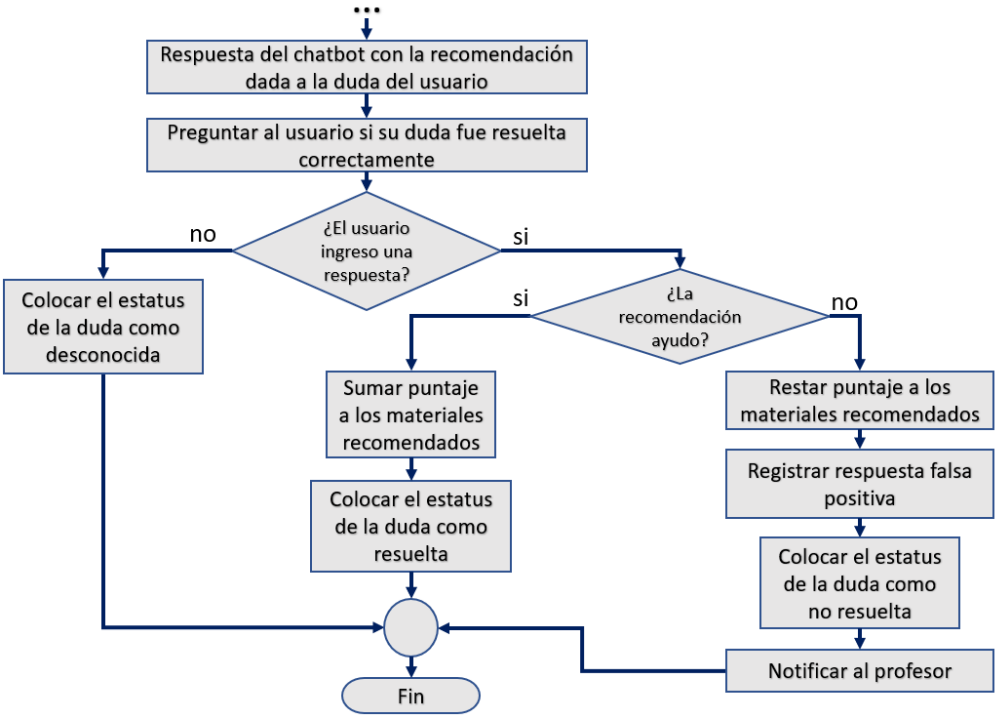

Figura 4.12: Diagrama de flujo del control de respuestas falsas positivas.

4.2.6 Detección de sentimientos

Se implemento la detección de sentimientos, por medio de frases específicas ingresadas por los alumnos al *chatbot*, se implementaron diferentes *entities* de sentimientos, siendo divididos en dos grupos:

- Básicos: tristeza, alegría, miedo, enojo, hambriento, desesperado, desconocido.
- De peligro: muerte (que incluye depresión, suicidio y asesinato).

Se desarrolló un mecanismo, para que en caso de que, a través de la frase de entrada del usuario, se detecte algún sentimiento de peligro (que haga alusión a suicidios, asesinados, daños o problemas mayores), se guarde el estado de ánimo del estudiante y se notifique a los profesores que le imparten clase, con el fin de incentivar a los profesores a llevar a cabo acciones y evitar problemas. La Figura 4.13, muestra el algoritmo para la detección de sentimientos.

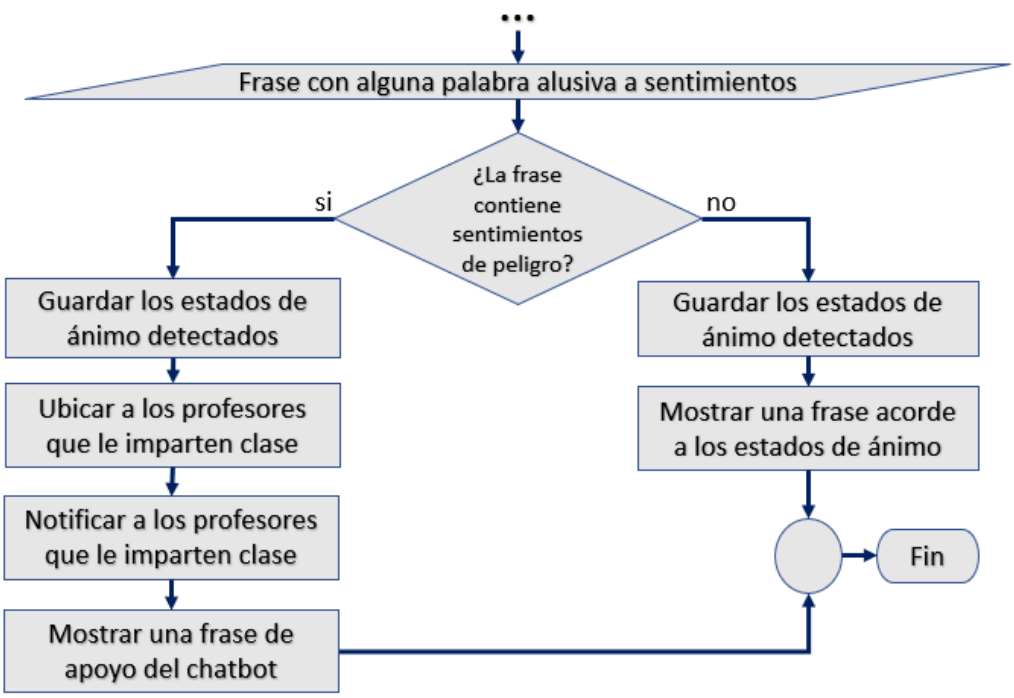

Figura 4.13: Diagrama de flujo de la detección de sentimientos y su manejo posterior.

4.3 Características implementadas

4.3.1 *Intents* desarrollados e implementados

Se creó la tabla que resume los *intents* que fueron implementados en el agente de *chatbot*, con la finalidad de reconocer un mayor número de intenciones y obtener mayores cantidades de respuestas ante preguntas planteadas por los usuarios. Vea la tabla A.1 en el Apéndice A.

4.3.2 Contextos implementados

Algunas anotaciones de los contextos implementados son:

- El contexto *nombreU*, sirve para mantener la información del usuario que actualmente está interactuando con el agente, es el contexto de entrada y salida de todos los intents con excepción del contexto de groserías, del *Default Fallback Intent* y no es un contexto de entrada del *Default Welcome Intent*.
- El contexto *nombreU*, tiene un ciclo de cien intensiones [2], lo que implica que pasada la

[2]Intenciones: forma de referirse a un cliclo de ejecución de un *intent*, que comprende desde que un *intent*

ejecución de cien *intents*, el *chatbot* aún sepa con quién esta conversando.

- Los contextos *dudaTarea*, *dudaGeneral* y *dudaExamen*, que sirven para mantener la información de una duda (que ha sido planteada y cuenta con una respuesta para posteriormente poder evaluar si su respuesta ha sido falsa positiva), son los contextos de salida de *Dudas-tarea*, *Dudas-generales-alumnos*, *Preguntas-acerca-examen*, respectivamente.
- Los contextos dudaTarea, dudaGeneral y dudaExamen, tienen un ciclo de cinco intensiones, lo cual permite conservar información acerca de la última duda hecha por un alumno y las recomendaciones por parte del *chatbot* a dicha duda, esto hasta la ejecución posterior de cinco *intents*.
- Los contextos *dudaTarea*, *dudaGeneral* y *dudaExamen*, son los Los contextos de entrada de *Resolucion-duda-tarea*, *Resolucion-duda-general* y *Resolucion-duda-examen*, respectivamente.

4.3.3 Funcionalidades implementadas para los diferentes roles de usuario del sistema

Las características que fueron implementadas para los diferentes roles de usuario en la segunda versión del sistema, divido en módulos, se resumen en las siguientes tablas:

Sistema V2	
Rol: Administrador	
Módulo	**Detalles**
Administración de los usuarios.	• Permite una interacción con el agente de Chatbot mediante la entrada y salida de texto. • Capacidad de consultar los materiales extra clase que ha subido un profesor para cierta materia, grado y grupo.
Gestión de grupos/ estudiantes.	• Da la posibilidad de visualizar posibles alumnos a agregar en cierto grado y grupo. • Permite agregar y quitar alumnos en determinado grado y grupo. • Permite visualizar los alumnos pertenecientes a cierto grado y grupo.

es elegido, realiza las operaciones pertinentes y retorna una respuesta a un usuario.

Sistema V2	
Gestión de clases.	• Permite visualizar y buscar materias registradas en cierto grado del sistema. • Da la capacidad de activar o desactivar una materia del sistema. • Da la capacidad de registrar una nueva materia para determinado grado. • Da la posibilidad de asignar materias a un determinado profesor
Administración de avisos generales.	• Permite registrar avisos generales. • Permite desactivar avisos generales. • Permite visualizar los avisos generales. • Permite seleccionar a los usuarios con acceso al aviso general.
Percepción de estudiantes.	• Permite consultar el estado de ánimo de los estudiantes de forma grupal o individual.
Configuración de cuenta.	• Permite cambiar la contraseña de usuario.

Cuadro 4.1: Tabla de funcionalidades del administrador en la versión 2.

Sistema V2	
Rol: Alumno	
Módulo	**Detalles**
Chatbot.	• Permite una interacción con el agente de *Chatbot* mediante la entrada y salida de texto.Capacidad de consultar los materiales extra clase que ha subido un profesor para cierta materia, grado y grupo. • Capacidad de consultar los pendientes actuales y pasados. • Capacidad de consultar calificaciones. • Capacidad de consultar exámenes. • Capacidad de consultar avisos generales y específicos. • Capacidad para expresar sus sentimientos. • Capacidad de consultar tareas. • Capacidad de solicitar materiales de apoyo. • Capacidad de poder formular dudas relacionadas a sus clases. • Capacidad para consultar su rendimiento en clases.
Gestión de archivos y mensajes.	• Capacidad para enviar mensajes y archivos a diferentes grupos de usuarios. • Capacidad para recibir mensajes y archivos provenientes de otros usuarios. • Capacidad visualizar archivos y mensajes recibidos
Administración de tareas.	• Permite por medio del módulo de *Chatbot* añadir y consultar tareas para determinadas materias, grados y grupos. • Permite consultar detalles en las tareas. • Permite entregar las tareas a los profesores. • Permite ver la retroalimentación de los profesores acerca de las tareas entregadas.
Progreso en clases.	• Permite conocer el progreso general y por cada clase.
Configuración de cuenta.	• Permite cambiar la contraseña de usuario.

Cuadro 4.2: Tabla de funcionalidades del alumno en la versión 2.

Sistema V2	
Rol: Profesor	
Módulo	**Detalles**
Chatbot.	• Permite una interacción con el agente de *Chatbot* mediante la entrada y salida de texto. • Permite añadir materiales extra clase a determinada materia de cierto grado y grupo. • Permite consultar materiales extra clase de determinada materia en cierto grado y grupo. • Permite consultar exámenes para determinadas materias, grados y grupos. • Permite consultar información de los exámenes. • Permite revisar dudas de los exámenes. • Permite registrar nuevos exámenes. • Permite consultar tareas para determinadas materias, grados y grupos. • Permite consultar detalles de las tareas. • Permite retroalimentar las tareas entregadas por los alumnos. • Permite registrar nuevas tareas. • Permite gestionar avisos para determinadas clases, grados y grupos. • Permite consultar las calificaciones de los alumnos. • Permite consultar el rendimiento grupal e individual de los alumnos. • Capacidad para enviar mensajes a diferentes usuarios. • Permite consultar el estado de ánimo de los estudiantes de forma grupal o individual. • Permite aconsejar a los estudiantes. • Permite conocer estados de alerta en estudiantes. • Permite conocer las dudas de los estudiantes en materiales, tareas, exámenes y generales.
Administración de tareas.	• Permite consultar tareas para determinadas materias, grados y grupos. • Permite consultar detalles en las tareas. • Permite revisar dudas de las tareas. • Permite recibir las tareas de los alumnos. • Permite retroalimentar las tareas entregadas por los alumnos.

Sistema V2	
Dudas de estudiantes.	• Permite consultar las dudas de los alumnos de forma grupal e individual. • Permite resolver dudas de los alumnos.
Administración de calificaciones.	• Permite visualizar las calificaciones de los alumnos. • Permite administrar las calificaciones de los alumnos.
Progreso de grupos.	• Permite consultar el rendimiento grupal de los alumnos. • Permite consultar el rendimiento individual de los alumnos.
Gestión de archivos y mensajes.	• Capacidad para enviar mensajes y archivos a diferentes grupos de usuarios. • Capacidad para recibir mensajes y archivos provenientes de otros usuarios. • Capacidad de visualizar archivos y mensajes recibidos.
Percepción de estudiantes.	• Permite consultar el estado de ánimo de los estudiantes de forma grupal o individual. • Permite aconsejar a los estudiantes.
Configuración de cuenta.	• Control en permitir que alumnos externos (que no toman clases con el presente profesor), no puedan enviarle mensajes ni archivos. • Permite cambiar la contraseña de usuario.

Cuadro 4.3: Tabla de funcionalidades del profesor en la versión 2.

4.3.4 Comparativa entre versiones

Por último, se presentan tablas donde se muestran las funcionalidades de la versión 2, su descripción y si la funcionalidad ya estaba implementada en la versión 1 del sistema para los diferentes tipos de usuario:

Comparativa entre V1 y V2 del Sistema: Administrador		
Funcionalidad implementada en la Versión 2:	**Descripción:**	**Versión 1:**
Visualización y búsqueda de usuarios registrados en el sistema.	Permite la visualización y búsqueda de usuarios registrados en el sistema.	SI.
Modificación del rol de un usuario.	Permite cambiar de rol a un usuario.	SI.

Comparativa entre V1 y V2 del Sistema: Administrador		
Activación y desactivación de la cuenta de un usuario.	Permite desactivar y activar cuentas de usuarios.	SI.
Adición de alumnos a determinado grado y grupo.	Permite agregar y quitar alumnos en determinado grado y grupo.	NO.
Visualización de los alumnos.	Permite visualizar los alumnos pertenecientes a cierto grado y grupo.	NO.
Búsqueda y visualización de materias.	Permite visualizar y buscar materias registradas en cierto grado del sistema.	SI.
Activación o desactivación de materias.	Capacidad de activar o desactivar una materia del sistema.	SI.
Registro de nuevas materias.	Capacidad de registrar una nueva materia para determinado grado.	SI.
Asignación de materias a profesores.	Posibilidad de asignar materias a un determinado profesor.	NO.
Registro y visualización de avisos generales.	Permite registrar y visualizar avisos generales.	NO.
Desactivación de avisos generales.	Permite desactivar avisos generales.	NO.
Visualización del estado de ánimo de los estudiantes.	Permite consultar el estado de ánimo de los estudiantes de forma grupal o individual.	NO.
Modificación de la contraseña de usuario.	Permite cambiar la contraseña de usuario.	NO.

Cuadro 4.4: Tabla de comparativa de versiones para el rol de Administrador.

Comparativa entre V1 y V2 del Sistema: Profesor		
Características implementadas en la Versión 2:	**Descripción:**	**Versión 1:**
Carga de materiales extra clase.	Permite añadir materiales extra clase a determinada materia de cierto grado y grupo.	SI.
Consulta de materiales extra clase.	Permite consultar materiales extra clase de determinada materia en cierto grado y grupo.	SI.

Comparativa entre V1 y V2 del Sistema: Profesor		
Consulta exámenes de determinadas materias, grados y grupos.	Permite consultar exámenes para determinadas materias, grados y grupos.	NO.
Revisión de dudas de exámenes.	Permite revisar dudas de los exámenes.	NO.
Registro de nuevos exámenes.	Permite registrar nuevos exámenes para determinada materia, grado y grupo.	SI.
Consulta de tareas.	Permite consultar tareas y detalles de las mismas, para determinadas materias, grados y grupos.	NO.
Adición de retroalimentación a tareas.	Permite retroalimentar las tareas entregadas por los alumnos.	NO.
Registro de nuevas tareas.	Permite registrar nuevas tareas.	SI.
Revisión de tareas cargadas por los alumnos.	Permite recibir y visualizar las tareas que han sido enviadas por los alumnos.	NO.
Gestión de avisos.	Permite gestionar avisos para determinadas clases, grados y grupos.	NO.
Consulta de calificaciones.	Permite consultar las calificaciones de los alumnos.	NO.
Carga y actualización de calificaciones de alumnos. *	Permite modificar y registrar las calificaciones de los alumnos.	NO.
Consulta del rendimiento de los alumnos.	Permite consultar el rendimiento grupal e individual de los alumnos.	NO.
Consulta del estado de ánimo de los estudiantes.	Permite consultar el estado de ánimo de los estudiantes de forma grupal o individual.	NO.
Aconsejamiento de los estudiantes.	Permite aconsejar a los estudiantes respecto a sus dificultades, dudas y estado de ánimo.	NO.

Comparativa entre V1 y V2 del Sistema: Profesor		
Notificaciones de alertas de los estudiantes.	Permite conocer estados de alerta de acuerdo al estado de ánimo de los estudiantes.	NO.
Consulta y notificación de dudas de los estudiantes.	Permite conocer las dudas de los estudiantes en materiales, tareas, exámenes y generales.	NO.
Consulta de dudas de alumnos.	Permite consultar las dudas de los alumnos de forma grupal e individual.	NO.
Resolución de dudas de alumnos.	Permite resolver dudas de los alumnos.	NO.
Gestión de archivos. *	Capacidad para enviar y recibir mensajes junto con archivos a diferentes grupos de usuarios.	NO.
Configuración de cuenta. *	Permite tener control sobre de que alumnos se puede o no, recibir mensajes y archivos. Permite cambiar la contraseña de usuario.	NO.
(*) Acción no realizada por medio del Chatbot.		

Cuadro 4.5: Tabla de comparativa de versiones para el rol de Profesor.

Comparativa entre V1 y V2 del Sistema: Alumno		
Características implementadas en la Versión 2:	**Descripción:**	**Versión 1:**
Consulta de materiales extra clase.	Capacidad de consultar los materiales extra clase que ha subido un profesor para cierta materia, grado y grupo.	SI.
Consulta de pendientes.	Capacidad de consultar los pendientes actuales y pasados.	NO.
Consulta de calificaciones.	Capacidad de consultar calificaciones.	NO.
Consulta de exámenes.	Capacidad de consultar exámenes.	NO.

Comparativa entre V1 y V2 del Sistema: Alumno		
Consulta de avisos generales y específicos.	Capacidad de consultar avisos generales y específicos.	NO.
Registro de estado de ánimo.	Capacidad para expresar sentimientos de un alumno y que se registren en el sistema.	NO.
Consulta de tareas.	Capacidad de consultar tareas.	NO.
Planteamiento de dudas.	Capacidad del alumno para expresar sus dudas de exámenes, tareas, etc.	NO.
Consulta del rendimiento.	Permite conocer el progreso general y por cada clase.	NO.
Envió y recepción de mensajes y archivos. *	Capacidad para enviar y recibir mensajes junto con archivos hacia y desde diferentes grupos de usuarios.	NO.
Envió de tareas. *	Permite entregar las tareas a los profesores.	NO.
Consulta de retroalimentación de tareas.	Permite ver la retroalimentación de los profesores acerca de las tareas entregadas.	NO.
Configuración de cuenta. *	Permite cambiar la contraseña de usuario.	NO.
(*) Acción no realizada por medio del Chatbot.		

Cuadro 4.6: Tabla de comparativa de versiones para el rol de Alumno.

Capítulo 5

Validación

En el capítulo son presentadas las pruebas y validaciones de todos los módulos del sistema e *intents* del *chatbot*, con el fin de validar el funcionamiento del sistema bajo diferentes condiciones, e.g., entradas de datos diversas, diferentes navegadores Web y tamaños variados de pantalla. El capítulo se divide en dos secciones: la primera valida el funcionamiento del *chatbot* probando todos los *intents* que le fueron definidos (cf. Sección 5.1), mostrando adémas ejemplos acompañados de figuras que dan detalles de algunas pruebas realizadas, por razones de espacio no se exponen todas las pruebas del sistema. La segunda parte permite validar, mediante pruebas unitarias, todos los módulos que conforman el sistema, para por último, validar el sistema usando diferentes navegadores Web y tamaños de pantalla (cf. Sección 5.2).

5.1 Validación del *chatbot*

Se efectuó la validación de cada *intent* que conforma el *chatbot* en dos fases:

1. Validación individual en el momento en que se terminó su desarrollo en su versión inicial. Se validó el correcto funcionamiento del *intent* ingresando texto similar a las frases de entrenamiento definidas, ver Figura 5.1, verificando:

    - Que el *intent* se detectaba correctamente.
    - Que se reconociera y se pudiésen extraer datos del contexto.
    - Que, en caso de que por medio del *intent* se realizaran operaciones CRUD a la base de datos, éstas se realicen correctamente.
    - Que solo se permitiera realizar operaciones a los usuarios autorizados.
    - Que la respuesta devuelta sea una de las que fueron definidas previamente.

- Que, en caso contar con un contexto de salida, éste contuviera todos los datos correctos.

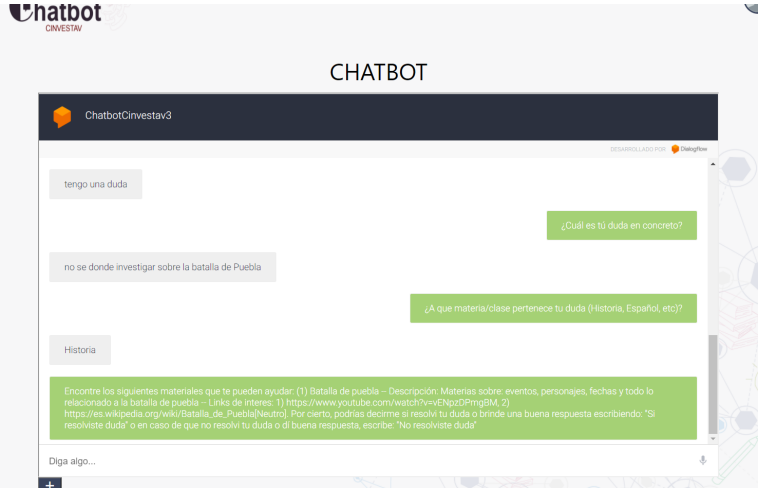

Figura 5.1: Módulo de *Chatbot* probado mediante el uso de texto.

Una vez validados los puntos anteriores, se verifica si el *intent* cumple con su objetivo, ver tabla B.1 en el Apéndice B. En caso afirmativo se le asigna el estatus de "validado", dándose por terminadas las validaciones de ese *intent*. En caso negativo, se registran las deficiencias y/o errores detectados en tablas de registro de errores, ver tabla 5.1. Las deficiencias se mantienen en la tabla hasta el momento en que el objetivo del *intent* y los puntos enlistados son cumplidos, marcando como "solucionado" el estatus en la tabla de errores y como "validado" el estatus del *intent*.

2. Validación general de la interacción con el *chatbot*, se validó la correcta interacción para un alumno y profesor registrados. La forma de verificar la correctitud, es através del cumplimiento de las tareas que buscaba realizar el usuario mediante una conversación con el *chatbot*.

Campo	Descripción del campo.
Problema	Título del problema.
Intent	Nombre del *intent* en donde radica el problema.
Precondiciones	Condiciones que se necesitan para poder generar el escenario donde se produjo el error.
Pasos	Descripción de los pasos de cómo se genera el problema. (Teniendo ->pasa ->cuando).
Resultado Esperado	Cómo debería funcionar correctamente en el *intent*.
Severidad	**Baja** (error cosmético o visual), **media** (funciona, pero es poco práctico, se debe solucionar), **alta** (notorio, se debe solucionar lo antes posible), **critica** (vital, puede llevar al colapso del sistema, de alta prioridad su solución).
Estatus	**Solucionado** (el error ha sido eliminado), **en proceso** (se está buscando una solución), **no solucionado** (de momento no se ha podido solucionar).

Cuadro 5.1: Estructura de la tabla para el registro de errores en los *intents*.

5.1.1 Ejemplo de validación de *intents*

Los siguientes son dos ejemplos de la validación de *intents*, el primero "Añadir-examen" permite verificar que por medio del *chatbot* se realizan las validaciones pertinentes y los profesores pueden añadir exámenes a determinados grados y grupos, el segundo "Consultar-examen", verifica que los alumnos y profesores puedan consultar los exámenes próximos que les corresponde.

Validando el *intent* Añadir-examen:

Para la validación de "Añadir-examen", se procedió a verificar que el *intent* era detectado en caso de ser ingresada una frase en texto similar a sus frases de entrenamiento definidas, ver Figura 5.2.

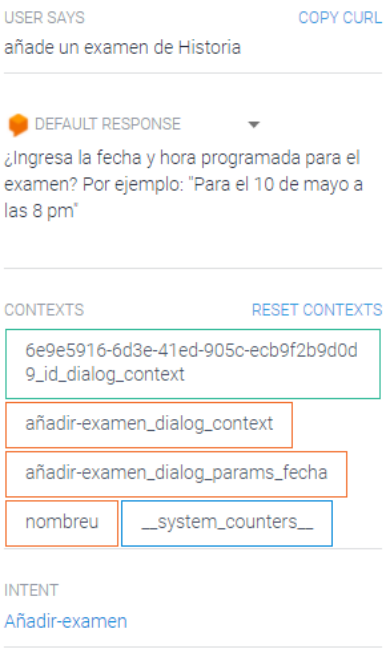

Figura 5.2: Ingresando una frase similar a las definidas en el entrenamiento de "Añadir-examen", detectada correctamente por dicho *intent*.

Seguido, ver Figura 5.3, se verificaba que se detecten y extraigan datos correctamente de los posibles contextos de entrada que pudiese tener el *intent*.

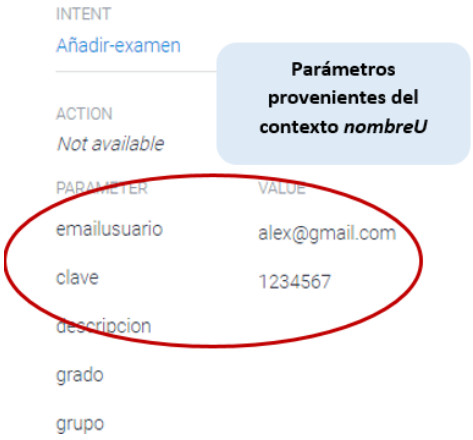

Figura 5.3: Datos provenientes del contexto de entrada `nombreU`.

Posteriormente para la validación del *intent*, se tomaron en cuenta los siguientes casos como posibles a presentarse al momento de la ejecución:

- Que el usuario que desea añadir el examen no está registrado en el sistema.

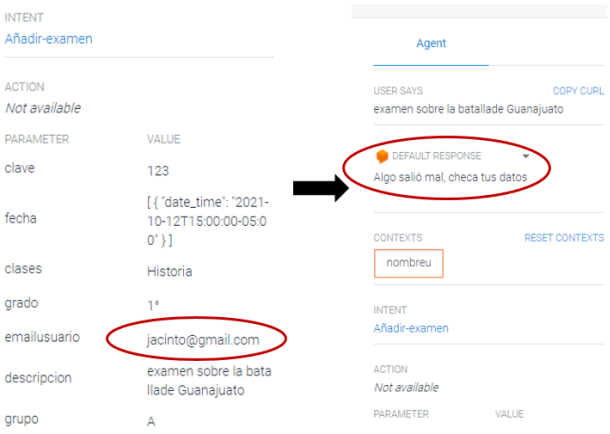

Figura 5.4: Situación cuando un usuario no registrado desea añadir un examen.

- Que el usuario al querer añadir el examen, no tiene el rol de profesor, ver Figura 5.5.

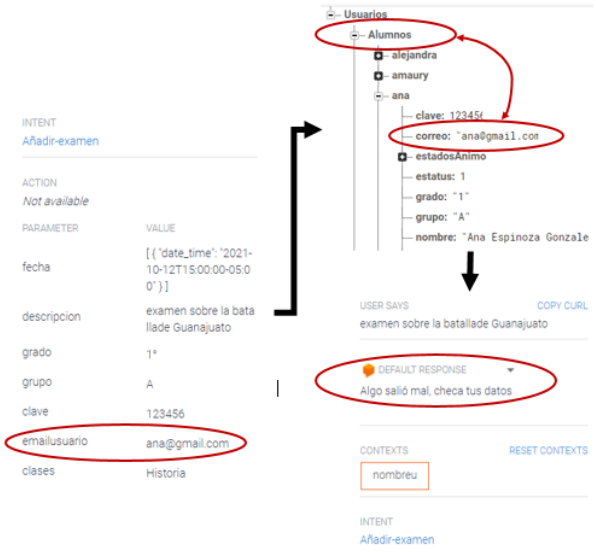

Figura 5.5: Situación cuando un usuario registrado como alumno desea añadir un examen.

- Que el profesor que quiere agregar el examen, no imparte la clase para la cual desea añadirlo o bien la clase a la que le quiere añadir el examen no existe en el grado y grupo seleccionados.

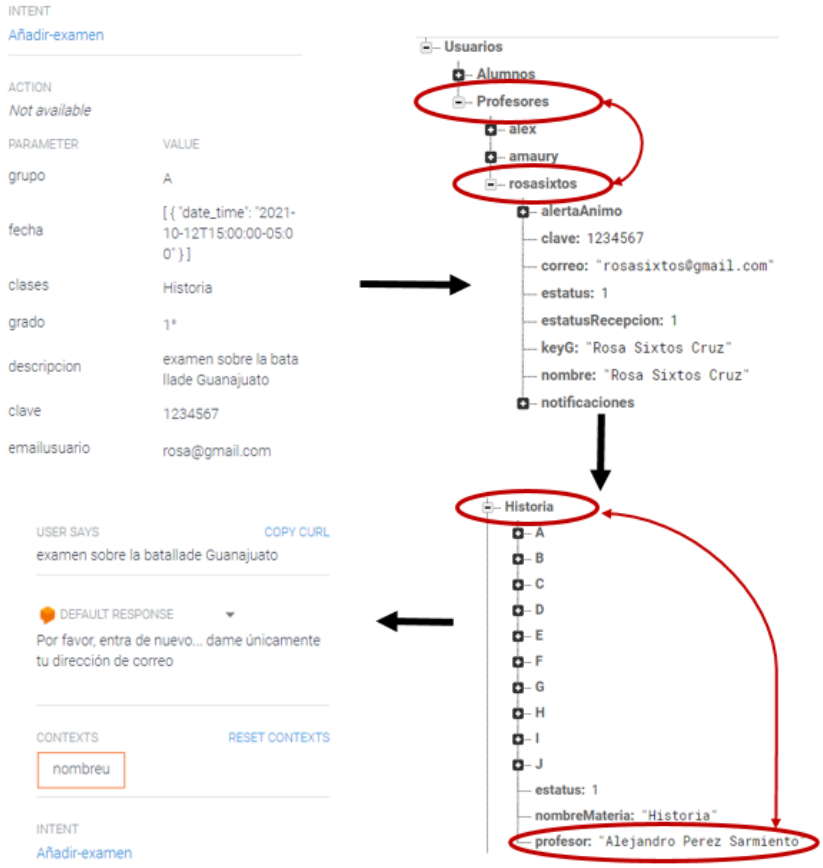

Figura 5.6: Situación cuando un profesor añade un examen en alguna clase que no imparte actualmente.

- Que los campos no acepten cadenas de texto vacías o inválidas, como pudieran ser grados inexistentes (e.g. cuarto, quinto, etc).

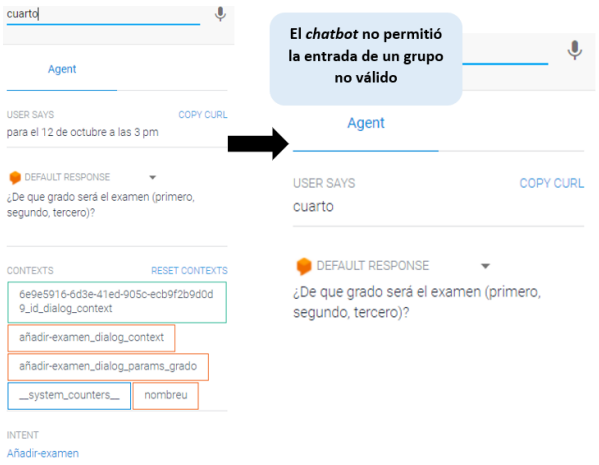

Figura 5.7: Situación presente cuando un profesor quiere ingresa un grado inexistente.

A la par que se llevan a cabo las verificaciones arriba mostradas, se comprueba que el *chatbot* devuelva alguna de las respuestas previamente definidas y que el resultado de las operaciones se vea reflejado en la base de datos.

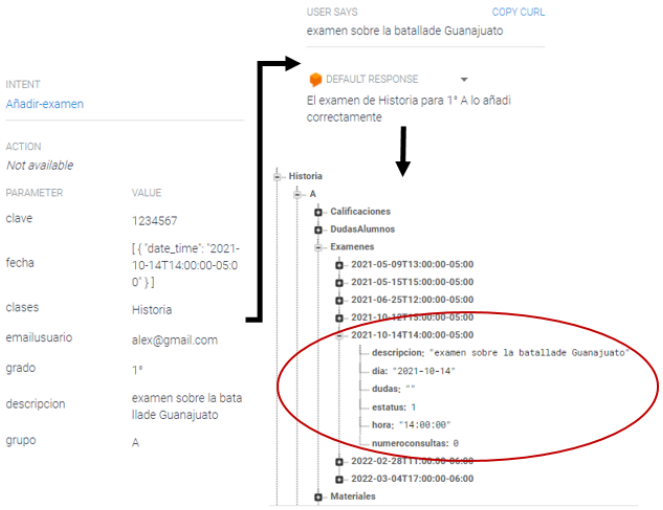

Figura 5.8: *Intent* ejecutado correctamente, se reflejan los resultados en la base de datos.

Con las pruebas realizadas, se compara el resultado del *intent* con su objetivo principal, aquí se muestra el ejemplo para permitir que los profesores añadan exámenes, por lo cual se verifica que solo los profesores pudieran añadir correctamente sus exámenes.

Validando el *intent* Consultar-examen:

En caso de la validación de "Consultar-examen", se verifica que el *intent* sea detectado en caso de ingresar una frase similar a las que se tienen almacenadas como frases de entrenamiento, por ejemplo: ver Figura 5.9.

Figura 5.9: Insertando una frase similar a las del entrenamiento de "Consultar-examen", detectada correctamente.

En la Figura 5.10, se verifica que se detecte y extraigan los datos de los contextos entrantes del *intent*.

Figura 5.10: Datos provenientes del contexto de entrada llamado `nombreU`.

Se verificaron como posibles casos a presentarse los siguientes puntos:

- Que el usuario que desea consultar sus exámenes no está registrado en el sistema, ver Figura 5.11.

Figura 5.11: Caso cuando un usuario no registrado, desea consultar exámenes.

- Que el usuario, que quiere consultar los exámenes de determinada clase, no imparte o toma la clase, además casos en los cuales la clase no está registrada en el sistema. Ver Figura 5.12.

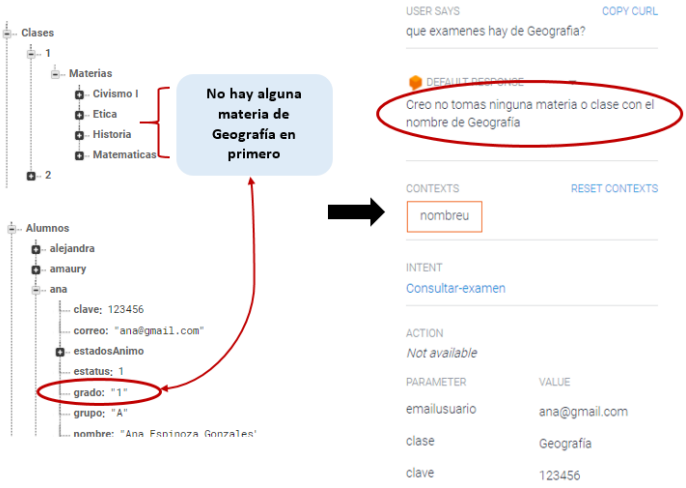

Figura 5.12: Situación cuando un usuario desea consultar los exámenes de cierta clase que no imparte o a la que no asiste actualmente.

Por último, ver Figura 5.13, se comprueba que el *chatbot* devuelva una respuesta entre las que tiene definidas. Por lo que se tiene que el resultado del *intent* con su objetivo principal coincide, siendo para este caso el objetivo: “Permite a los profesores y alumnos consultar sus exámenes”.

Figura 5.13: Respuesta correcta del *intent* para la consulta de exámenes.

5.2 Validación del sistema en *Angular*

Para la validación del sistema desarrollado en *Angular* se hizo uso de pruebas unitarias, las cuales fueron aplicadas a los diferentes módulos del sistema. Las pruebas se documentaron usando tablas similares a la tabla 5.2. Cuando una unidad cumple con todas sus pruebas se le asigna el estatus de "validada" y se dan por terminadas sus pruebas unitarias. En caso de que una unidad no apruebe todas sus pruebas, se registra en tablas de errores, ver tabla 5.3.

Prueba unitaria #	
Nombre	Título de la prueba unitaria.
Descripción	Descripción general de la prueba unitaria.
Objetivos	Objetivos que se buscan con la realización de la prueba unitaria.
Funciones/Métodos implicados	Nombre de los métodos y funciones en código que se ven implicados en la prueba unitaria.
Condiciones	Que se necesita para poder generar este escenario de prueba.
Resultado obtenido	Resultado obtenido al terminar la prueba.
Resultado esperado	Resultado que se espera obtener con la prueba.
Estatus	**Aprobada** (la prueba ha sido validada como correcta), **no aprobada** (la prueba ha resultado con errores), **en espera** (de momento no se ha realizado la prueba).

Cuadro 5.2: Estructura de una tabla para el registro de una prueba unitaria.

Columna	Descripción de la columna.
Problema	Título del problema (bug).
Prueba unitaria	Nombre de la prueba unitaria donde surgió el problema.
Pasos	Descripción de cómo se genera el problema (teniendo -> pasa ->cuando). Brinda también detalles del problema.
Severidad	**Baja** (cosmética, visual), **media** (funciona pero es poco práctica, se debe solucionar), **alta** (notoria, se debe solucionar lo antes posible), **crítica** (vital, puede llevar al colapso del sistema, es de alta prioridad su solución).
Estatus	**Solucionado** (el error ha sido eliminado), **en proceso** (se está buscando una solución), **no solucionado** (de momento no se ha podido solucionar).

Cuadro 5.3: Estructura de la tabla para el registro de errores en una prueba unitaria.

En la tabla C.1 del Apéndice C, se resumen las pruebas unitarias realizadas a cada módulo que conforma el sistema, la tabla tiene los siguiente campos:

- Nombre: para la identificación por nombre de la prueba unitaria.
- Descripción: descripción resumida de la prueba unitaria.
- Módulo: nombre del módulo en el sistema, al cual se le fue aplicada la prueba unitaria.

La validación de la ejecución del sistema web construido con *Angular* se realizó por medio de diferentes navegadores web como: *Google Chrome*, *Mozilla Firefox* y *Microsoft Edge*, validando con ello que su interfaz y funcionamiento se adaptase a dichos navegadores.

Para los distintos tamaños de pantalla que se pudiésen presentar, se efectuaron pruebas de responsividad. Usando para ello la herramienta de *Toogle device toolbar* contenida por omisión en el navegador *Google Chrome* (ver Figura 5.14). La herramienta permite visualizar la ejecución de una página o sistema Web en diferentes tamaños de pantalla y orientaciones. Los tamaños se pueden definir en base a diferentes dispositivos disponibles en la herramienta, los cuales hacen alusión a dispositivos comerciales comunes, e.g, *iPhone 4/5/6/7/8/X*, *iPad*, *iPad Pro*, *Galaxy Fold*, *Nokia N9*, *Moto G4*, *Galaxy Note II/III*, *BlackBerry Z30*, etc. Sin embargo, el tamaño del dispositivo puede ingresarse manualmente. Los tamaños se establecen por la medida absoluta de *point*, la cual es una unidad física de longitud, siendo la equivalencia entre un *pixel* (px) y un punto de: $1\ px = 0.75\ pt$.

Para las pruebas del sistema en *Angular*, se definieron tres tipos de pantalla, pensando en que los dispositivos más usados por los usuarios se pueden dividir en tres grupos: computadoras

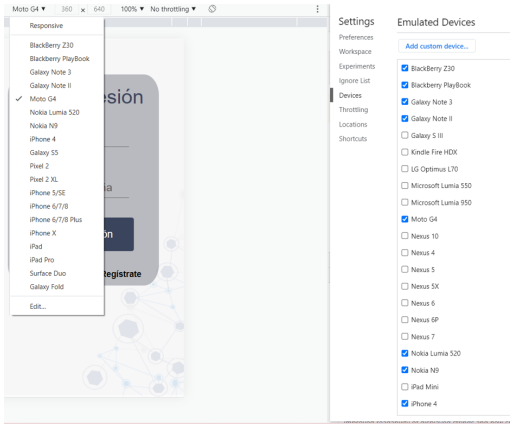

Figura 5.14: Herramienta para la evaluación de responsividad en sitios *Toogle device toolbar* disponible en *Google Chrome*.

(de escritorio y laptops), tabletas y teléfonos celulares. Fueron usados los tamaños promedio de 1,920 *pt* X 1080 *pt* para computadores, de 768 *pt* X 1024 *pt* para tabletas y 375 *pt* X 812 *pt* para teléfonos móviles. El sistema se probó de forma física en la pantalla de una computadora portátil de modelo y marca *ASUS ROG G531GT*, mientras de forma simulada fue probado en pantallas de los dispositivos: *iPhone X*, *iPad*, *Moto G4*, *Nokia N9* y *Galaxy S5*. Puede verse en las Figuras 5.15, 5.16 y 5.17 del módulo de "Configuración de Cuenta" en tres dispositivos diferentes. En las Figuras 5.18, 5.19 y 5.20 se presenta la ventana del módulo "Administración de tareas" para los mismos tipos de dispositivos. En las figuras se pueden apreciar los diferentes tamaños de pantalla evaluados para un mismo módulo del sistema.

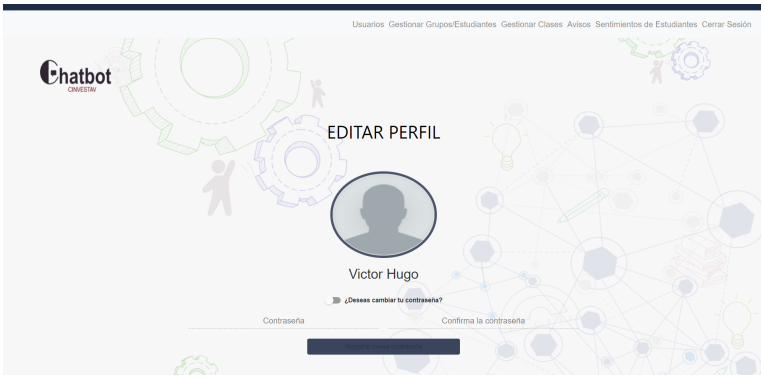

Figura 5.15: Visualización del módulo de 'Configuración de cuenta" en tamaño de pantalla de laptop *ASUS ROG G531GT*.

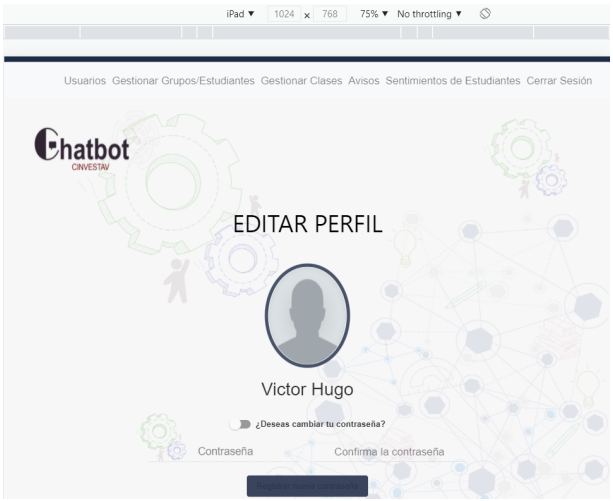

Figura 5.16: Visualización del módulo de 'Configuración de cuenta" en tamaño de pantalla de una tableta *iPad*.

Figura 5.17: Visualización del módulo de 'Configuración de cuenta" en tamaño de pantalla de un télefono celular *Moto G4*.

Figura 5.18: Visualización del módulo de "*Administración de tareas*" en tamaño de pantalla de laptop *ASUS ROG G531GT*.

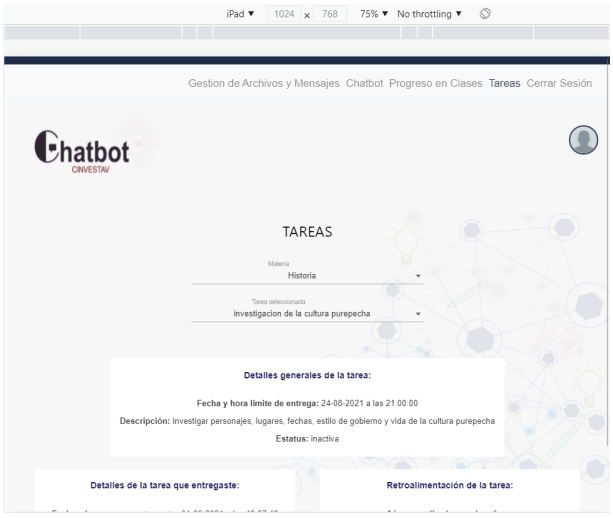

Figura 5.19: Visualización del módulo de"*Administración de tareas*" en tamaño de pantalla de una tableta *iPad*.

Figura 5.20: Visualización del módulo de "*Administración de tareas*" en tamaño de pantalla de un télefono celular *Moto G4*.

Capítulo 6

Conclusiones y trabajo futuro

En este último capítulo, se presentan las conclusiones finales que dejo el trabajo de tesis (cf. Sección 6.1), además de posibles trabajos a futuro que pudiése realizarse para enriquecer lo realizado hasta ahora (cf. Sección 6.2).

6.1 Conclusiones

Actualmente las tecnologías se vinculan con gran parte de las actividades diarias que mueven a la sociedad, gracias a ellas que se han modificado las formas y tiempos en que se realizan actividades cotidianas. Además, la crisis sanitaria actual ha dejado en evidencia la necesidad de que muchas de las actividades diarias, se tornen hacia nuevos ambientes, en donde la tecnología juegue un papel más relevante, en este sentido, la educación se ha vuelto virtual a través de la comunicación a distancia.

Esta situación ha provocado que la tecnología se consolide como el puente que conecta a todos los actores implicados en la educación a distancia. Por otra parte, es importante destacar el incremento del uso de tecnologías de *chatbots*. Estas herramientas por medio de interacciones, regularmente por canales de texto, pueden mantener conversaciones con usuarios. Gracias a dicha conversación, el chatbot puede asistir fluidamente al usuario en algunas sus tareas cotidianas.

Por ello se presentó a lo largo de este desarrollo el análisis y la posterior construcción de un sistema enfocado a dar soporte al proceso enseñanza aprendizaje en escuelas a nivel secundaria en México. El sistema propuesto, por medio de servicios externos de *Google*, y diferentes módulos implementados principalmente en *Angular*, puede ayudar en las labores diarias de una escuela.

En esta propuesta se consideraron tres roles de usuarios: administrador, profesor y alumno, donde cada uno tiene su propia interfaz, y sus operaciones correspondientes definidas en componentes. Además se utilizan dos bases de datos, una base de datos relacional en *MySQL* para la autenticación y registro de usuarios y una base de datos *NoSQL*, para los datos propios del sistema y a los que puede acceder el *chatbot*. Este *chatbot* está nutrido de *intents*, recordando que un *intent* o intención sirve para identificar la intención de los usuarios durante una conversación. Por medio de la clasificación de *intents*, se puede procesar la entrada, realizar operaciones definidas para el *intent* y devolver una salida. El *chatbot* fue alojado en un módulo del sistema que permite acceso a los usuarios profesores y alumnos y por el cual pueden mantener conversaciones. A través de las conversaciones se facilita la realización de algunas de las acciones relacionadas a su entorno educativo, como: consultar exámenes, calificaciones, tareas, materiales, preguntar sobre cuestiones académicas y dudas de sus trabajos, etc.

Algunos puntos a destacar son que el sistema permite recomendar material ante las dudas de los estudiantes relacionadas a sus tareas, exámenes o preguntas de índole general. Para realizar la recomendación se usa un algoritmo que analiza los materiales extra-clase subidos por el profesor y los compara con el texto de las dudas ingresadas por los alumnos. También se implementó un mecanismo para la detección de estados de ánimo por medio de una conversación entre el *chatbot* y el alumno, en donde el *chatbot* es capaz de detectar diferentes tipos de sentimientos. Los sentimientos de peligro como son asesinato, suicidio o depresión se registran, para posteriormente aconsejar al estudiante y alertar sus profesores de una situación de precaución. El sistema permite igualmente llevar un control del rendimiento de los alumnos en base a sus calificaciones, los alumnos y profesores, pueden recibir consejos y un resumen general para que puedan ver sus puntos de deficiencia y mejora.

Todos los módulos del sistema e *intents* que nutren al *chatbot*, fueron sometidos a pruebas de funcionalidad, de manera que se buscó garantizar que el sistema funcionará sin errores tanto estéticos, por ejemplo los que pudiésen ser generados por diferentes tamaños de pantalla, como errores críticos resultados de errores en las bases de datos, sus conexiones, etc.

Sin embargo, debido al tiempo que tomó el análisis y desarrollo del sistema, no se implementaron pruebas de usabilidad con usuarios finales. Con esto se fortalecería el sistema y se garantizaría que el sistema funcione correctamente. Y como consecuencia se podrían hacer evaluaciones de usabilidad y utilidad del *chatbot*. Otro punto del cual carece el sistema y que estuvo fuera de los alcances de esta tesis, es la integración del módulo de *chatbot* con redes sociales, lo que traería un aumento en el uso del sistema, debido a que gran parte de los usuarios finales hace uso cotidiano de algún tipo de red social.

Con todo lo anterior, se tiene una herramienta que puede apoyar las actividades que se

desarrollan en el ámbito escolar a nivel secundaria. Esta propuesta puede servir desde dos trincheras: la primera, e intención final de esta propuesta, es que se pueda llevar a ambientes reales para que sirva como una herramienta más de soporte entre profesores y alumnos. La segunda trinchera, es que sirva como una herramienta para el análisis de cómo las tecnologías, en especial el uso de *chatbots*, pueden impactar en la educación de los estudiantes y el trabajo de los profesores.

6.2 Trabajo Futuro

El trabajo arrojó diferentes aspectos de oportunidad y necesidades de mejora que se pudiésen explorar en trabajos futuros, entre los puntos destacan:

- Integrar el *chatbot* con redes sociales como *Facebook Messenger* y *WhatsApp*.
- Implementar diferentes avatares y personalidades para el agente de *chatbot* del sistema.
- Optimizar partes del código para las funciones contenidas en el proyecto *firebaseFunctions*.
- Crear mayor número de *intents*, contextos y entidades para el *chatbot*.
- Dejar de utilizar la base de datos relacional para la autenticación de usuarios, migrando dicha parte a la base de datos *NoSQL* de *Realtime Database*.
- Realizar pruebas más extensas y exhaustivas de usabilidad del sistema.
- Mejorar el diseño para que el *chatbot* sea fácilmente configurable y por lo tanto usado en ambientes diversos.
- Integrar el *chatbot* en ambientes de aulas inteligentes.
- Agregar diversas modalidades de interacción.

Apéndice A

Tabla de *intents* para la versión 2

En la tabla A.1 son presentados los *intents* generados para el *chatbot* del proyecto de la tesis, la primera columna contiene el nombre del *intent*, la segunda los roles que pueden hacer uso de el y la tercera columna contiene una descripción breve de para que funciona el *intent*.

***Intents* para el Sistema en su Segunda Versión**		
Intent	**Usuarios**	**Descripción**
Acceder-nuevamente.	Alumnos Profesores.	Permite que un alumno o profesor pueda ser reconocido por el Chatbot para comenzar con la interacción.
Actualizar-estado-retroalimentacion-dudas.	Alumnos.	Sirve para que los alumnos indiquen que ya vieron la retroalimentación de sus dudas, a fin de que el chatbot no les recuerde que tienen nuevas retroalimentaciones, a menos que un profesor vuelva a registrar alguna retroalimentación.
Atender-urgencia.	Profesores.	Sirve para que los profesores puedan indicar que ya se han atendido las urgencias de alumnos que se tenían activas, con el fin de que no se le muestre que debe atenderlas.
Añadir-examen.	Profesores.	Permite a los profesores añadir exámenes a determinados grados, grupos y materias.
Añadir-material-clase.	Profesores.	Permite a los profesores añadir materiales de apoyo a determinados grados, grupos y materias.
Añadir-recordatorio-clase.	Profesores.	Permite a los profesores añadir recordatorios de clases (avisos) a determinados grados, grupos y materias.
Añadir-tarea.	Profesores.	Permite a los profesores añadir tareas para determinados grados, grupos y materias.

***Intents* para el Sistema en su Segunda Versión**		
Consulta-nombre-profesor.	Alumnos.	Consulta para conocer el nombre de un profesor que le imparte determinada clase.
Consultar-avisos.	Alumnos Profesores.	Permite consultar los avisos generales. Además de los recordatorios de clases (en caso de que sea consulta de un alumno).
Consultar-calificacion.	Alumnos.	Permite a los estudiantes consultar las calificaciones que tienen en determinada materia.
Consultar-cantidad-consultas-examenes -materiales.	Profesores.	Permite que los profesores consulten qué tan buscado o recomendado ha sido un determinado material para resolver dudas sobre exámenes.
Consultar-cantidad-consultas-generales-materiales.	Profesores.	Permite que los profesores consulten qué tan buscado o recomendado ha sido un determinado material para resolver dudas generales.
Consultar-cantidad-consultas-materiales.	Profesores.	Permite que los profesores consulten qué tan buscado o recomendado ha sido un determinado material.
Consultar-cantidad-consultas-tareas-materiales.	Profesores.	Permite que los profesores consulten qué tan buscado o recomendado ha sido un determinado material para resolver dudas de tareas.
Consultar-dudas-alumno.	Profesores.	Para consultar todas las dudas que ha realizado determinado alumno y poder observar sus detalles y estatus.
Consultar-dudas-resueltas-examen.	Profesores.	Permite que un profesor pueda consultar las dudas de los exámenes, que han sido resueltas correctamente (a percepción de los mismos alumnos).
Consultar-dudas-resueltas-general.	Profesores.	Permite que un profesor pueda consultar las dudas generales que han sido resueltas correctamente (a percepción de los mismos alumnos).
Consultar-dudas-resueltas-tareas.	Profesores.	Permite que un profesor pueda consultar las dudas de las tareas que han sido resueltas correctamente (a percepción de los mismos alumnos).
Consultar-dudas-sin-resolver-examen.	Profesores.	Permite que un profesor pueda consultar las dudas de los exámenes que no han sido resueltas (a percepción de los mismos alumnos).
Consultar-dudas-sin-resolver-general.	Profesores.	Permite que un profesor pueda consultar las dudas generales que no han sido resueltas (a percepción de los mismos alumnos).

***Intents* para el Sistema en su Segunda Versión**		
Consultar-dudas-sin-resolver-tareas.	Profesores.	Permite que un profesor pueda consultar las dudas de las tareas que no han sido resueltas (a percepción de los mismos alumnos).
Consultar-estatus-usuario.	Alumnos Profesores.	Permite que los profesores y alumnos puedan saber si su cuenta está activa.
Consultar-examen.	Alumnos Profesores.	Permite que un profesor o alumno consulte los exámenes que tiene programados y próximos a efectuarse.
Consultar-examen-con-mas-falsas-positivas.	Profesores.	Permite consultar el examen que ha tenido mayor número de respuestas falsas positivas a las dudas de los alumnos (Según la percepción de ellos).
Consultar-examen-mas-solicitado.	Profesores.	Permite conocer cuál ha sido el examen por el que han preguntado o buscado más los alumnos.
Consultar-examen-mas-solicitado-grado-grupo.	Profesores.	Permite conocer el examen por determinado grado y grupo que ha sido más consultado o buscado por los alumnos.
Consultar-falsas-positivas-examenes.	Profesores.	Permite consultar las respuestas falsas positivas a las dudas de los alumnos (Según la percepción de ellos), en los exámenes.
Consultar-falsas-positivas-generales.	Profesores.	Permite consultar las respuestas falsas positivas a las dudas generales de los alumnos (Según la percepción de ellos).
Consultar-falsas-positivas-tareas.	Profesores.	Permite consultar las respuestas falsas positivas a las dudas y consultas de tareas de los alumnos (Según la percepción de ellos).
Consultar-material.	Alumnos Profesores.	Permite a los profesores y alumnos consultar los materiales que se hayan subido a determinada clase.
Consultar-material-especifico.	Alumnos.	Permite a los alumnos consultar algún material en especifico que se haya subido a determinada clase.
Consultar-materias-actuales.	Alumnos Profesores.	Permite a los alumnos y profesores consultar las materias que toman o imparten actualmente.
Consultar-nombres-alumnos.	Profesores.	Da la capacidad a los profesores de consultar el nombre de sus alumnos por clase, grado y grupo.
Consultar-nombres-materiales.	Alumnos Profesores.	Permite conocer el nombre de los materiales que se encuentran registrados para determinada clase.
Consultar-recomendacion-correcta-material.	Profesores.	Da la posibilidad de que los profesores consulten qué tan bien ha sido recomendado determinado material de apoyo.

***Intents* para el Sistema en su Segunda Versión**		
Consultar-recomendacion-incorrecta-material.	Profesores.	Da la posibilidad de que los profesores consulten qué tan mal ha sido recomendado determinado material de apoyo.
Consultar-recordatorio-clase.	Alumnos Profesores.	Permite que los profesores y alumnos consulten los recordatorios que tengan de sus clases.
Consultar-rendimiento-alumnos.	Profesores.	Permite a los profesores consultar el rendimiento de sus alumnos inscritos en determinada clase, grado y grupo.
Consultar-retroalimentacion-dudas-alumno.	Alumnos.	Permite que los alumnos puedan consultar la retroalimentación a dudas que han planteado anteriormente.
Consultar-tarea.	Alumnos.	Permite que los alumnos consulten las tareas que tienen, así como el estatus de las mismas.
Consultar-tarea-con-mas-falsas-positivas.	Profesores.	Permite a los profesores consultar cuál ha sido la tarea con mayor cantidad de respuestas falsas positivas, según la clase, grado y grupo.
Consultar-tarea-hoy.	Alumnos.	Los alumnos pueden consultar si hay alguna tarea que deban entregar a mas tardar el día en que hagan su consulta.
Consultar-tarea-hoy-maestro.	Profesores.	Los profesores pueden consultar si hay alguna tarea que deban entregar sus alumnos a mas tardar el día en que hagan su consulta.
Consultar-tarea-pendientes.	Alumnos.	Los alumnos pueden consultar las tareas pendientes por enviar.
Consultar-tareas-entregadas.	Profesores.	Permite a los profesores consultar las tareas que le han entregado en cierto grado, grupo y materia.
Consultar-tareas-no-entregadas.	Profesores.	Permite a los profesores consultar las tareas que no le han entregado en cierto grado, grupo y materia.
Consultar-titulos-tareas.	Alumnos Profesores.	Permite que los profesores y alumnos puedan consultar la lista y nombres de sus tareas.
Consultar-urgencia-individual.	Profesores.	Permite que los profesores puedan consultar si tienen o han tenido urgencias de determinado alumno.
Consultar-urgencias.	Profesores.	Permite que los profesores conozcan si hay alguna urgencia de alumnos por atender.
Conversar-casualmente-inicio.	Alumnos Profesores.	Permite conversar casualmente a los alumnos y profesores con el *chatbot*.

***Intents* para el Sistema en su Segunda Versión**		
Conversar-casualmente-sentimiento.	Alumnos.	Permite detectar sentimientos y alertas de los estudiantes, previniendo a profesores en caso de detectar una situación de peligro en algún alumno.
Dar-consejo-urgencia-individual.	Profesores.	Permite que los profesores puedan dar consejos a un determinado alumno que lo necesite.
Default Fallback Intent.	Alumnos Profesores.	Se activa en caso de que no se encuentre un *intent* coincidente.
Default Welcome Intent.	Alumnos Profesores.	Permite que los alumnos y profesores puedan iniciar una conversación con el *Chatbot* y este último los pueda identificar.
Dudas-generales-alumnos.	Alumnos.	Permite que los alumnos puedan expresar sus dudas generales y el *chatbot* pueda responder con recomendaciones a los alumnos.
Dudas-tarea.	Alumnos.	Permite que los alumnos puedan expresar sus dudas acerca de tareas y el *chatbot* pueda responder con recomendaciones a los alumnos.
Groserias.	Alumnos Profesores.	Se activa en caso de que se detecten palabras antisonantes.
Preguntas-acerca-examen.	Alumnos.	Permite que los alumnos puedan expresar sus dudas acerca de exámenes y el *chatbot* pueda responder con recomendaciones a los alumnos.
Preguntas-cursos.	Alumnos Profesores.	Permite a los profesores y alumnos consultar acerca de cuándo empiezan o terminan sus cursos.
Preguntas-inscripcion.	Alumnos Profesores.	Permite a los profesores y alumnos preguntar acerca de inscripciones o reinscripciones sus cursos.
Quitar-consejos-animo-profesores.	Alumnos.	Permite que los alumnos puedan desactivar la visibilidad de los consejos que los profesores les hayan enviado anteriormente.
Resolucion-duda-examen.	Alumnos.	Permite que los alumnos indiquen si su duda en relación con un examen ha sido resuelta o no.
Resolucion-duda-general.	Alumnos.	Permite que los alumnos indiquen si su duda general ha sido resuelta o no.
Resolucion-duda-tarea.	Alumnos.	Permite que los alumnos indiquen si su duda en relación con una tarea ha sido resuelta o no.

***Intents* para el Sistema en su Segunda Versión**		
Resolver-urgencia-individual.	Profesores.	Da la capacidad a los profesores de indicar que han atendido todas las urgencias de alumnos que tenía, con ello no se les mostraran urgencias a menos que surja una nueva.
Tareas-enviadas-tarde-estudiante.	Alumnos.	Permite que un alumno pueda consultar las tareas que ha enviado tarde.
Ver-examen-mas-consultado.	Profesores.	Permite que un profesor pueda consultar los exámenes que han sido mas buscados o han generado más preguntas a sus alumnos de determinada clase.
Ver-examenes-consultados.	Profesores.	Permite que un profesor pueda consultar la información estadística de las consultas y búsquedas hechas de los alumnos a sus exámenes en determinada clase.
Ver-material-mas-consultado.	Profesores.	Permite a los profesores conocer qué material ha sido más recomendado o buscado.
Ver-materiales-consultados.	Profesores.	Permite que un profesor pueda consultar la información estadística de las consultas y búsquedas hechas de los alumnos a materiales de apoyo de determinada clase.
Ver-tarea-mas-consultada.	Profesores.	Permite a los profesores conocer qué tarea ha sido la que genera mas preguntas o consultas.
Ver-tareas-consultadas.	Profesores.	Permite que un profesor pueda consultar la información estadística de las consultas y búsquedas hechas de los alumnos a tareas de determinada clase.

Cuadro A.1: Tabla de intents implementados en la versión 2.

Apéndice B

Tabla de objetivos por *intent*

En la tabla B.1 se presentan los objetivos de cada *intent* que nutre al *chatbot*, la primera columna contiene el nombre del *intent* y la segunda el objetivo de dicho *intent*. Se usa el objetivo de cada *intent* como punto de referencia para saber si un *intent* logra su cometido, cuando un *intent* se ejecuta tanto, en su parte central como sus operaciones secundarias, sin errores, se dictamina que el *intent* a alcanzado su objetivo.

Intent	**Objetivo**
Acceder-nuevamente.	Identificar si un usuario está registrado en el sistema, en caso positivo obtener su información.
Actualizar-estado-retroalimentacion-dudas.	Que los alumnos indiquen que ya recibieron la retroalimentación a sus dudas.
Atender-urgencia.	Que los profesores indiquen que ya han atendido las urgencias de sus alumnos.
Añadir-examen.	Permitir que los profesores añadan exámenes.
Añadir-material-clase.	Permitir que los profesores añadan material de apoyo a clases.
Añadir-recordatorio-clase.	Permitir que los profesores agreguen recordatorios de clases.
Añadir-tarea.	Permitir que los profesores añadan tareas.
Consulta-conocer.	Permitir al *chatbot* presentarse ante el usuario.
Consulta-nombre-profesor.	El alumno puede consultar el nombre de sus profesores.
Consultar-avisos.	Consultar los avisos registrados.
Consultar-calificacion.	El alumno puede consultar sus calificaciones.

Intent	**Objetivo**
Consultar-cantidad-consultas-examenes-materiales.	El profesor consulta qué tanto ha sido recomendado un material para resolver dudas de exámenes.
Consultar-cantidad-consultas-generales-materiales.	El profesor consulta qué tanto ha sido recomendado un material para resolver dudas de generales.
Consultar-cantidad-consultas-materiales.	El profesor puede consultar la cantidad de veces que ha sido recomendado un material.
Consultar-cantidad-consultas-tareas-materiales.	El profesor consulta qué tanto ha sido recomendado un material para resolver dudas de tareas.
Consultar-dudas-alumno.	El profesor puede consultar todas las dudas que ha tenido cierto alumno.
Consultar-dudas-resueltas-examen.	El profesor consulta las dudas de examen que han sido resueltas correctamente.
Consultar-dudas-resueltas-general.	El profesor consulta las dudas de generales que han sido resueltas correctamente.
Consultar-dudas-resueltas-tareas.	El profesor consulta las dudas de tareas que han sido resueltas correctamente.
Consultar-dudas-sin-resolver-examen.	El profesor consulta las dudas de examen que no han sido resueltas correctamente.
Consultar-dudas-sin-resolver-general.	El profesor consulta las dudas de generales que no han sido resueltas correctamente.
Consultar-dudas-sin-resolver-tareas.	El profesor consulta las dudas de tareas que no han sido resueltas correctamente.
Consultar-estatus-usuario.	El usuario puede consultar si su cuenta existe y está activa.
Consultar-examen.	Permite a los profesores y alumnos consultar sus exámenes.
Consultar-examen-con-mas-falsas-positivas.	Permite a los profesores consultar qué examen tiene mayor número de respuestas falsas positivas para sus alumnos.
Consultar-examen-mas-solicitado.	El profesor pregunta qué examen ha sido más buscado.
Consultar-examen-mas-solicitado-grado-grupo.	El profesor pregunta qué examen ha sido más buscado por determinado grado y grupo.

Intent	**Objetivo**
Consultar-falsas-positivas-examenes.	El profesor pregunta por las respuestas falsas positivas resultantes de dudas sobre exámenes.
Consultar-falsas-positivas-generales.	El profesor pregunta por las respuestas falsas positivas a dudas generales.
Consultar-falsas-positivas-tareas.	Los profesores pueden consultar qué tarea tiene mayor número de respuestas falsas positivas para los alumnos.
Consultar-material.	Los profesores y alumnos pueden consultar los materiales registrados.
Consultar-material-especifico.	Los alumnos consultan algún material en especifico registrado.
Consultar-materias-actuales.	Los profesores y alumnos consultan sus materias actuales.
Consultar-nombres-alumnos.	Los profesores pueden consultar el nombre de sus alumnos.
Consultar-nombres-materiales.	Los profesores y alumnos purden consultar el nombre de los materiales registrados.
Consultar-recomendacion-correcta-material.	Los profesores pueden ver que tanto ha sido recomendado correctamente un material.
Consultar-recomendacion-incorrecta-material.	Los profesores de ver qué tan útil ha sido el material recomendado.
Consultar-recordatorio-clase.	Los profesores y alumnos pueden consultar los recordatorios de clase.
Consultar-rendimiento-alumnos.	Los profesores puede consultar el rendimiento de sus alumnos.
Consultar-retroalimentacion-dudas-alumno.	Los alumnos tienen acceso a la retroalimentación a sus dudas anteriores.
Consultar-tarea.	Los alumnos consultan sus tareas pendientes.
Consultar-tarea-con-mas-falsas-positivas.	El profesor consulta qué tarea ha tenido más dudas con respuestas falsas positivas.
Consultar-tarea-hoy.	Los alumnos pueden saber qué tareas se entregan el mismo día que se realiza la consulta.
Consultar-tarea-hoy-maestro.	Los profesores pueden saber qué tareas deben entregarse el día en que se realiza la consulta.

Intent	**Objetivo**
Consultar-tarea-pendientes.	Los alumnos preguntan por las tareas tienen pendientes por enviar.
Consultar-tareas-entregadas.	El profesor consulta qué tareas le han sido entregadas.
Consultar-tareas-no-entregadas.	El profesor consulta qué tareas no le han sido entregadas.
Consultar-titulos-tareas.	Profesores y alumnos consultan sus tareas asociadas.
Consultar-urgencia-individual.	Los profesores consultan si tienen urgencias por parte de un alumno.
Consultar-urgencias.	Permitir a los profesores saber si hay urgencias por atender.
Conversar-casualmente-inicio.	Brindar una conversación casual entre usuarios y el *chatbot*.
Conversar-casualmente-sentimiento.	Detectar sentimientos y estados de ánimo expresados por alumnos al *chatbot*.
Dar-consejo-urgencia-individual.	Los profesores pueden dar consejo a urgencias de sus alumnos.
Default Fallback Intent.	Detectar casos en donde no se ubicó algún *intent* coincidente a la entrada del usuario.
Default Welcome Intent.	Iniciar la conversación e identificar si el usuario está registrado en el sistema, en caso positivo obtener su información.
Dudas-generales-alumnos.	Permite a los alumnos expresar sus dudas generales.
Dudas-tarea.	Permite a los alumnos expresar sus dudas sobre tareas.
Groserias.	Detectar palabras o frases antisonantes de los usuarios.
Preguntas-acerca-examen.	Los alumnos pueden expresar sus dudas sobre exámenes.
Preguntas-cursos.	Los profesores y alumnos pueden formular preguntas sobre el inicio y fin de cursos.
Preguntas-inscripcion.	Los profesores y alumnos pueden formular preguntas sobre la inscripción y reinscripción a cursos.
Quitar-consejos-animo-profesores.	Los alumnos pueden cambiar la visibilidad de los consejos que les han dado sus profesores. Para evitar que el *chatbot* les recuerde constantemente que tienen consejos dados por los profesores.

Intent	**Objetivo**
Resolucion-duda-examen.	Permitir que los alumnos indiquen si su duda fue resuelta o no. Registrar posibles respuestas falsas positivas en dudas de exámenes.
Resolucion-duda-general.	Permite que los alumnos indiquen si su duda general ha sido resuelta o no. Registrar posibles respuestas falsas positivas a dudas generales.
Resolucion-duda-tarea.	Permite que los alumnos indiquen si su duda relacionada a tareas ha sido resuelta o no. Registrar posibles respuestas falsas positivas a dudas de tareas.
Resolver-urgencia-individual.	El profesor puede indicar que han resuelto las urgencias de cierto alumno.
Tareas-enviadas-tarde-estudiante.	Permite que un alumno visualice las tareas que ha enviado fuera de tiempo.
Ver-examen-mas-consultado.	Permite que un profesor consulte los exámenes por los cuales preguntan más sus alumnos.
Ver-examenes-consultados.	Permite que un profesor consulte información estadística de las dudas a los exámenes hechas por sus alumnos.
Ver-material-mas-consultado.	Permite que un profesor conozca cuales son los materiales más consultados por sus alumnos.
Ver-materiales-consultados.	Permite que un profesor consulte información estadística de los materiales consultados por sus alumnos.
Ver-tarea-mas-consultada.	Permite que un profesor consulte las tareas por los cuales preguntan más sus alumnos.
Ver-tareas-consultadas.	Permite que un profesor consulte información estadística de las dudas de tareas hechas por sus alumnos.

Cuadro B.1: Tabla de objetivos de cada *intent*.

Apéndice C

Pruebas unitarias realizadas en los módulos del sistema

La tabla C.1 se usa para el registro de todas las pruebas unitarias realizadas en cada módulo del sistema, la tabla está compuesta por el nombre de la prueba unitaria y la descripción de dicha prueba.

Nombre de la prueba	**Descripción**
Módulo: *Login*	
Iniciar componente *Login*.	Permite verificar que el componente *Login* ha sido cargado correctamente.
Comprobar autenticación.	Permite verificar si se realiza la autenticación correcta de usuarios registrados en el sistema. Para luego redireccionar al componente *Menu*.
Redirección a *Register*.	Verifica si se redirige correctamente a la ruta de registro de usuarios desde el *Login*.
Módulo: Registro de usuarios	
Iniciar componente *Register*.	Permite verificar que el componente *Register* ha sido cargado correctamente.
Registrar nuevos usuarios.	Permite verificar si se registran nuevos usuarios en el sistema.
No registrar nuevos usuarios.	Permite verificar si no se registran usuarios existentes en el sistema.
Validar campos de registro.	Permite validar que todos los campos del formulario de registro son correctos.

Nombre de la prueba	**Descripción**
Redireccionar a nuevos usuarios.	Verifica que se redirige correctamente a los nuevos usuarios a su pantalla inicial del sistema.
Verificar seguridad de nuevos usuarios.	Verifica que los nuevos usuarios que no están activados, no puedan realizar acciones en el sistema una vez que entran a su pantalla inicial.
Redireccionar a *Login*.	Verifica si se redirige correctamente a la ruta de *Login* desde *Register*.
Módulo: Gestionar usuarios	
Iniciar componente *Validate-users*.	Permite verificar que el componente *Validate-users* ha sido cargado correctamente.
Filtrar usuarios.	Verifica si se realiza la búsqueda de usuarios mediante el filtro del componente *Validate-users*.
Consultar usuarios.	Verifica que los usuarios se muestren correctamente en la tabla de usuarios del componente *Validate-users*.
Confirmar desactivación de usuarios.	Prueba si con el cuadro de diálogo del componente *Confirm-dialog* es confirmado el cambio de estatus, de un usuario, a inactivo.
Cambiar estatus de usuarios.	Permite probar si el cambio de estatus de un usuario se realiza correctamente.
Redireccionar desde *Validate-users*.	Verifica si se redirige correctamente a las rutas definidas en el menú del *header*, contenido en *Validate-users*.
Módulo: Gestión de grupos/estudiantes	
Iniciar componente *User-lessons*.	Permite verificar que el componente *Users-lessons* ha sido cargado correctamente.
Iniciar componente *Add-student*.	Permite verificar que el componente *Add-student* ha sido cargado correctamente y es mostrado en una ventana emergente.
Verificar conexión entre el componente *User-lessons* con *Add-student*.	Ayuda a verificar que existe una conexión y transmisión/recepción de información entre los componentes *User-lessons* y *Add-student*.

Nombre de la prueba	**Descripción**
Filtrar usuarios.	Verifica si se realiza la búsqueda de alumnos activos de cierto grado y grupo mediante el filtro del componente *User-lessons*.
Consulta de usuarios desde *User-lessons*.	Verifica que los alumnos activos pertenecientes a cierto grado y grupo se muestren correctamente en la tabla de usuarios definida en *User-lessons*.
Consulta de usuarios desde *Add-student*.	Verifica que solo los usuarios activos de tipo alumno, sin grado y grupo, se muestren correctamente en la tabla de usuarios definida en *Add-student*.
Añadir alumnos al grado y grupo seleccionado.	Valida si se añaden correctamente los alumnos seleccionados al grado y grupo que se definió.
Confirmación de eliminar alumnos.	Prueba si se realiza con el componente *Confirm-dialog* la verificación de si se desean eliminar alumnos.
Eliminar alumnos.	Permite probar si se eliminan correctamente los alumnos seleccionados, así como su información previa en el grado y grupo al cual pertenecian.
Redireccionar desde *User-lessons*.	Verifica si se redirige correctamente a las rutas definidas en el menú del *header* contenido en *Users-lessons*.
	Módulo: Gestión de clases
Iniciar componente *Class-g*.	Permite verificar que el componente *Class-g* ha sido cargado correctamente.
Iniciar componente *Add-subject*.	Permite verificar que el componente *Add-subject* ha sido cargado correctamente y es mostrado en una ventana emergente.
Verificar conexión entre el componente *Class-g* con *Add-subject*.	Ayuda a verificar que existe una conexión y transmisión/recepción de información entre los componentes *Class-g* y *Add-subject*.
Consultar profesores.	Verifica en el componente si se muestran los profesores activos en el sistema.
Consultar materias activas.	Valida que solo se consulten y muestren las materias activas para el grado seleccionado.

Nombre de la prueba	**Descripción**
Validar formulario de *Add-subject*.	Verifica que el formulario de *Add-subject* esté realizando las validaciones correctas.
Añadir materias.	Valida si se añaden correctamente materias al sistema, con el grado y profesor seleccionado.
Confirmar desactivación de materias.	Prueba si se realiza con el componente *Confirm-dialog* la verificación de si se desea desactivar una materia.
Desactivar materia.	Permite probar si se desactiva correctamente la materia seleccionada, cambiando el estatus de la misma.
Redireccionar desde *Class-g*.	Verifica si se redirige correctamente a las rutas definidas en el menú del *header* contenido en *Class-g*.
Módulo: Administración de avisos generales	
Iniciar componente *Reminders*.	Permite verificar que el componente *Reminders* ha sido cargado correctamente.
Iniciar componente *View-reminder*.	Permite verificar que el componente *View-reminder* ha sido cargado correctamente y es mostrado en una ventana emergente.
Verificar conexión entre el componente *Reminders* con *View-reminder*.	Ayuda a verificar que existe una conexión y transmisión/recepción de información entre los componentes *Reminders* y *View-reminder*.
Consultar avisos activos.	Verifica si se consultan y muestran en una tabla los avisos activos en el sistema.
Consultar avisos inactivos.	Verifica si se consultan y muestran en una tabla los avisos inactivos en el sistema.
Validar formulario de *Reminders*.	Verifica que el formulario de *Reminders* esté realizando las validaciones correctas.
Registrar aviso.	Valida si se agrega correctamente un aviso en el sistema.
Confirmar activación de avisos.	Valida la activación de avisos.
Confirmar desactivación de avisos.	Valida la desactivación de avisos.

Nombre de la prueba	**Descripción**
Activar aviso.	Permite probar si se activa correctamente un aviso, para el cual su fecha límite no supere la fecha actual.
Desactivar aviso.	Permite probar si se desactiva correctamente un aviso activo.
Ver detalles del aviso seleccionado.	Valida que los datos del aviso seleccionado se consultan y muestran correctamente en el componente *View-reminder*.
Redireccionar desde *Reminders*.	Verifica si se redirige correctamente a las rutas definidas en el menú del *header* contenido en el componente *Reminders*.
Módulo: Percepción de estudiantes	
Iniciar componente *Feelings*.	Permite verificar que el componente *Feelings* ha sido cargado correctamente.
Iniciar componente *Feeling-group*.	Permite verificar que el componente *Feeling-group* ha sido cargado correctamente dentro de *Feelings*.
Iniciar componente *Feeling-individual*.	Permite verificar que el componente *Feeling-individual* ha sido cargado correctamente dentro de *Feelings*.
Iniciar componente *Add-advice*.	Permite verificar que el componente *Add-advice* ha sido cargado correctamente y es mostrado en una ventana emergente.
Verificar conexión entre el componente *Feelings* con *Feeling-group*.	Ayuda a verificar que existe una conexión y transmisión/recepción de información entre los componentes *Feelings* y *Feeling-group*.
Verificar conexión entre el componente *Feelings* con *Feeling-individual*.	Ayuda a verificar que existe una conexión y transmisión/recepción de información entre los componentes *Feelings* y *Feeling-individual*.
Verificar conexión entre el componente *Feelings* con *Add-advice*.	Ayuda a verificar que existe una conexión y transmisión/recepción de información entre los componentes *Feelings* y *Add-advice*.

Nombre de la prueba	**Descripción**
Consultar estados de ánimo generales.	Verifica si se consultan y muestran en tabla los estados de ánimo registrados en el sistema para el grado y grupo seleccionados.
Contar alumnos que expresaron estados de ánimo.	Realiza el conteo de alumnos en el grado y grupo seleccionado que expresaron estados de ánimo al sistema.
Conteo alumnos que no expresaron estados de ánimo.	Realiza el conteo de alumnos en el grado y grupo seleccionado que no expresaron estados de ánimo al sistema.
Graficación del conteo de estados de ánimo sí o no expresados.	Verifica que sean graficados los conteos de los alumnos que si expresaron sus estados de ánimo o no, una vez dados su grado y grupo.
Graficación de estados de ánimo general.	Verifica que los conteos de estados de ánimo generales para el grado y grupo seleccionados se grafiquen.
Consultar alumnos del grado y grupo seleccionados.	Valida que se muestran los alumnos activos que pertenecen al grado y grupo seleccionado.
Consultar estados de ánimo individuales.	Consulta una vez seleccionado el nombre del estudiante, sus estados de ánimo registrados.
Visualizar los estados de ánimo individuales.	Verifica que los estados de ánimo del estudiante seleccionado se grafican y muestran en una tabla.
Consultar apoyos registrados.	Verifica si se consultan y muestran los apoyos a estados de ánimo registrados.
Registrar apoyo a estados de ánimo.	Verifica que se puedan añadir apoyos a un estado de ánimo no resuelto y que sea de peligro.
Validar campos del registro de apoyo.	Verifica que el formulario para añadir apoyos se valide correctamente.
Redireccionar desde *Feelings*.	Verifica si se redirige correctamente a las rutas definidas en el menú del *header* contenido en *Feelings*.
Módulo: Configuración de cuenta	
Iniciar componente *Configure-account*.	Permite verificar que el componente *Configure-account* ha sido cargado correctamente.

Nombre de la prueba	**Descripción**
Validar formulario de *Configure-account*.	Verifica que el formulario de *Configure-account* esté realizando las validaciones correctas.
Cambiar contraseña.	Valida si se modifica correctamente la contraseña del usuario actual en el sistema.
Confirmar cambio de contraseña.	Prueba si se realiza con el componente *Confirm-dialog*, la verificación de si se desea cambiar una contraseña.
Confirmar recepción de mensajes.	Prueba para ver si los profesores, desde el componente *Confirm-dialog*, confirman si desean recibir mensajes de alumnos externos a sus clases.
Cambiar estatus de recepción de mensajes.	Verifica si se puede cambiar el estatus de recepción de mensajes para los usuarios profesores.
Redireccionar desde *Configure-account*.	Verifica si se redirige correctamente a las rutas definidas en el menú del *header* contenido en *Configure-account*.
Módulo: *Chatbot*	
Iniciar componente *Chatbot*.	Permite verificar que el componente Chatbot ha sido cargado correctamente.
Iniciar componente *Add-files*.	Permite verificar que el componente *Add-files* ha sido cargado correctamente y es mostrado en una ventana emergente.
Iniciar componente *Add-homework*.	Permite verificar que el componente *Add-homework* ha sido cargado correctamente y es mostrado en una ventana emergente.
Verificar conexión entre el componente *Chatbot* con *Add-files*.	Ayuda a verificar que existe una conexión y transmisión/recepción de información entre los componentes *Chatbot* y *Add-files*.
Verificar conexión entre el componente *Chatbot* con *Add-homework*.	Ayuda a verificar que existe una conexión y transmisión/recepción de información entre los componentes *Chatbot* y *Add-homework*.
Verificar conexión entre *Dialogflow* con el sistema.	Verifica si se realiza la conexión entre el *chatbot* alojado en *Dialogflow* con el sistema.

Nombre de la prueba	**Descripción**
Consultar materias desde *Add-files*.	Verifica si se consultan y muestran, desde *Add-files*, las materias que actualmente da el profesor conectado en el sistema en los grados y grupos seleccionados.
Consultar materias desde *Add-homework*.	Verifica si se consultan y muestran, desde *Add-homeworks*, las materias a las que actualmente asiste el alumno conectado en el sistema.
Consultar tareas *Add-homework*.	Verifica si se consultan y muestran, desde *Add-homeworks*, una vez seleccionada la materia, las tareas que tiene el alumno conectado en el sistema.
Validar formulario de *Add-files*.	Verifica que el formulario de *Add-files* esté realizando las validaciones correctas.
Validar formulario de *Add-homework*.	Verifica que el formulario de *Add-homework* esté realizando las validaciones correctas.
Registrar material.	Valida si se registra correctamente un material en el sistema.
Registrar tarea.	Valida si se registra correctamente una tarea en el sistema.
Confirmar registrar la tarea.	Prueba si se realiza, con el componente *Confirm-dialog*, la verificación de si se desea registrar una nueva tarea.
Confirmar subir el material.	Prueba si se realiza, con el componente *Confirm-dialog*, la verificación de si se desea subir cierto material.
Redireccionar desde *Chatbot*.	Verifica si se redirige correctamente a las rutas definidas en el menú del *header* contenido en *Chatbot*.
Módulo: Gestión de archivos y mensajes	
Iniciar componente *General-files*.	Permite verificar que el componente General-files ha sido cargado correctamente.
Consultar archivos recibidos.	Verifica si se consultan y muestran en una tabla los avisos recibidos por la cuenta del usuario actualmente logeado.
Consultar alumnos destinatarios.	Verifica si se consultan y muestran en tabla los alumnos, dados su grado y grupo, como posibles destinatarios a recibir mensajes.

Nombre de la prueba	**Descripción**
Consultar profesores destinatarios.	Verifica que se devuelva el nombre de los profesores que pueden recibir mensajes.
Validar formulario de *General-files*.	Verifica que el formulario de *General-files* esté realizando las validaciones correctas.
Enviar archivo y mensaje.	Valida si se envía correctamente un archivo y mensaje a los destinatarios seleccionados.
Confirmar envio de mensaje.	Prueba si se realiza con el componente *Confirm-dialog*, la verificación de si se desea enviar un mensaje.
Redireccionar desde *Reminders*.	Verifica si se redirige correctamente a las rutas definidas en el menú del *header* contenido en *Reminders*.
Módulo: Administración de calificaciones	
Iniciar componente *Scores*.	Permite verificar que el componente *Scores* ha sido cargado correctamente.
Iniciar componente *Score-user*.	Permite verificar que el componente *Score-user* ha sido cargado correctamente y es mostrado en una ventana emergente.
Verificar conexión entre el componente *Scores* con *Score-user*.	Ayuda a verificar que existe una conexión y transmisión/recepción de información entre los componentes *Scores* y *Score-user*.
Llenar automáticamente parámetros de filtro.	Verifica que en los filtros solo se muestren los grados, grupos y materias en donde imparte clases el profesor que actualmente está logeado en el sistema.
Consultar calificaciones.	Verifica si se consultan y muestran en una tabla, las calificaciones de los alumnos que toman clase en la materia, grado y grupo seleccionados.
Consultar bimestre reportado.	Verifica que, seleccionados el grado, grupo y materia, se muestre el bimestre hasta el cual se ha calificado la clase.
Modifica bimestre reportado.	Verifica que se seleccionados el grado, grupo y materia, se pueda modificar el bimestre hasta el cual se ha calificado la materia.

Nombre de la prueba	**Descripción**
Registrar calificaciones.	Verifica si se registran o modifican las calificaciones de un alumno seleccionado.
Validar formulario de *Score-user*.	Verifica que el formulario de *Score-user* esté realizando las validaciones correctas.
Confirmar el registro de calificaciones.	Prueba si se realiza con el componente *Score-user* la verificación de si se desean actualizar las calificaciones de un alumno.
Confirmar desactivación de avisos.	Prueba si se realiza con el componente *Confirm-dialog* la verificación de si se desea desactivar un aviso.
Redireccionar desde *Scores*.	Verifica si se redirige correctamente a las rutas definidas en el menú del *header* contenido en *Scores*.
Módulo: Progreso de grupos	
Iniciar componente *Progress-group*.	Permite verificar que el componente *Progress-group* ha sido cargado correctamente.
Iniciar componente *General-progress*.	Permite verificar que el componente *General-progress* ha sido cargado correctamente.
Iniciar componente *Individual-progress*.	Permite verificar que el componente *Individual-progress* ha sido cargado correctamente.
Iniciar componente *Pass-fail*.	Permite verificar que el componente *Pass-fail* ha sido cargado correctamente.
Iniciar componente *Rating-percentages*.	Permite verificar que el componente *Rating-percentages* ha sido cargado correctamente.
Verificar conexión entre el componente *Progress-group* con *General-progress*, *Individual-progress*, *Pass-fail* y *Rating-percentages*.	Ayuda a verificar que existe una conexión y transmisión/recepción de información entre los componentes *Progress-group* con *General-progress*, *Individual-progress*, *Pass-fail* y *Rating-percentages*.

Nombre de la prueba	**Descripción**
Consultar progreso general de calificaciones.	Verifica si se consultan y muestran en una gráfica los promedios generales de calificaciones en cada bimestre de los alumnos que toman clase en la materia, grado y grupo seleccionados.
Consultar alumnos aprobados y reprobados.	Verifica si se consultan y muestran en gráficas los alumnos y la cantidad de ellos que aprobaron y reprobaron cada bimestre en la materia, grado, grupo y bimestre seleccionados.
Consultar porcentajes calificaciones de alumnos.	Verifica si se consultan y muestran en una gráfica los porcentajes por cada rango de calificaciones obtenidos por los alumnos según la materia, grado, grupo y bimestre seleccionados.
Consultar rendimiento individual de un alumno.	Verifica si se consulta el rendimiento individual por medio de una gráfica de determinado alumno.
Brindar observaciones del rendimiento individual de un alumno.	Permite realizar y mostrar observaciones, en base al rendimiento de un alumno seleccionado.
Redireccionar desde *Progress-group*.	Verifica si se redirige correctamente a las rutas definidas en el menú del *header* contenido en *Progress-group*.
Módulo: Dudas de estudiantes	
Iniciar componente *Doubts*.	Permite verificar que el componente *Doubts* ha sido cargado correctamente.
Iniciar componente *Material-doubts*.	Permite verificar que el componente *Material-doubts* ha sido cargado correctamente.
Iniciar componente *Homework-doubts*.	Permite verificar que el componente *Homework-doubts* ha sido cargado correctamente.
Iniciar componente *Exam-doubts*.	Permite verificar que el componente *Exam-doubts* ha sido cargado correctamente.
Iniciar componente *General-doubts*.	Permite verificar que el componente *General-doubts* ha sido cargado correctamente.

Nombre de la prueba	**Descripción**
Iniciar componente *Details-doubts*.	Permite verificar que el componente *Details-doubt* ha sido cargado correctamente y es mostrado en una ventana emergente.
Verificar conexión entre el componente *Doubts* con *Material-doubts*, *Homework-doubts*, *Exam-doubts* y *General-doubts*.	Ayuda a verificar que existe una conexión y transmisión/recepción de información entre los componentes *Doubts* con *Material-doubts*, *Homework-doubts*, *Exam-doubts* y *General-doubts*.
Consultar materiales disponibles.	Verifica que se obtienen y muestran en filtro los materiales que se tienen disponibles en el sistema una vez seleccionados el grado, grupo y materia.
Consultar percepciones del material seleccionado.	Verifica si se consultan y muestran las percepciones de los alumnos, respecto al material elegido para resolver dudas de tareas, exámenes y generales de una clase, grado y grupo seleccionados.
Consultar dudas de tarea.	Verifica si se consultan y muestran en una tabla las dudas de determinada tarea en la materia, para un grado y grupo seleccionados.
Consultar dudas de examen.	Verifica si se consultan y muestran en tabla las dudas de determinado examen en la materia, para un grado y grupo seleccionados.
Consultar percepciones de alumnos en dudas de exámenes.	Verifica si se consultan y muestran en una gráfica las percepciones de los alumnos, respecto a las recomendaciones que se les han dado a sus dudas de exámenes, para una clase en la materia, grado y grupo seleccionados.
Consultar dudas generales.	Verifica si se consultan y muestran en tabla las dudas generales en la materia, para un grado y grupo seleccionados.

Nombre de la prueba	**Descripción**
Consultar percepciones de alumnos en dudas generales.	Verifica si se consultan y muestran en una gráfica las percepciones de los alumnos, respecto a las recomendaciones que se les han dado a sus dudas generales, para una clase en la materia, grado y grupo seleccionados.
Consultar detalles de dudas de estudiantes.	Verifica si se muestran detalles como los materiales recomendados de cierta duda, en el componente Details-doubts.
Agregar retroalimentación a dudas.	Verifica si se puede realizar el registro de una retroalimentación a cierta duda en el componente Details-doubts.
Redireccionar desde *Doubts*.	Verifica si se redirige correctamente a las rutas definidas en el menú del *header* contenido en *Doubts*.
Módulo: Administración de tareas	
Iniciar componente *Homeworks*.	Permite verificar que el componente *Homeworks* ha sido cargado correctamente.
Iniciar componente *Homeworks-student*.	Permite verificar que el componente *Homeworks-student* ha sido cargado correctamente.
Iniciar componente *View-details-homework*.	Permite verificar que el componente *View-details-homework* ha sido cargado correctamente y es mostrado en una ventana emergente.
Verificar conexión entre el componente *Homeworks* con *View-details-homework* y *Feedback-homework*.	Ayuda a verificar que existe una conexión y transmisión/recepción de información entre los componentes *Homeworks* con *View-details-homework* y *Feedback-homework*.
Verificar conexión entre el componente *Homeworks-student* con *View-details-homework* y *Feedback-homework*.	Ayuda a verificar que existe una conexión y transmisión/recepción de información entre los componentes *Homeworks-student* con *View-details-homework* y *Feedback-homework*.

Nombre de la prueba	**Descripción**
Consultar tareas desde *Homeworks*.	Verifica que para un profesor se obtienen y muestran en una tabla, las tareas y el estatus de las mismas en el sistema para cada alumno, una vez seleccionados el grado, grupo y materia.
Validar formulario de retroalimentar tarea.	Permite verificar que se realiza la validación de los campos del formulario, para retroalimentar una tarea.
Registrar retroalimentación de una tarea.	Permite que un profesor pueda retroalimentar determinada tarea de algún alumno perteneciente a un grado, grupo y materia.
Consultar tareas desde *Homeworks-student*.	Permite ver si se consultan las tareas que tiene un alumno para determinada clase. Así como los detalles de dicha tarea.
Consultar retroalimentación de tareas desde *Homeworks-student*.	Permite que un estudiante pueda consultar la retroalimentación de cierta tarea seleccionada.
Redireccionar desde *Homeworks*.	Verifica si se redirige correctamente a las rutas definidas en el menú del *header* contenido en *Homeworks*.
Redireccionar desde *Homeworks-student*.	Verifica si se redirige correctamente a las rutas definidas en el menú del *header* contenido en *Homeworks-student*.
Módulo: Progreso en clases	
Iniciar componente *Progress-student*.	Permite verificar que el componente *Progress-student* ha sido cargado correctamente.
Consultar el progreso de determinado estudiante.	Verifica que para el estudiante conectado en el sistema, se obtenga y grafique su rendimiento general para todas las materias que cursa.
Consultar el progreso por materia de estudiante.	Verifica que para el estudiante logeado en el sistema, una vez seleccionada una materia, se muestre su rendimiento.
Redireccionar desde *Progress-student*.	Verifica si se redirige correctamente a las rutas definidas en el menú del *header* contenido en *Progress-student*.

Cuadro C.1: Pruebas unitarias realizadas a los módulos del sistema.

Referencias

Basogain, X., Olabe, M. n., & Olabe, J. C. (2017). Transition to a modern education system through e-learning. In *Proceedings of the 2017 International Conference on Education and E-Learning*, (pp. 41–46).

Benotti, L., Martínez, M. C., & Schapachnik, F. (2014). Engaging high school students using chatbots. In *Proceedings of the 2014 conference on Innovation & technology in computer science education*, (pp. 63–68).

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of machine Learning research*, *3*(Jan), 993–1022.

Bollweg, L., Shahriar, A., Stemmermann, R., & Weber, P. (2019). Who am i, and if so how many? a multi-entity chatbot business-interview simulator for individualized practical assignments in moocs. In *EdMedia+ Innovate Learning*, (pp. 928–936). Association for the Advancement of Computing in Education (AACE).

Brandtzaeg, P. B. & Folstad, A. (2017). Why people use chatbots. In *International Conference on Internet Science*, (pp. 377–392). Springer.

Brown, C. & Salmi, J. (2020). Putting fairness at the heart of higher education. *University World News. The Global Window on Higher Education*, *18*.

Carter, L. (2006). Why students with an apparent aptitude for computer science don't choose to major in computer science. *ACM SIGCSE Bulletin*, *38*(1), 27–31.

Chaix, B., Bibault, J.-E., Pienkowski, A., Delamon, G., Guillemasse, A., Nectoux, P., & Brouard, B. (2019). When chatbots meet patients: one-year prospective study of conversations between patients with breast cancer and a chatbot. *Jmir Cancer*, *5*(1), e12856.

Clarizia, F., Colace, F., Lombardi, M., Pascale, F., & Santaniello, D. (2018). Chatbot: An education support system for student. In *International Symposium on Cyberspace Safety and Security*, (pp. 291–302). Springer.

Cunningham-Nelson, S., Boles, W. W., Trouton, L., & Margerison, E. (2019). A review of chatbots in education: Practical steps forward.

Davison, R., Martinsons, M. G., & Kock, N. (2004). Principles of canonical action research. *Information systems journal*, *14*(1), 65–86.

Deacon, J. (2009). Model-view-controller (mvc) architecture. *http://www. jdl. co. uk/briefings/MVC. pdf*.

Dennen, V. P., Aubteen Darabi, A., & Smith, L. J. (2007). Instructor–learner interaction in online courses: The relative perceived importance of particular instructor actions on performance and satisfaction. *Distance education*, *28*(1), 65–79.

Federal, A. E. (2018). Guía operativa para la organización y funcionamiento de los servicios de educación.

Fonte, F. A. M., Nistal, M. L., Rial, J. C. B., & Rodríguez, M. C. (2016). Nlast: A natural language assistant for students. In *2016 IEEE Global Engineering Education Conference (EDUCON)*, (pp. 709–713). IEEE.

Garcia-Peñalvo, F. J. (2014). *Online Tutor 2.0: Methodologies and case studies for successful learning: Methodologies and case studies for successful learning*. IGI Global.

Graells, P. M. (2000). Las tic y sus aportaciones a la sociedad. *Departamento de pedagogea aplicada, facultad*.

Graells, P. M. (2013). Impacto de las tic en la educacion: funciones y limitaciones. *3C TIC. Cuadernos de Desarrollo Aplicados a las TIC*, *2*(1).

Grice, H. P. (1975). Logic and conversation. In *Speech acts* (pp. 41–58). Brill.

Koda, T. & Maes, P. (1996). Agents with faces: The effect of personification. In *Proceedings 5th IEEE International Workshop on Robot and Human Communication. RO-MAN'96 TSUKUBA*, (pp. 189–194). IEEE.

Kuligowska, K. (2015). Commercial chatbot: performance evaluation, usability metrics and quality standards of embodied conversational agents. *Professionals Center for Business Research*, *2*.

L. Sánchez, S. Mendoza, J. U. (2020). Towards a set of heuristics for evaluating chatbots.

Lam, C. (2010). *Hadoop in action*. Simon and Schuster.

Lawrence, J. (1993). *Introduction to neural networks*. California Scientific Software.

Lee, J. J. (2009). Size matters: an exploratory comparison of small-and large-class university lecture introductions. *English for Specific Purposes*, *28*(1), 42–57.

Li, J., Zhou, M. X., Yang, H., & Mark, G. (2017). Confiding in and listening to virtual agents: The effect of personality. In *Proceedings of the 22nd International Conference on Intelligent User Interfaces*, (pp. 275–286).

Lozano, J. (2005). En la universidad se deberia enseñar e-learning. *Recuperado de http : //www.noticias.com/articulo/06 − 02 − 2005/jose − lozano/universidad − se − deberiaensenar − e − learning − 4dil.html*.

Mendoza, S., Hernandez-Leon, M., Sanchez-Adame, L. M., Rodriguez, J., Decouchant, D., & Meneses-Viveros, A. (2020). Supporting student-teacher interaction through a chatbot. In *International Conference on Human-Computer Interaction*, (pp. 93–107). Springer.

Miklos, T. & Arroyo, M. (2008). El futuro de la educacion a distancia y del e-learning en america latina: una vision prospectiva. *Mexico: iLCE*.

OECD (2016). Pisa 2015 results in focus.

Olabe, J., Basogain, X., Olabe, M., Maz, I., & Castao, C. (2014). Solving math and science problems in the real world with a computational mind. *Journal of New Approaches in Educational Research (NAER Journal)*, *3*(2), 75–82.

Ritzmann, S., Hagemann, V., & Kluge, A. (2014). The training evaluation inventory (tei)-evaluation of training design and measurement of training outcomes for predicting training success. *Vocations and Learning*, *7*(1), 41–73.

Roca, S., Sancho, J., García, J., & Alesanco, A. (2020). Microservice chatbot architecture for chronic patient support. *Journal of Biomedical Informatics*, *102*, 103305.

Rozga, S. (2018). Practical bot development. *1*(1).

Santuario et al. (2020). Educación superior y covid-19: una perspectiva comparada.

Shaw, A. (2012). Using chatbots to teach socially intelligent computing principles in introductory computer science courses. In *2012 Ninth International Conference on Information Technology-New Generations*, (pp. 850–851). IEEE.

Sproull, L., Subramani, M., Kiesler, S., Walker, J. H., & Waters, K. (1996). When the interface is a face. *Human-computer interaction*, *11*(2), 97–124.

Techlabs, M. (2018). Complete guide on bot frameworks.

UNESCO, I. (2020). El coronavirus-19 y la educacion superior: impacto y recomendaciones.

Verleger, M. & Pembridge, J. (2018). A pilot study integrating an ai-driven chatbot in an introductory programming course. In *2018 IEEE Frontiers in Education Conference (FIE)*, (pp. 1–4). IEEE.

Villegas-Ch, W., Arias-Navarrete, A., & Palacios-Pacheco, X. (2020). Proposal of an architecture for the integration of a chatbot with artificial intelligence in a smart campus for the improvement of learning. *Sustainability*, *12*(4), 1500.

Wilson, C., Sudol, L. A., Stephenson, C., & Stehlik, M. (2010). Running on empty: The failure to teach k-12 computer science in the digital age. *Association for Computing Machinery*, *26*.

Xiao, Z., Zhou, M. X., Liao, Q. V., Mark, G., Chi, C., Chen, W., & Yang, H. (2020). Tell me about yourself: Using an ai-powered chatbot to conduct conversational surveys with open-ended questions. *ACM Transactions on Computer-Human Interaction (TOCHI)*, *27*(3), 1–37.