DEPARTAMENTO DE TECNOLOGÍAS DE LA INFORMACIÓN

LIC. EN TECNOLOGÍAS Y SISTEMAS DE INFORMACIÓN

Da Capo: una propuesta de aplicación web con agentes BDI para la enseñanza de música

Proyecto Terminal

Nieto Rocha Alberto
2163071736@cua.uam.mx

Asesor: Dr. Wulfrano Arturo Luna Ramírez
Asesora: Dra. Anabel Velásquez Durán

10 de febrero de 2022

Resumen

Las condiciones actuales de vida nos imponen el uso de herramientas digitales. Uno de sus más grandes usos es la educación, esto ya que dichas herramientas permiten llegar a más personas, tener una amplia disponibilidad de recursos y ayudan a satisfacer la demanda de la educación. Aprender música puede traer distintos beneficios a la salud como mejorar el estado del ánimo. En este contexto surge la idea de crear una herramienta inteligente capaz de enseñar conceptos básicos de música. Se hace una propuesta de esta herramienta con la integración de *agentes BDI*, y *Flask* para crear un sistema web.

Keywords: Agentes BDI, agentes tutores inteligentes, Python, Flask, tutores de música, Prometheus, OpenUP, teoría musical.

Índice de figuras

Índice de tablas

Índice de códigos

Índice

1. Introducción

1.1. Motivación/Justificación

La necesidad de crear un agente tutor de música nace a partir de las condiciones de vida actual, en la que las capacidades físicas de interacción humana se han visto limitadas por la amenaza mundial del virus SARS-CoV-2 y COVID-19 como enfermedad asociada.. Las herramientas tecnológicas han logrado solventar algunos de los problemas generados por el distanciamiento social, entre ellos se encuentra la impartición de clases. Gran parte de las escuelas por todo el mundo han optado por las actividades a distancia, gracias al uso de internet y plataformas virtuales que les permiten impartir clases, asignar actividades, revisar tareas, entre otras cosas.

Enseñar música puede resultar beneficioso para la salud, en un estudio realizado por Seinfeld et al. [10], se impartieron clases de teoría básica de música y lecciones de piano durante cuatro meses a un grupo de adultos mayores, los cuales no tenían conocimientos acerca del tema. En la investigación determinaron que aprender música y lo que esto conlleva, puede ayudar a combatir la depresión y a promover un mejor estado de ánimo. Lo anterior son los beneficios de aprender música en la edad adulta, pero como menciona el artículo, estudiar música a cualquier edad es una actividad muy lucrativa en términos de salud.

La educación a distancia también se ha dado en la música. Por ejemplo, algunas escuelas de música han impartido cursos formales e informales en verano, dichos cursos se orientan a la enseñanza de temas desde nivel básico hasta niveles avanzados. Para afrontar esta situación, se optaron por actividades como grabar videos y audios, en los cuales los alumnos realizaban los ejercicios. De este modo al contar con estas grabaciones, los profesores pueden dar una retroalimentación a los estudiantes que les permita enfocar sus esfuerzos en los puntos que así lo requieran. Como menciona Lourdes et al. [11] uno de los problemas de la educación a distancia, surge por la alta demanda y dificultades que representa atender a tantos alumnos; los profesores se ven superados en la carga de trabajo y esto insta a no poder brindar una retroalimentación adecuada a cada alumno, esto sin tener en cuenta que hay educadores poco adiestrados en el uso de las tecnologías.

El proyecto que se presenta en este documento, tiene la finalidad de brindar soporte a los estudiantes de música de cualquier edad que deseen reforzar los conocimiento y habilidades básicas que tienen sobre el tema, presentando

conceptos y ejercicios prácticos. Esto con ayuda de un tutor inteligente musical llamado Da Capo; esta herramienta tecnológica, será capaz de adaptarse a las necesidades de aprendizaje de cada alumno y ofrecerá retroalimentación constante.

1.2. Definición del problema

El interés por la educación musical ha ido en aumento, y es de esperarse que esto continúe así [12]. La alta demanda de alumnos que tienen las escuelas [13] de música y en muchas ocasiones el poco tiempo que se tiene para acudir a éstas son un detonante para que las personas abandonen el interés por la música. En ocasiones, el tener un educador que supervise y retroalimente los ejercicios teóricos y prácticos puede ser crucial para que el alumno decida o no continuar con sus estudios. Actualmente, existen unas cuantas aplicaciones para la enseñanza de música, éstas usualmente se centran en impartir lecciones generales para todos sus usuarios, en lugar de que el contenido se adapte a las necesidades únicas del estudiante, además suelen ser software de pago. Es por esto que es necesaria la creación de métodos inteligentes que permitan supervisar a los estudiantes de música en su formación. Una solución para desarrollar estos métodos, es hacer uso de tecnologías que en su núcleo dispongan de la inteligencia artificial. Estas son conocidas como *agentes*, existen distintos enfoques para crearlos, pero en lo que concierne a este proyecto se empleará el modelo BDI que por sus siglas en inglés hace referencia a las creencias, deseos e intenciones. Estos son atractivos, ya que saben qué objetivos perseguir y son capaces de planear como realizarlos [14].

2. Métodos de enseñanza musical mediados por tecnologías

La educación formal de la música ha estado orientada a enseñanzas clásicas, en las cuales el profesor y los alumnos se reúnen para llevar a cabo el acto de la enseñanza/aprendizaje. La tecnología ha permitido innovar en estos típicos actos, permitiendo a los alumnos hacer uso de herramientas de software que son capaces de fungir como sus tutores [15]. Estas herramientas tiene la particularidad de poder brindar una atención personalizada e instantánea a cada alumno, lo que permite que los estudiantes sean capaces de avanzar a ritmo propio y ayuda a prevenir el dejar vacíos en la comprensión

de algún tema. En este apartado se hablará de los métodos tradicionales para enseñar música y como éstos pueden ser adaptados para enseñar conceptos básicos de teoría musical junto con las tecnologías actuales [16, 17]. También se presentarán las tecnologías que adoptan el rol de docente y los métodos que se utilizan para desarrollarlas [15, 18, 9, 19].

2.1. Educación musical contemporánea

2.1.1. Metodologías en la música

La teoría de la música en palabras de Claude et al. [1] puede ser enseñada desde tres puntos de vista diferentes: (1) por el arte, (2) la práctica o (3) por la historia. Éste menciona que se pueden mezclar los tres puntos de vista para descubrir, practicar y explorar la teoría musical en toda su rica extensión y musicalidad.

La teoría de la música	
1. La práctica musical	Componer, interpretar, improvisar, escuchar, analizar La teoría siempre está presente de manera implícita. Es preciso integrarla para que resulte eficaz.
2. La práctica de la teoría	Tocar, hacer ejercicios, solfear La teoría debe volverse inconsiente, tal como la práctica de la lengua materna; por tanto debe ejercitarse mucho para formar el oído y una técnica.
3. Historia de la música	Conocer a los creadores, los teóricos, las corrientes estéticas La teoría ha evolucionado de acuerdo con las necesidades artísticas y estéticas, por ello resulta indispensable ubicarlas para evitar importantes malentendidos.

Figura 1: Puntos de vista que conforman la teoría musical. Adaptado de [1].

Teniendo en mente lo anterior, podemos hablar de distintos métodos, estos consisten en un conjunto de principios pedagógicos, prácticas y metas las cuales están bien definidas y juntas persiguen un objetivo. Los métodos suelen ser sistemáticos y el contenido presentado se muestra de forma secuencial, es decir, primero se enseñan las bases de la música y después los temas más complejos. Usualmente los métodos se enfocan en las mentes más jóvenes (niños) y esto es porque sus cerebros tienen mayor plasticidad. Pero como se mencionó anteriormente, hay estudios que indican los beneficios que tiene aprender música en cualquier etapa de la vida [10].

Uno de los métodos para enseñar música es llamado talento Suzuki, este nace en Japón y toma como base la forma en que los niños aprenden su lengua materna. Cuando un niño aprende su lengua materna se somete a un procesos de observación, imitación y repetición, esto mismo es lo que se quie-

re con este método. De manera general el método consiste en lo siguiente: el niño desde una edad temprana tiene que estar en contacto con la música para que aprenda el vocabulario y la estructura de ella. En un determinado momento los padres pasan a ser los tutores musicales del niño y empiezan a impartirle sus primeras lecciones de instrumento, en las cuales los ejercicios que se le presentan al niño son muy sencillos y éste tiene que repetirlos una y otra vez. Es el constante contacto con la música y la repetición lo que hace que el niño siente las bases de la música [16]. Por otro lado, está el método Kodaly, este nace por la necesidad de mejorar el canto en los niños, en este se realizan actividades como leer a primera vista, aprender a dominar los ritmos, identificación de los tonos básicos y con básicos se refiere a que el niño debe aprender a identificar notas por sí solas, ya que domine eso se va aumentando el grado de complejidad, agregando ritmos y otras notas. Este método tiene como enfoque principal enseñar a cantar, leer, escribir música, entrenar el oído y fomentar la improvisación [17].

Con la conjunción de algunas de las actividades que se realizan en estos métodos (repetición, leer y escribir música, entrenamiento de oido) es posible enseñar conceptos musicales fundamentales a personas con poco conocimiento del tema, buscando reforzar conceptos como: escala diatónica[1], sonido[2], acordes[3], y ritmo[4].

Como una primera versión de Da Capo, los contenidos serán diseñados para abordar temas teóricos y habilidades prácticas. Con ejercicios en donde el alumno será capaz de identificar conceptos, mejorar su oído musical, y conocer historia de la música. Esto haciendo uso de diversos materiales didácticos como: textos, sonidos y videos. Cabe aclarar que tanto el método *Suzuki* y *Kodaly* se enfocan más que nada en enseñar a los niños a tocar un instrumento, en el caso de Da Capo este no se enfocará ni en los niños ni en tocar un instrumento. El sistema se centrará en un público que ya cuente con algún conocimiento del tema y no se enseñará a tocar un instrumento. Esto es dado a que se requeriría ser muy escrupuloso al diseñar actividades para niños y al enseñar a tocar un instrumento, y ya que el tiempo del proyecto es muy breve, realizar esto sería muy complejo.

[1] *Escala diatónica*: sucesión de siete sonidos, más la repetición del primero, dispuestos por grados conjuntos según las leyes de la tonalidad [1].

[2] *Sonido*: es el resultado de las vibraciones de un cuerpo sonoro, se compone de cinco elementos: altura, duración, intensidad, timbre y espacio [1, 20].

[3] *Acordes*: sobreposición de al menos tres sonidos simultáneos que forman un todo [1].

[4] *Ritmo*: es el orden y la proporción en que se agrupan los sonidos en el tiempo [1].

2.2. Agentes

Los agentes son sistemas informáticos que se encuentra situados en un ambiente, son capaces de percibir los cambios que se presenten y de acuerdo a éstos interactuar de manera autónoma y persistente. Su existencia se resume en: cumplir los objetivos para los cuales han sido diseñados. Las principales características de los agentes son: *autonomía*, *persistencia*, *reactividad*, *proactividad* y *habilidad social* [2].

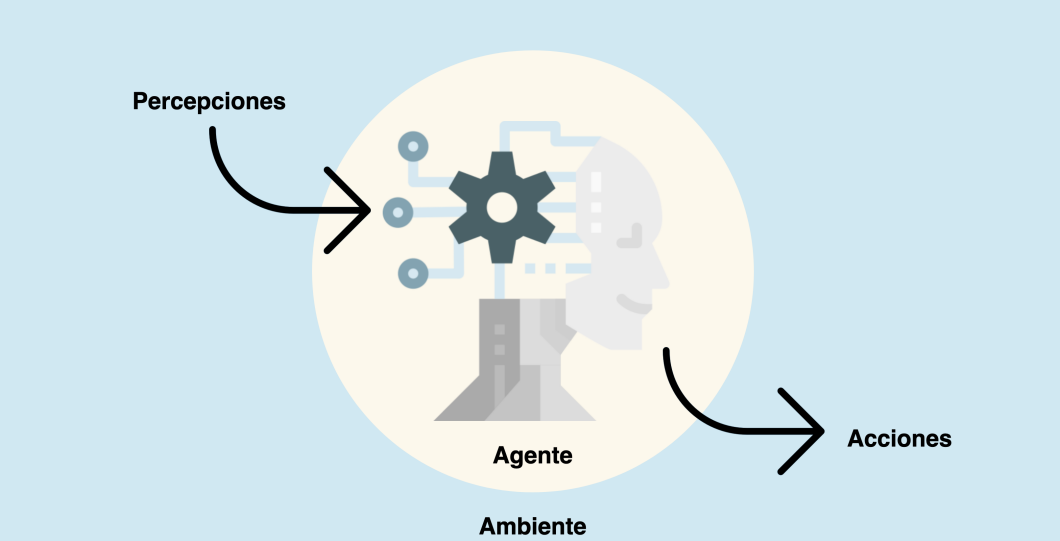

Figura 2: Vista de un agente sencillo y sus interacciones con el ambiente. Adaptado de [2].

La *reactividad* se puede considerar como la interacción perpetua con su ambiente, esto lo obliga a responder a los cambios que se presenten. Para poder responder adecuadamente a los cambios, necesita estar dotado de iniciativa, esta característica es conocida como *proactividad*. Como su conducta no puede ser un producto de la casualidad, los agentes deben ser *autónomos*, es decir, deben basar su conducta en la medida de sus experiencias. Esto implica que el agente no se base sólo en el conocimiento con el que fue creado, mientras más base su comportamiento en el aprendizaje y en el conocimiento incorporado, más autónomo será. La *habilidad social* en los agentes, se describe como su capacidad de interactuar con otros agentes e incluso humanos, se deben valer de ciertos lenguajes y protocolos para establecer una comunicación certera, ya que con esto pueden lograr ciertas metas que sólo es posible conseguir en conjunto con sus semejantes, a los conjuntos de éstos se les conoce como sistemas multiagentes. La *persistencia* se expresa como el

funcionamiento constante del sistema.

2.2.1. Agentes tutores

Los agentes tutores inteligentes guían al alumno en el proceso de aprendizaje, brindando retroalimentación constante, por medio de evaluaciones el agente tutor inteligente elige los temas en los que el estudiante presenta algún déficit de aprendizaje. Algunas de las investigaciones que se han realizado acerca del tema, señalan que los agentes tutores inteligentes tienen cuatro componentes: la base de conocimiento, el modelo del estudiante, el modelo del tutor y la interfaz de usuario [18, 9, 15].

2.2.1.1. Componentes

- Base de conocimiento
  La base de conocimiento es la encargada de almacenar todos los materiales que el agente tutor inteligente está enseñando, en ésta pueden estar contenidos: sonidos, imágenes, textos, entre otras cosas. También deben estar las soluciones y explicaciones de los temas.

- Modelo del estudiante
  Esta tiene que ver con toda la información que concierne al estudiante, como las credenciales de acceso que identifican al estudiante, las actividades que realiza el estudiante, los errores y aciertos acerca de los temas vistos.

- Modelo del tutor
  El modelo del tutor es el encargado de elegir los temas de la base del conocimiento que el alumno verá, para poder tomar esta decisión el tutor analiza la información recaudada por el modelo del estudiante.

- Interfaz de usuario
  Es la conexión entre el sistema tutor y el estudiante. Muestra en pantalla los ejercicios y materiales de forma amena. Es necesario que este componente este refinado para brindar una retroalimentación adecuada al alumno, ya que esto le ayudara a entender los errores que tenga. También debe considerarse la facilidad de uso, ya que el usuario debe concentrarse en los temas presentados, no en cómo utilizar la plataforma.

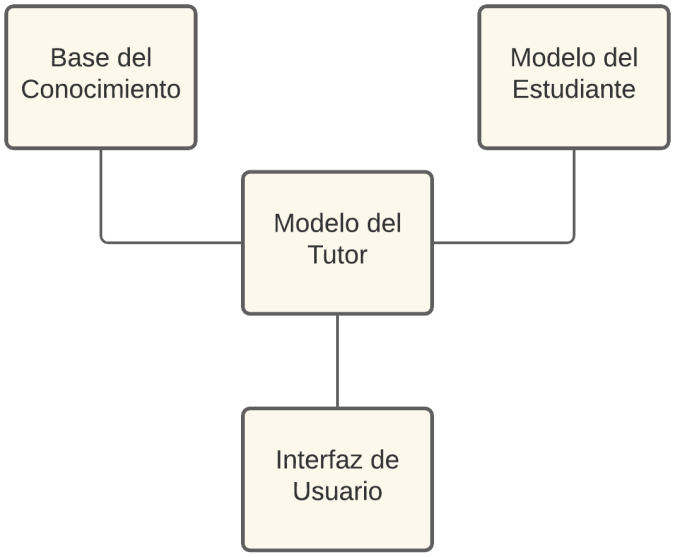

Figura 3: Componentes de un agente tutor. Adaptado de [3].

2.2.2. Agentes tutores de música

Para crear un agente tutor inteligente de música, se deberán tomar como base los componente de los agentes tutores, a estos se les pueden agregar otros componentes que el desarrollador considere necesarios y claro, la base de conocimiento se deberá encontrar cargada con contenidos relacionados al ámbito musical. Por ejemplo, en el artículo relacionado con *WMITS*, [9] se comenta que, además de los componentes de los agentes tutores, utilizan paquetes de terceros que se conectan directamente con el modelo del tutor. Uno de ellos es *Jasper SICStus* que se usa para interactuar con el motor de *Prolog*[5]. El otro, *mm.mysql-2.0.8-bin.jar*, es una biblioteca que permite que la base de datos *MySQL*[6] se conecte con el modelo del tutor. También usan *XML*[7] para guardar los archivos generados por su editor de notación musical.

[5]Prolog: es un lenguaje de programación lógico e interpretado usado habitualmente en el campo de la Inteligencia artificial.

[6]MySQL: sistema de gestión de bases de datos relacionales.

[7]XML: es un metalenguaje que permite definir lenguajes de marcas, y es utilizado para almacenar datos en forma legible.

2.2.3. Agentes BDI

García et al. [21], menciona que el modelo BDI es una de las arquitecturas más prometedoras. Esto es debido a su capacidad de razonamiento práctico, que en otras palabras es el proceso de decidir qué acciones realizar para lograr sus objetivos. La arquitectura BDI está basada en creencias, deseos e intenciones [22], e integra las posturas de *Dennett* y *Bratman*. Ellas son: *aproximación intencional* y *Planes e interacciones y el razonamiento práctico*, respectivamente. Esta arquitectura ha sido ampliamente probada en una diversidad de sistemas que van desde manufactura hasta navegación espacial.

Las *creencias* son la información que conoce el agente sobre su entorno, estas se actualizan por medio de las percepciones y las intenciones. Los *deseos* son las metas del agente. Las *intenciones* son pilas (como las estructuras de datos) de planes en ejecución, estos últimos se almacenan en una biblioteca de planes. Los planes son habilidades y/o conocimientos procedimentales que el agente BDI tiene a su disposición para alcanzar ciertos estados. Dentro de los planes se encuentran las acciones, las cuales pueden actualizar las creencias y/o modificar el ambiente [4, 23]. En la Figura 4, se muestra lo explicado anteriormente.

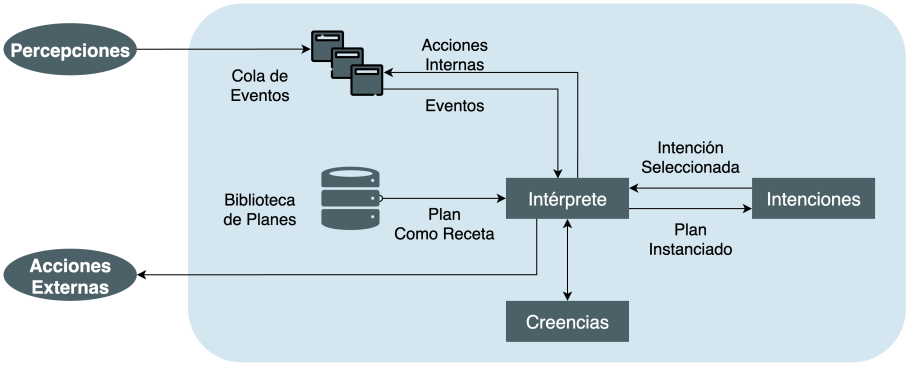

Figura 4: Arquitectura de un agente BDI. Adaptado de [4].

2.2.4. Desarrollo de software

Para obtener un sistema que cumpla con los requisitos demandados por un usuario, es necesario manejar adecuadamente el desarrollo del proyecto. Para ello se emplean metodologías de ingeniería de desarrollo de software, éstas generalmente están constituidas por un conjunto de herramientas y métodos que al ser puestos en práctica, asisten en el proceso de elaboración

del proyecto. Para llevar a cabo el desarrollo de Da Capo, será indispensable emplear dos metodologías: una para la planeación general del proyecto (*OpenUP*) y la otra para el desarrollo del agente (*Prometheus*).

2.2.4.1. OpenUP

OpenUP [5] es una metodología de desarrollo de software ágil[8] y unificada que ayuda a los equipos de desarrollo a realizar un producto de alta calidad, de una forma eficiente.
OpenUP aborda solo el contenido fundamental de un proyecto, dejando de lado temas específicos que puedan suscitarse en un proyecto. Esta particularidad permite que *OpenUP* sea extensible y que sirva como base para usarse con otros procesos y métodos que permitan abarcar la totalidad de los temas del proyecto. La metodología tiene la característica de ser un proceso unificado ágil que aplica enfoques iterativos e incrementales dentro de un ciclo de vida estructurado. Otra de sus característica es que posibilita el ahorro de tiempo del equipo, ya que solo es necesario que se elaboren los documentos y diagramas indispensables para el proyecto. *OpenUP* es apropiada para proyectos pequeños y de escasos recursos y es aplicable a un conjunto amplio de plataformas y aplicaciones de desarrollo.

Con la metodología *OpenUP* los proyectos se llevan a cabo con micro-incrementos, los cuales son el resultado de trabajar durante un periodo de tiempo de unas cuantas horas o bien de algunos cuantos días. Con los micro-incrementos el software poco a poco toma forma, esto permite que el equipo de desarrollo pueda ir palpando su trabajo, lo cual genera confianza entre los miembros del equipo y hace que el trabajo en equipo se fortalezca. Los micro-incrementos se dan durante las iteraciones, las cuales tienen una duración de un par de semanas, al término de éstas se tiene un prototipo o demo del sistema el cual puede ser mostrado al cliente para que este último gane confianza en el equipo y si es necesario solicitar cambios menores en el sistema. En la Figura 5 se muestra lo mencionado anteriormente.

[8]Las metodologías ágiles son un conjunto de herramientas que permiten adaptar el modelo de trabajo a las condiciones del proyecto, aportando flexibilidad, eficiencia y, por lo tanto, un mejor producto a menor coste. https://www.luisan.net/blog/transformacion-digital/que-son-las-metodologias-agiles

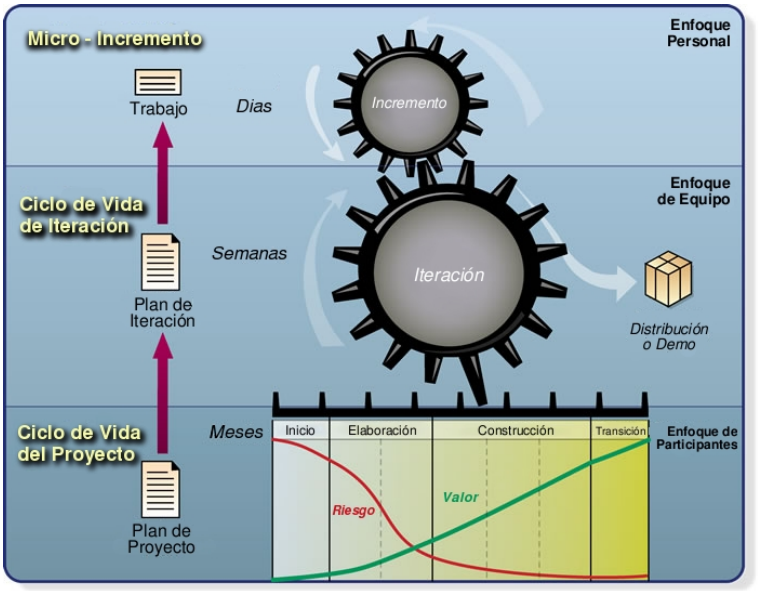

Figura 5: Organización de OpenUP. Tomado de [5].

Como se puede observar en la Figura 5 durante el ciclo de vida del proyecto existen distintas fases, en éstas se realiza lo siguiente:

1. Concepción
   En la fase de concepción el equipo se centra en entender el proyecto, además identifican las funcionalidades claves del sistema y se determina al menos una posible solución. Se calculan los costos, horarios y riesgos del proyecto.

2. Elaboración
   Durante la etapa de elaboración se obtienen más detalles sobe los requisitos del proyecto, se diseña, implementa, valida y establece la arquitectura del sistema. También los cálculos de riesgos y costos del producto son afinados.

3. Construcción
   La construcción se enfoca en el diseño, implementación y pruebas de las funcionalidades del proyecto. El propósito de esta fase es completar el desarrollo del sistema.

4. Transición
   Se hace un despliegue del sistema con los usuarios finales, la retroalimentación recibida permitirá refinar el sistema en iteraciones futuras, esta iteración también cubre el entrenamiento de los usuarios para la utilización del sistema.

2.2.4.2. Metodología Prometheus

Para diseñar y desarrollar el agente inteligente *Da Capo*, se hizo uso de la metodología *Prometheus*[23]. Ésta ha sido desarrollada durante varios años por *Lin Padgham* y *Michael Winikoff* integrantes del RMIT Intelligent Agents Group[9] en colaboración con Agent Oriented Software[10]. La metodología *Prometheus* está diseñada para que pueda ser utilizada por inexpertos en el tema de agentes. Destaca entre otras metodologías de desarrollo de software, en que soporta la creación de agentes inteligentes, y agentes BDI, además provee soporte durante el diseño e implementación éstos.

Prometheus consta de tres fases, la primera es la fase de especificación del sistema la cual consiste en encontrar las percepciones, acciones y en general qué debe hacer el sistema de agentes. La segunda fase es llamada diseño arquitectónico, es en ésta donde con base en las especificaciones encontradas en la fase anterior se define qué agentes se necesitarán en el sistema, también se definen las interacciones que tendrán los agentes entre sí, también en esta etapa se describen vagamente las funciones de los agentes. La fase tres es la de diseño detallado, en ella se describe la estructura interna de los agentes, con qué interactúa, qué datos recibe, los datos que genera, las acciones que realiza. En la figura 7 se muestra lo anteriormente mencionado.

[9]RMIT Intelligent Agents Group: es un área del grupo de ciencias de la computación de la universidad RMIT en Australia http://www.rmitagents.com/.

[10]Agent Oriented Software: Compañía de IA en Australia https://aosgrp.com/.

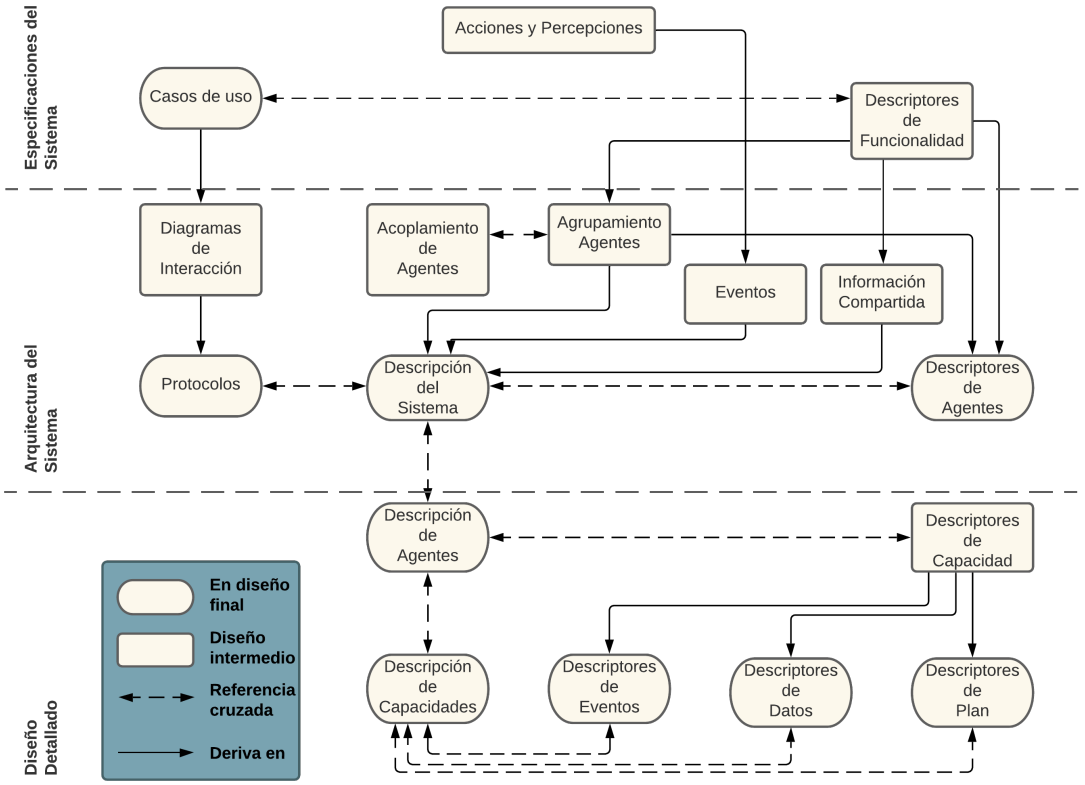

Figura 6: Vista general de *Prometheus*. Traducción realizada a partir de [6].

En la sección de la metodología, se muestra cómo se hizo uso de ambas metodologías para llevar a cabo el proyecto.

3. Enseñanza de la música mediante agentes inteligentes

3.1. Trabajo previo o relacionado

Existe una diversidad de herramientas para asistir la educación por medio de la tecnología, una de ellas es llamada *Maestoso*, esta herramienta se enfoca en personas sin conocimientos sobre música y en palabras de sus autores es un sistema de tutoría inteligente que por medio de un lápiz óptico[11] permite

[11]Lápiz óptico: es un periférico de entrada para computadoras, que permite ser utilizado para señalar objetos en un monitor, es similar a las pantallas táctiles pero este tiene mayor precisión

realizar distintos ejercicios de escritura de notas, los cuales son calificados automáticamente por el sistema, éste brinda retroalimentación y pistas de lo que deben de hacer en los ejercicios. Las lecciones que se presentan al alumno, son diseñadas previamente por humanos [7].

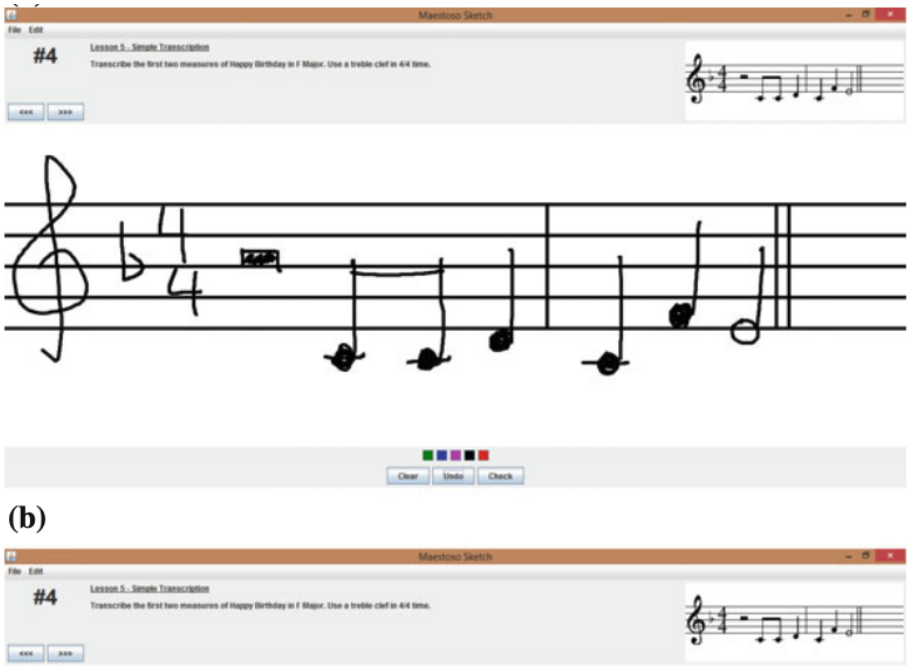

Figura 7: Vista de *Maestoso*. Imagen tomada de [7].

En el ámbito de los agentes tutores inteligentes se encuentran aplicaciones enfocadas en la música y otras enfocadas en áreas distintas.

Slang es una aplicación enfocada en la enseñanza del inglés, ha sido creada en el *MIT*[12]. Ésta hace uso de los componentes de los agentes tutores los cuales fueron descritos en la sección 2.2.1, para presentar una educación con retroalimentación constante, única y adapta a las necesidades del alumno. El sistema permite al alumno realizar actividades como: grabar oraciones, identificar objetos por su nombre, deletrear palabras, relacionar palabras con su significado, etc. Ésto permite al usuario mejorar sus capacidades auditivas, orales, escritas y conceptuales [3].

[12]MIT: Instituto tecnológico de Massachusetts, es una universidad privada ubicada en Estados Unidos.

Figura 8: Imagen de un ejercicio de *Slang*. Tomada de su sitio web.

EarMaster es un entrenador de teoría musical de paga que cuenta con más de 2500 lecciones interactivas, cubriendo los aspectos fundamentales del entrenamiento auditivo, canto a primera vista y entrenamiento rítmico. Para utilizar la herramienta, se hace uso de los periféricos del dispositivo como el teclado, audífonos, micrófono, pantalla táctil con estos se llevan a cabo los diferentes actividades que proponen. Las actividades pueden incluir decir una nota en voz alta y que el sistema detecte si es la nota correcta, identificar acordes, identificar ritmos, etc. Cabe recalcar que el sistema muestra una retroalimentación constante, la cual permite saber exactamente qué se ésta haciendo, en la Figura 9 se muestra esto. En las notas de prensa que se encuentran en el sitio de internet, no se especifica si el sistema utiliza algún tipo de tutor inteligente para seleccionar el contenido que el usuario verá. En una prueba realizada se pudo detectar, que al hacer varios ejercicios mal, el sistema los ponía nuevamente, también al hacer varios ejercicios de forma correcta el sistema terminaba la lección, esto podría indicar que hay un cierto componente que puede o no ser un tutor inteligente el cual decide el contenido que se muestra [8].

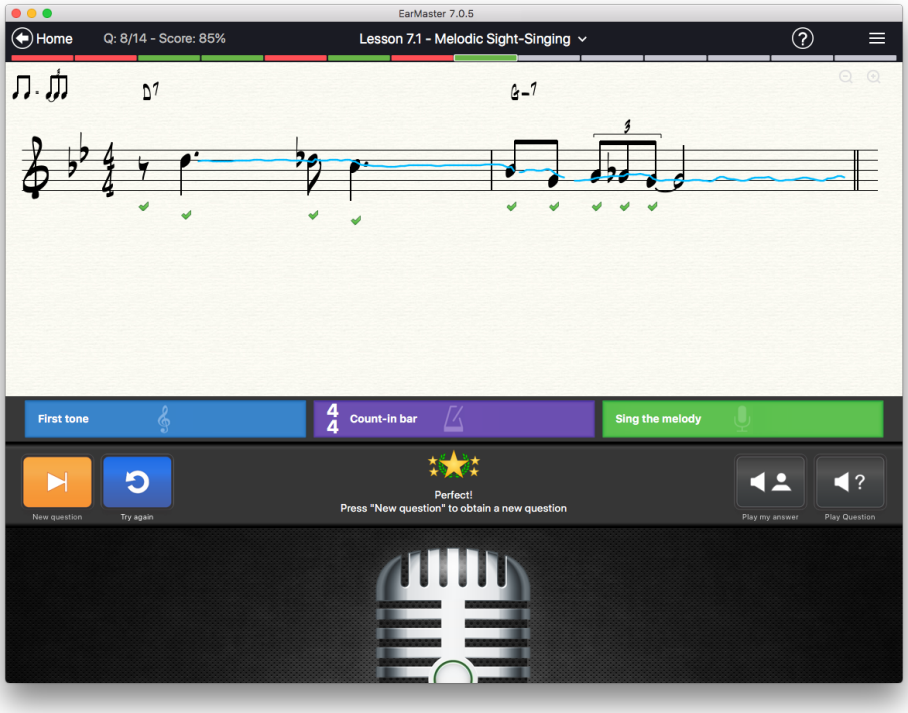

Figura 9: Imagen de *EarMaster*. Tomada de [8].

WMITS(Web-based Music Intelligent Tutoring Systems) es un sistema basado en web para la enseñanza de la música. Este sistema enseña distintos conceptos de música como intervalos, acordes y armonía. Por ejemplo, uno de los ejercicios consiste en mostrar un pentagrama en el que el alumno tiene que escribir distintos acordes, el software evalúa los acordes y dará la retroalimentación pertinente. De acuerdo con el autor del sistema, para mostrar los contenidos implementan agentes tutores inteligentes, estos definen el contenido que se mostrara al usuario. Dado que la investigación se realizó en el año 2007 el sistema quedó un poco limitado por las tecnologías de desarrollo Web de la época, una de esas limitaciones es la falta de ejercicios donde se analice la voz. El sistema parece que no ha seguido siendo desarrollado, ya que no es posible encontrarlo [9]. En la Figura 10 se muestra una vista del sistema, en donde el usuario se ha equivocado.

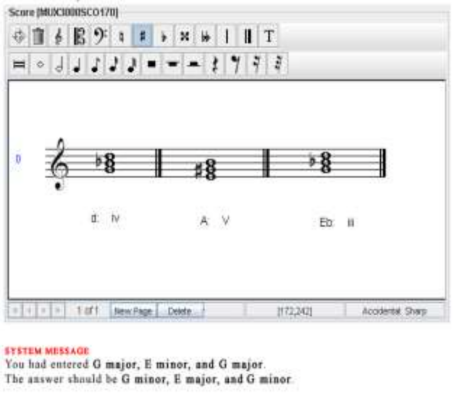

Figura 10: Imagen de *WMITS*. Tomada de [9].

3.2. Comparativa de trabajos/herramientas/plataformas

En la Tabla 1 se hace una comparativa entre los distintos sistemas educativos y Da Capo [9, 3, 7, 8].

Herramientas	Agente	Dominio	Plataformas	Libre	Tipo de agente
Slang	✓	Inglés	Web, IOS, Android	✗	Cognitivo
EarMaster	?	Teoría musical	PC, Mac, IOS, IpadOS	✗	Cognitivo
Maestoso	✓	Teoría musical	PC, XML	?	Reactivo
WMITS	✓	Teoría musical	XML, Web, MySQL	?	Cognitivo
Da Capo	✓	Teoría musical	Web-Dinámico, Jason, XML, RBDS	✓	BDI

Tabla 1: Comparativa entre distintos programas de computación educativos.

4. Objetivos

4.1. Objetivo general

Desarrollar un agente tutor con el enfoque BDI (creencias, deseos e intenciones) para la enseñanza de los conceptos básicos de la teoría musical (escalas, sonido, acordes, ritmo).

4.1.1. Objetivos específicos

1. Aplicar la metodología *OpenUP* para la planeación del proyecto.
2. Aplicar la metodología *Prometheus* para el diseño general del agente tutor y sus habilidades.
3. Diseñar los elementos pedagógicos del agente.
4. Diseñar una plataforma web para ser usada como interfaz del agente tutor.
5. Diseñar una base de datos para contener la información del agente y los usuarios.

5. Hipótesis

El enfoque BDI (creencias, deseos e intenciones) permite desarrollar un agente tutor el cual pueda enseñar los conceptos musicales de: escalas, sonidos, acordes y ritmo.

6. Metodología

Para llevar a cabo el proyecto se hizo uso de las metodologías: *OpenUP*, y *Prometheus*. La primera sirvió para planear el proyecto, ésta establece cuatro fases: inicio, elaboración, construcción, y transición. La segunda metodología se utilizó para el diseño y desarrollo de los agentes. En la etapa del diseño del agente se elaboraron diversos diagramas, en la Figura 11, se muestra una

guía de los elementos que los conforman y su significado.

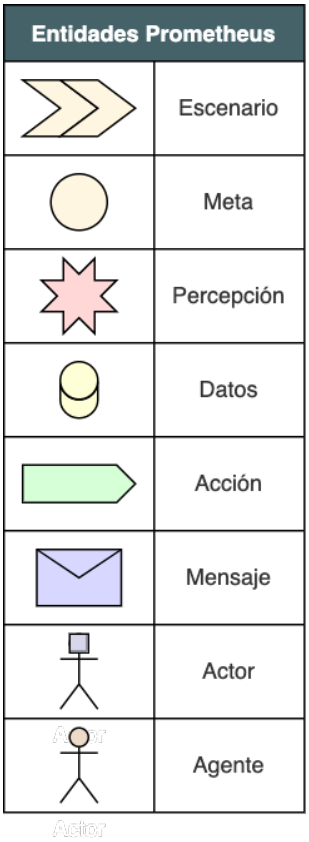

Figura 11: Elementos de la metodología Prometheus.

A continuación se explican las actividades realizadas durante el ciclo de vida del proyecto.

6.1. Inicio

Esta fase se realizó durante el Proyecto Terminal I. Las actividades realizadas estuvieron relacionadas con la redacción del protocolo. Dichas actividades consistieron en: investigar métodos para enseñar música, buscar sistemas similares al planteado, investigar conceptos, definir los objetivos y la hipótesis del proyecto. También se comenzó una búsqueda de alternativas al intérprete *Jason*, esto ya que durante el desarrollo se quería hacer uso del lenguaje de programación *Python* y *Jason* está implementado en *Java*.

6.2. Elaboración

La elaboración, se realizó a finales del Proyecto Terminal I y durante el Proyecto Terminal II. En ésta se integró la metodología *Prometheus* para el diseño del agente. En la primera fase de esta metodología llamada Especificaciones del Sistema, se realizaron actividades para identificar las metas, funcionalidades principales del sistema, escenarios, y acciones. Como resultado de esta fase se obtuvieron diversos diagramas, en la Figura 12, se muestran las metas del sistema. Los escenarios son complementarios a las metas, estos muestran la secuencia de pasos que se siguen en el sistema para realizar una funcionalidad. La Figura 13, indica los escenarios de Da Capo. En la Figura 14, se muestra un ejemplo de cómo se describe un escenario.

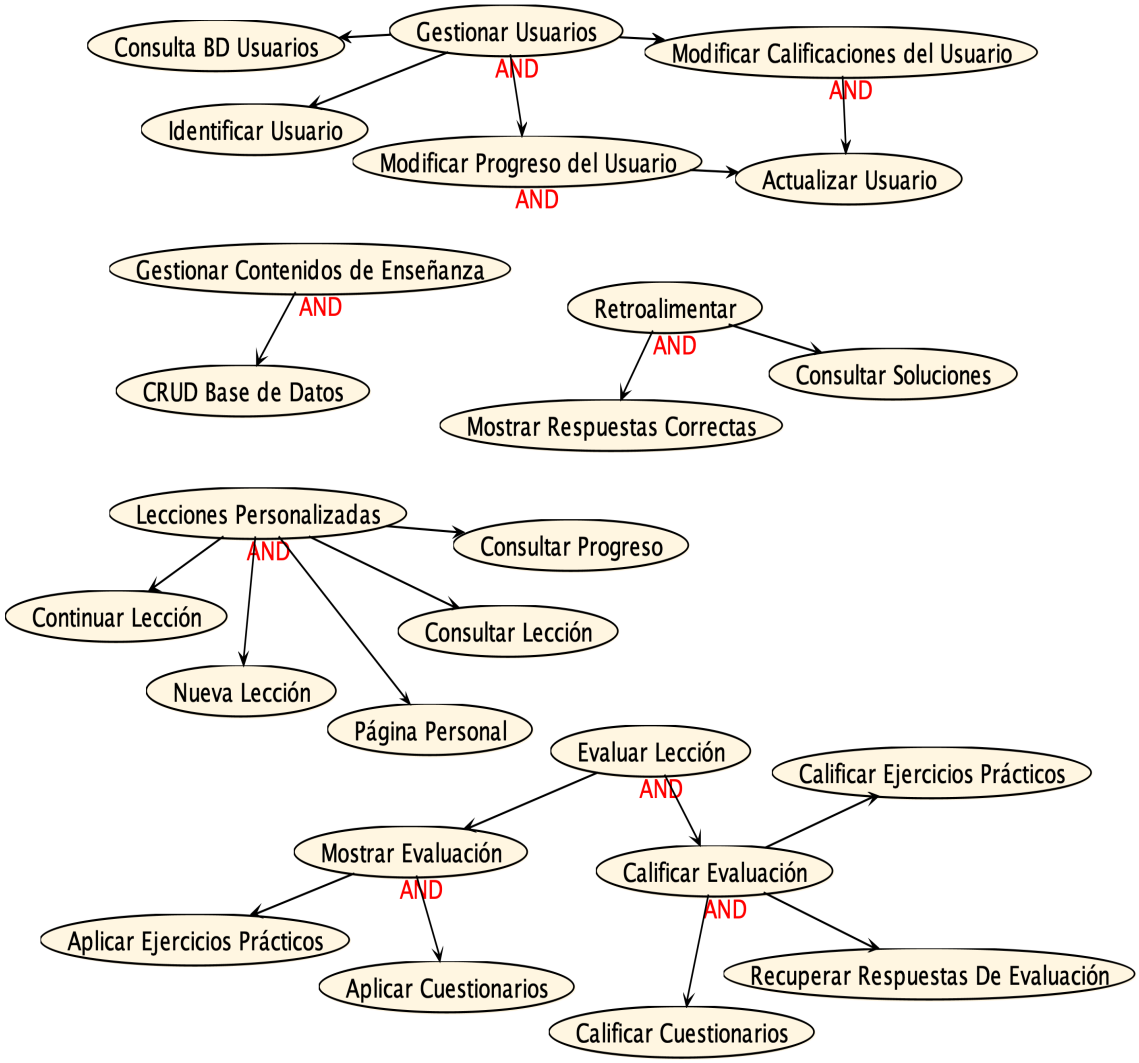

Figura 12: Diagrama de metas de Da Capo, en éste se muestran las metas principales (gestionar usuario, gestionar contenido de enseñanza, lecciones personalizadas, retroalimentar, evaluar lección) con sus respectivas submetas.

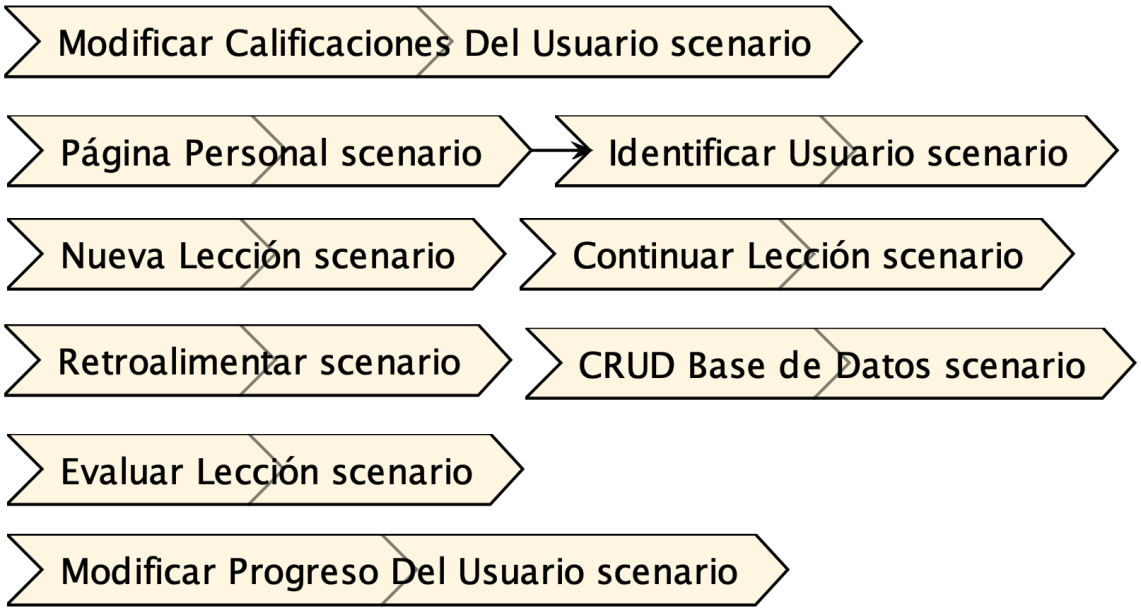

Figura 13: Escenarios de Da capo

Escenario Página Personal							
Nombre	Página Personal escenario						
Descripción	Muestra los distintos módulos al usuario, le permite continuar o iniciar una nueva lección. También despliega información relacionada con su estado academico como su progreso y calificaciones.						
Prioridad	Sin especificar						
Actores							
Iniciado por	Sistema						
Disparador	Iniciar						
Pasos	#	Tipo	Nombre	Role	Descripción	Datos usados	Datos generados
	1	Meta	Identificar Usuario		Un usuario intenta iniciar sesión en el sistema, se debe consulta la BD Usuario para ver si sus datos corresponden con los de algún usuario.	BD Usuario	
	2	Escenario	Identifica Usuario Escenario				
	3	Meta	Consultar Progreso		Se extraen los datos relacionados con el perfil educativo del estudiante.	BD Usuario	
	4	Acción	Desplegar Información Estudiante		Despliega información de interes para el estudiante, como sus calificaciones, progreso, etc.		
	5	Acción	Desplegar Módulos de Enseñanza		Despliega los módulos disponibles en Da capo		
Variación	No se encuentra al usuario. Se muestra un mensaje al usuario y se le sugiere voverl a internarlo con otro datos.						

Figura 14: Ejemplo de un Escenario

En la siguiente fase de la metodología *Prometheus*, llamada Diseño Arquitectónico, se definieron qué tipo de agentes se implementarían en el sistema. Además, se definió cómo estos iban a interactuar entre sí. La Figura 15, muestra los agentes del sistema: *Bibliotecario*, *Control Escolar*, y *Profesor*. Cada uno de estos tiene acciones y percepciones, según [23], las acciones son formas en las que el agente puede operar en el ambiente y las percepciones es la información relevante que viene desde el ambiente. En la Figura 16, se muestra cómo interactúan los distintos elementos de Da Capo. En ésta se muestran los elementos mencionados anteriormente y se agregan los escenarios que indican qué actividades tienen asignadas los agentes. También se muestran los intercambios de datos que hay entre los agentes, y se incluyen las bases de datos en donde se encuentra la información del usuario y el contenido educativo.

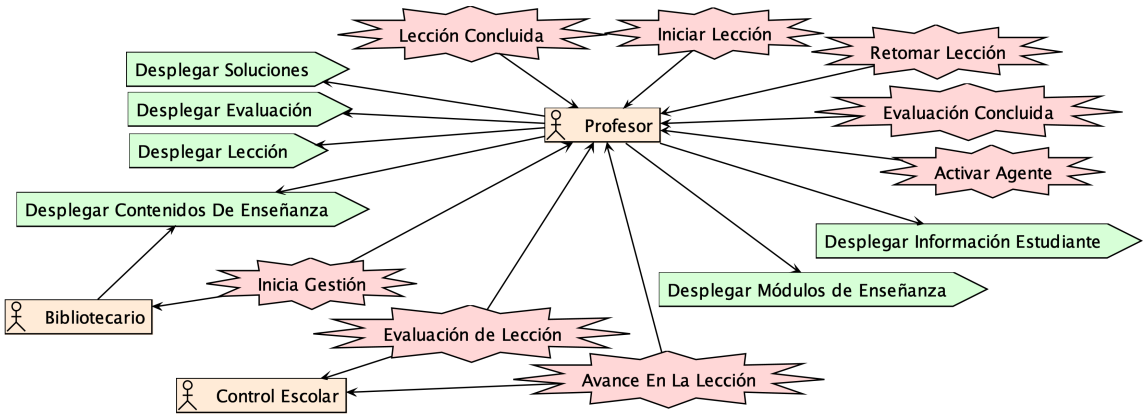

Figura 15: Diagrama general de Da Capo, en éste se listan los tres agentes, sus acciones y percepciones.

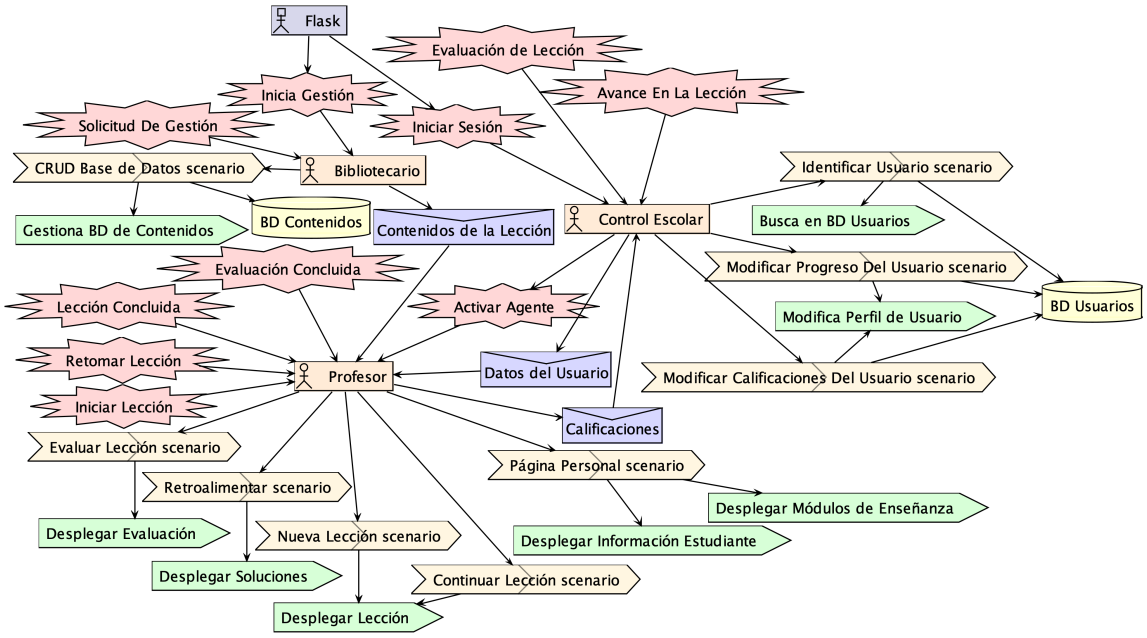

Figura 16: Diagrama de análisis general de Da Capo, en éste se puede ver cómo interactúan los agentes en su entorno.

Además de diseñar el agente con la metodología *Prometheus*, se diseñó la base de datos. Para esto, se realizó un análisis extenso del contenido educativo que podría tener Da Capo, se diseñaron objetos de aprendizaje, y con base a éstos se determinaron los datos que almacena la base de datos. Ésta se compone por 7 tablas, las cuales se muestran en la Figura 17. Es importante

señalar, que el análisis del contenido no se incluyó en el documento, dado que dentro del enfoque del proyecto es irrelevante.

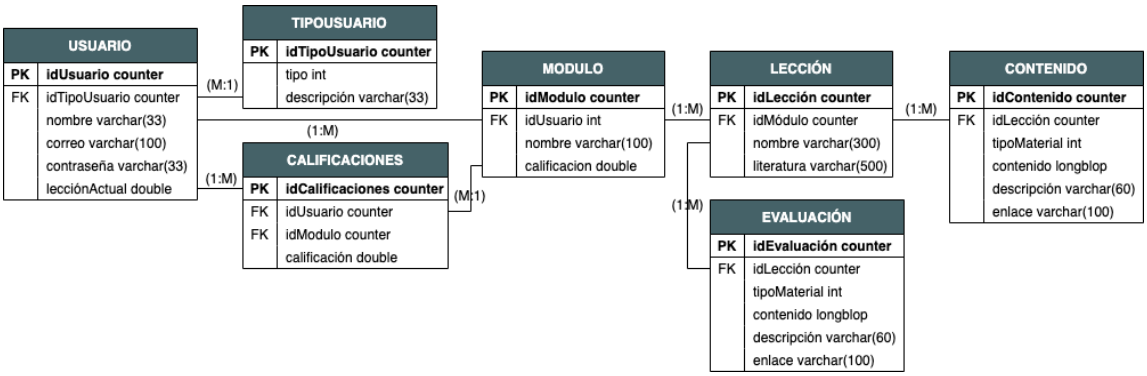

Figura 17: Diagrama relacional de la base de datos de Da Capo.

La Figura 18, muestra cómo interactúan las distintas partes del sistema durante el inicio de sesión de un usuario. En términos generales el usuario ingresa su usuario y contraseña y éstos son consultados en la base de datos, si la validación es correcta el agente inicia una interacción, consulta el progreso del alumno y los módulos disponibles en la base de datos y el resultado de las consultas se muestra al usuario.

La Figura 18, muestra cómo interactúan las distintas partes del sistema durante el inicio de sesión de un usuario. En términos generales el usuario ingresa su usuario y contraseña, estos datos son validados en la base de datos, si la verificación fue exitosa el agente se inicia. El cual recibe el ID del usuario y este consulta su progreso y el contenido educativo. Todos estos datos los envía el agente al ambiente, y son presentados en una interfaz al usuario.

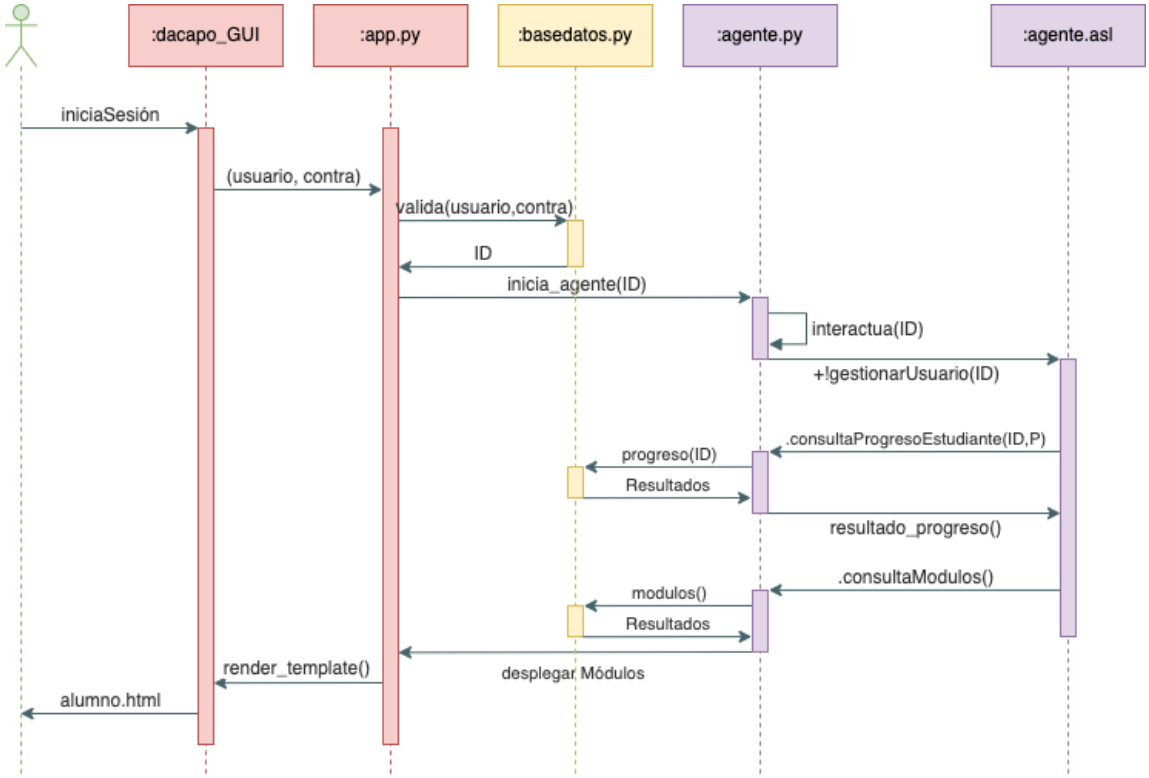

Figura 18: Diagrama de Interacción para *inició sesión* de Da Capo.

Para mostrar una lección completa, el usuario debe elegir el módulo que desea estudiar, el agente consulta y despliega la lección correspondiente. Una vez que el usuario termine la lección éste debe solicitar la evaluación, nuevamente el agente consulta y despliega la solicitud. Al término de la evaluación el agente recibe las respuestas del usuario, lo califica, y actualiza el perfil del usuario. Lo anterior se presenta con mayor detalle en la Figura 19.

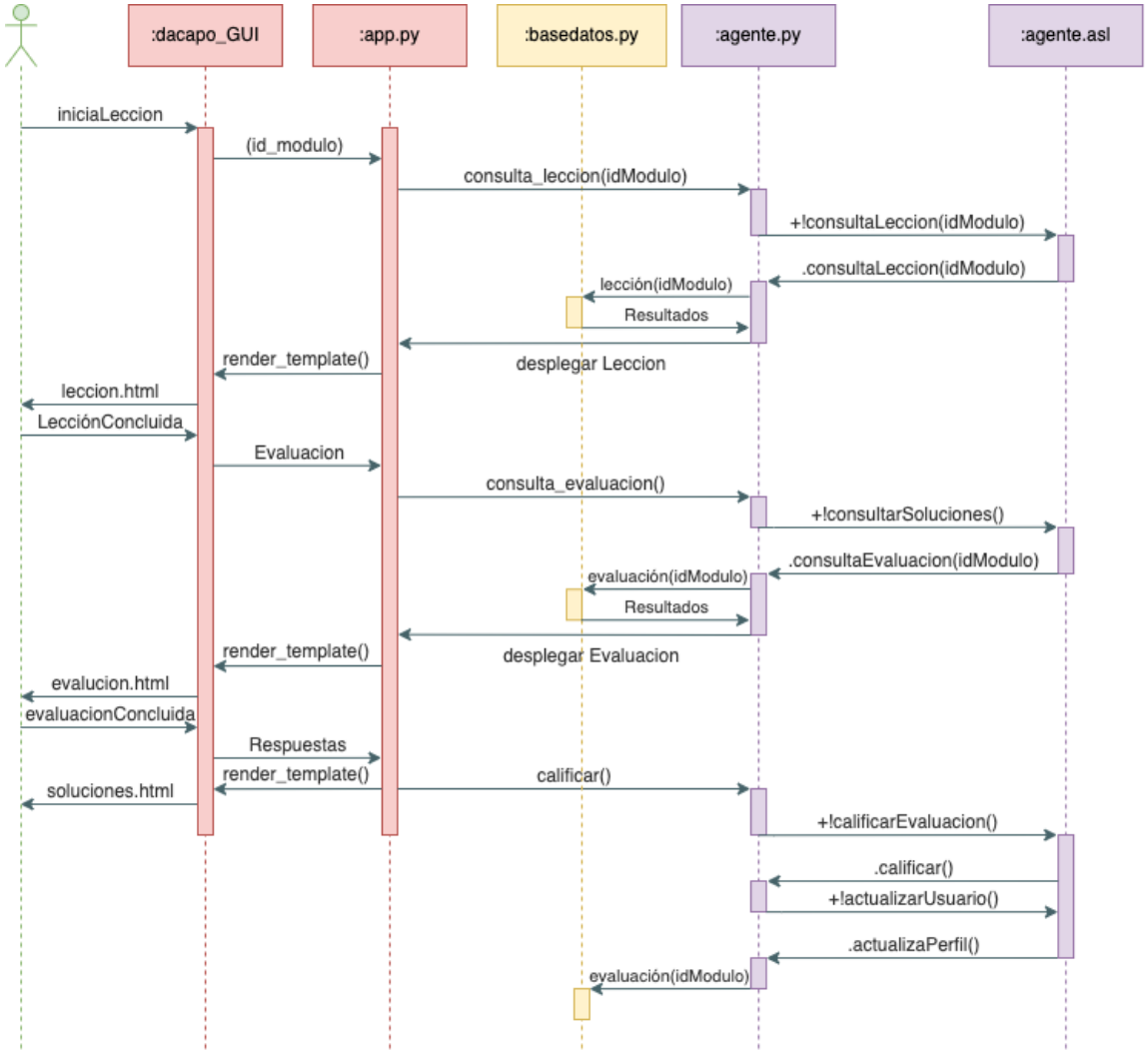

Figura 19: Diagrama de Interacción para *consulta lección* de Da Capo.

6.3. Construcción

Esta etapa se realizó durante el Proyecto Terminal II y Proyecto Terminal III. Es en ésta donde se realizó el desarrollo de *Da Capo*. Para ello, se integraron las siguiente tecnologías:

- Python3
  Como lenguaje de programación principal.

- Flask
  Se utilizó para la construcción del sitio web. Éste es un framework

desarrollado en *Python*, que emplea *Jinja2* como motor de plantillas y *WSGI* para que el servidor pueda llamar a la aplicación web escrita en *Python*.

- MySQL
  Se utilizó para gestionar la base de datos de los contenidos y la información de los usuarios.

- Python-AgentSpeak[24]
  Un interprete del lenguaje *Jason*, para el lenguaje *Python*.

- Herramientas de desarrollo web
  Como *HTML*, *CSS*, y *JavaScript*.

Dado que en *Da Capo*, *Flask* es el ambiente de los agentes, es necesario mencionar algunas de sus ventajas. *Flask*, es una herramienta fácil de aprender y de utilizar, esto permitió que durante el desarrollo del sistema se pudiera dedicar más tiempo a otras actividades. También es una herramienta flexible, que permite adaptarla a las necesidades específicas del proyecto. Además de esto, es una herramienta robusta que en un futuro en caso de ser necesario permitiría escalar el proyecto.

Durante la etapa de construcción, se escribieron cuatro archivos en *Python*, llamados *app.py*, *agente.py*, *basedatos.py*, y *variables.py*. También se escribió un archivo *ASL*, llamado *agente.asl*. A continuación se describe la finalidad de estos archivos:

- *app.py* para lo relacionado con *Flask*. En este código se encuentran las funciones encargadas de desplegar el sitio web. Los datos e interacciones realizados por el usuario son manejados y enviados al agente en este código. También se encarga de ejecutar el ambiente.

- *agente.py* encargado del control de los agentes. Dentro de este código se encuentran las funciones que reciben y mandan información al agente.

- *basedatos.py* para lo relacionado con la base de datos. La función de este archivo es contener todo lo relacionado con la base de datos, como establecer la conexión y realizar las distintas consultas.

- *variables.py* este archivo sirve como intermediario entre *Flask* y *Python-AgentSpeak*. Este contiene declaraciones de variables globales utilizadas en *app.py* y *agente.py*.

- *agente.asl* este archivo contiene los planes, metas, y acciones del agente. En este también se llama a los predicados que se encuentran en *agente.py*.

Para mostrar la página de inicio y que el usuario acceda al sistema, se utiliza la función **index** mostrada en el Código 1, en la línea 50 se llama a la plantilla de la página principal. El usuario introduce sus credenciales de acceso y pulsa el botón de iniciar. En la línea 33 y 34 se extraen los datos del formulario y son consultados por la función **validacion** en el Código 2, en el archivo *basedatos.py*. Una vez comprobada la existencia del usuario, se muestra la página correspondiente a su tipo de usuario.

```
1 # Bibliotecas
2 from datetime import timedelta
3 from flask import app, flash, Flask, redirect,
      render_template, request, session, url_for
4 import agente as ag
5 import basedatos as bd
6 import variables as g
7
8 # Instancia objeto Flask
9 app = Flask(__name__)
10
11
12 # Llave secreta para manejo de Cookies
13 app.secret_key = 'sistemadacapo'
14
15
16 # Evita que la sesión se cierre junto con el navegador y
17 # establece el tiempo que la sesión permanecerá activa
18 @app.before_request
19 def sesion():
20     session.permanent = True
21     app.permanent_session_lifetime = timedelta(minutes=10)
22
23
24 # Página principal
25 @app.route('/', methods=['GET', 'POST'])
26 def index():
27     if 'username' in session:
```

```
28          return render_template('indexSesion.html', nombre=
    session['username'])
29      else:
30          error = None
31          if request.method == 'POST':
32              # Recupera los datos del formulario
33              usuario = request.form['usrCorreo']
34              contra = request.form['usrContra']
35              # Valida los datos del formulario con la base de
    datos
36              resultado = bd.validacion(usuario, contra)
37              if resultado != 0:
38                  # Redirecciona al usuario a la página que
    corresponde a su tipo (administrador 0 /alumno 1)
39                  if resultado[2] == 0:
40                      session['username'] = resultado[1]
41                      return redirect(url_for('administrador'))
42                  elif resultado[2] == 1:
43                      session['username'] = resultado[1]
44                      session['id'] = resultado[0]
45                      return redirect(url_for('alumno'))
46              else:
47                  # Indica al usuario que los datos ingresados
    son erróneos
48                  error = 'Usuario o contraseña incorrectos'
49      return render_template('dacapo.jinja2', error=error)
```

Código 1: Función index en app.py.

```
1# Valida si el usuario se encuentra en el registro de la base
     de datos
2def validacion(usuario, contrasenha):
3      print(usuario)
4      consulta_usuario = """ SELECT USUARIO.idUsuario, USUARIO.
    nombre, TIPOUSUARIO.tipo FROM USUARIO, TIPOUSUARIO
5                      WHERE USUARIO.idTipoUsuario = TIPOUSUARIO.
    idTipoUsuario and
6                      USUARIO.correo = %s and USUARIO.contraseña
     = %s """
7      cursor.execute(consulta_usuario, (usuario, contrasenha))
8      resultado = cursor.fetchone()
9      if resultado:
10         print(Fore.GREEN + "Usuario encontrado")
11         return resultado
12     else:
13         print(Fore.RED + "Usuario no encontrado")
14         return 0
```

Código 2: Función validación en basedatos.py.

Una vez que el alumno inicia sesión, se carga la página principal del alumno esto en la función **alumno** del Código 3, en esta se hace el llamado a otra función que es la que inicia al agente (línea 6 y posteriormente línea 13). La línea 14, llama a una función de *agente.py*, la cual inicializa al agente y le indica el id del usuario. Esto se puede ver en el Código 4. En el mismo, se muestra cómo se hace el llamado al plan *gestionarUsuario*, en la línea 20. El plan se puede observar en el Código 5.

```
1 # Muestra página principal del alumno
2 @app.route('/alumno', methods=['GET', 'POST'])
3 def alumno():
4     if 'username' in session:
5         # Inicia al agente
6         agentes()
7         app.logger.info(f'Lista de modulos{g.lista_modulos}')
8         return render_template('alumno.jinja2', usuario=g.
      nombre_usuario, modulos=g.lista_modulos)
9     return redirect(url_for('index'))
10
11
12 # Crea agente y pasa el identificador del usuario
13 def agentes():
14     ag.inicia_agente(session['id'])
```

Código 3: Llamado al agente en app.py.

```
1 import agentspeak
2 import agentspeak.runtime
3 import agentspeak.stdlib
4 import app as flask
5 import basedatos as bd
6 import os
7 import variables as g
8 from colorama import Fore
9
10 # Variable global
11 agente_dice = "agente.py"
12
13 actions = agentspeak.Actions(agentspeak.stdlib.actions)
14 env = agentspeak.runtime.Environment()
15
16 agente: str
17
18 # Indica al agente que inicie el plan de gestionar usuario
19 def interactua(identificador):
20     plan = agentspeak.Literal("gestionarUsuario", (str(
      identificador),))
21     agente.call(
22         agentspeak.Trigger.addition,
```

```
23          agentspeak.GoalType.achievement,
24          plan,
25          agentspeak.runtime.Intention())
26      agente.run()
27
28  # Crea agente
29  def crea_agente():
30      global agente
31      with open(os.path.join(os.path.dirname(__file__), "agente
        .asl")) as source:
32          agente = env.build_agent(source, actions)
33      return agente
34
35  # Inicia agente
36  def inicia_agente(identificador):
37      global agente
38      agente = crea_agente()
39      env.run_agent(agente)
40      interactua(identificador)
```

Código 4: Función para activar el agente en agente.py.

El fragmento de Código 5, se encuentra en el archivo *agente.asl*. En éste, se muestra el plan **+!gestionarUsuario**, el cual se puede leer de la siguiente forma: si hay un usuario, consulta su progreso y consulta los módulos educativos. Donde **+!gestionarUsuario(ID)**, es el disparador el cual en este caso actualiza las metas del agente. La condición para ejecutar el plan viene después de los **:** y es **true**. El cuerpo del plan comienza después de **<-** y está conformado por tres acciones: **.consultaProgresoEstudiante**, **.consultaModulos** y **.print** (éste usado durante el desarrollo, para pruebas. No es relevante en el plan). Ambas acciones son ejecutadas en el archivo *agente.py*, (Código 6, línea 4 a 9 y línea 12 a 23 respectivamente), y lo que hacen es consultar a la base de datos el progreso del usuario y los módulos educativos que tiene el sistema. Es importante recalcar cómo los datos de la consulta son enviados a *Flask*, los resultados de las consultas son almacenados en las variables globales definidas en el archivo *variables.py*, y estas son leídas y usadas en *app.py*.

```
1  +!gestionarUsuario(ID): true
2      <-
3          .consultaProgresoEstudiante(ID,P);
4          .print("Progreso y nombre: ", P);
5          .consultaModulos("Consultar").
```

Código 5: Plan para consultar información del usuario en agente.asl.

```
import variables as g

# Consulta el progreso del usuario en la base de datos
@actions.add_function(".consultaProgresoEstudiante", (str, ))
def consultaProgresoEstudiante(identificador):
    resultado_progreso = bd.progreso(identificador)
    g.nombre_usuario = resultado_progreso[1]
    print(Fore.CYAN + "ConsultaProgresoEstudiante: ", g.
    nombre_usuario)
    return resultado_progreso[0]


@actions.add_procedure(".consultaModulos", (str, ))
def consultaModulos(progreso):
    print(Fore.CYAN + "Procedimiento consultaModulos")
    resultado = bd.modulos()
    g.lista_modulos.clear()
    g.lista_modulos.append(resultado)
    g.cantidad_lecciones.clear()
    for x in range(1, 5):
        lecciones = bd.contar_lecciones(x)
        g.cantidad_lecciones.append(lecciones)
    list(itertools.chain(*g.cantidad_lecciones))
    print(Fore.CYAN + "cantidad de lecciones:", g.
    cantidad_lecciones)

```

Código 6: Procedimientos del agente en agente.py.

Finalmente, están las plantillas *Jinja2*, estás son las encargadas de mostrar el sitio. Las plantillas contienen *HTML* y permiten escribir código similar a *Python* y son llamadas desde *app.py*. En el Código 7, se muestra la plantilla *alumno.jinja2*, de ésta se puede destacar la línea 19, donde se hace uso de la variable consultada por el agente que contiene el nombre del usuario. También en el bloque que va de la línea 22 a la línea 34, se emplean los datos consultados por el agente acerca de los módulos educativos. En la sección 7, en la Figura 22, se muestra el resultado de la plantilla.

```
{% extends "base.html" %}
{% block titulo %}Da Capo{% endblock %}
{% block head %}<link rel="stylesheet" href="{{url_for('
    static', filename='css/alumno.css')}}">{% endblock %}
{% block barraNavegacion %}
    <ul class="navegacion__lista">
        <li class="navegacion__item">
            <a href="{{url_for('index')}}" class="
    navegacion__link">Da Capo</a>
        </li>
```

```
9           <li class="navegacion__item navegacion--derecha">
10              <a class="navegacion__link navegacion--
    desactivado navegacion--resaltado">Inicio</a>
11          </li>
12          <li class="navegacion__item">
13              <a href="{{url_for('salir')}}" class="
    navegacion__link">Salir</a>
14          </li>
15      </ul>
16{% endblock %}
17{% block informacion %}
18      <div class="alumno">
19          <h3>{{ usuario.capitalize() }}</h3>
20      </div>
21{% endblock %}
22{% block contenido %}
23      {% for x in modulos[0] %}
24      <div class="contenido__modulo">
25          <div class="contenido__decoracion"></div>
26          <h3 class="contenido__titulo">{{x[1]}}</h3>
27<!--          <div class="contenido__progreso">
28              <div class="contenido__progreso contenido--avance
    "></div>
29              <h4 class="contenido__porcentaje">80%</h4>
30          </div>-->
31          <a href='leccion/{{x[0]}}' class="formulario__a"><
    button class="formulario__boton">Iniciar</button></a>
32      </div>
33      {% endfor %}
34{% endblock %}
```

Código 7: Plantilla página principal del usuario en alumno.jinja2

6.4. Transición

La etapa de transición, se desarrollé durante el Proyecto Terminal III. En ésta se concluyó con el desarrollo del sistema, y se redactó el documento final del Proyecto Terminal.

7. Prototipo

En esta sección se presenta el prototipo desarrollado durante el Proyecto Terminal.

Para desarrollar el proyecto, se creó un sitio web utilizando *Flask*. Para poder probar el sistema, el usuario debe ingresar a la página principal del sistema Figura 20.

Figura 20: Página principal Da Capo.

Debe iniciar sesión con las credenciales siguientes: correo: *invitado@correo.com* contraseña: *DaC4p0*. Figura 21.

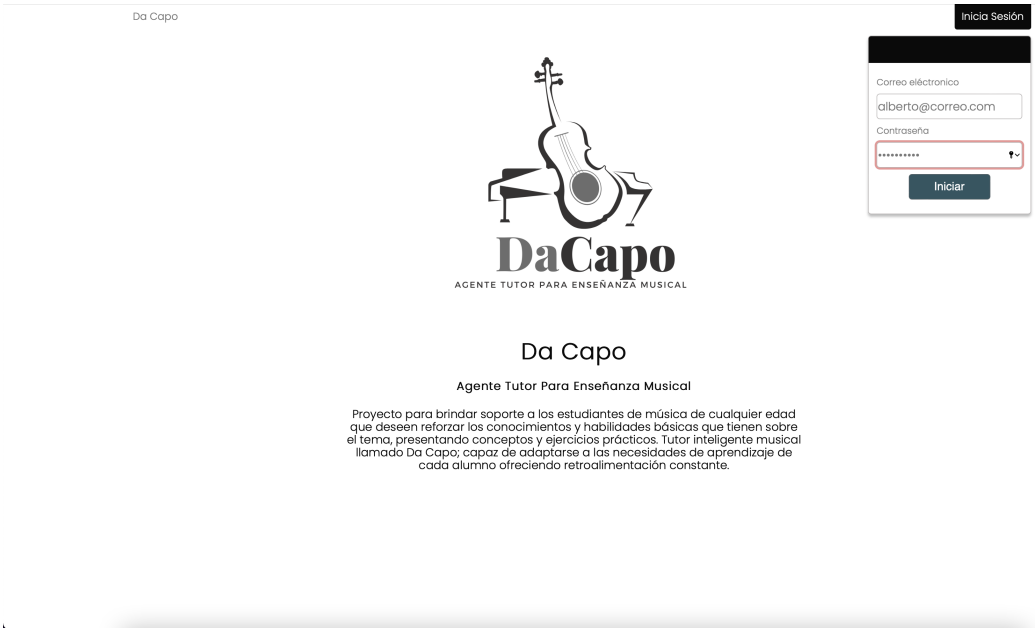

Figura 21: Página principal Da capo

Se desplegará la página principal del alumno, en esta se muestra el nombre del alumno y los módulos disponibles, Figura 22. En este caso solo el módulo llamado Generalidades de la música tiene contenido de ejemplo.

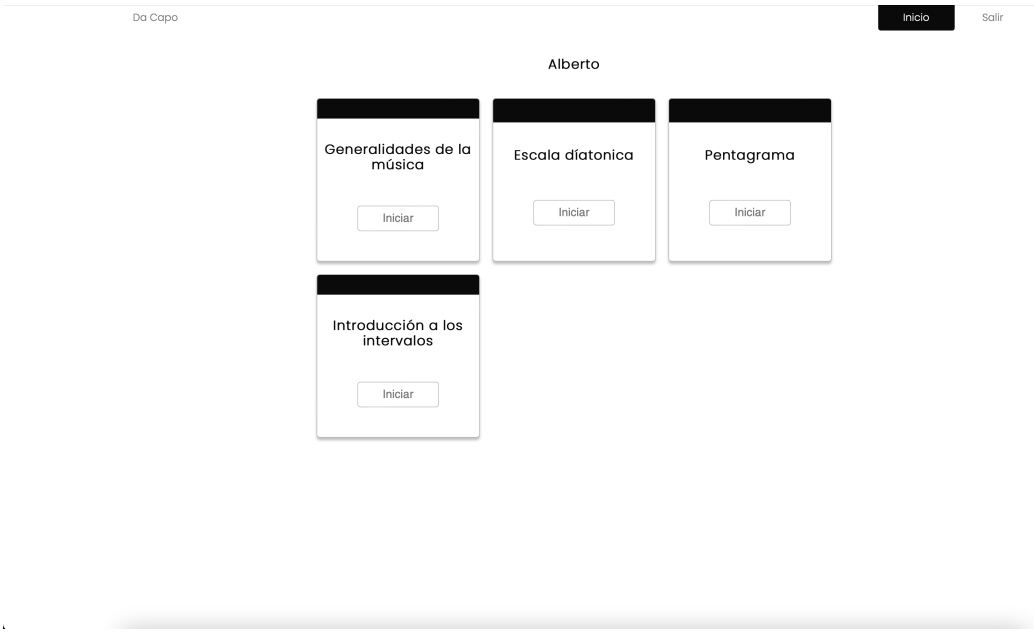

Figura 22: Página principal del alumno, Da capo.

Como ejemplo hay tres lecciones, la primera muestra texto, la segunda un video y la tercera una imagen. En la parte de abajo se muestra el botón que el usuario debe pulsar para ser evaluado (Figura 23).

Figura 23: Ejemplo de lecciones, Da Capo.

El sistema puede ser encontrado en el siguiente enlace: http://lab-ltsi.cua.uam.mx/dacapo/. Se puede ingresar con las siguientes credenciales:

- Correo: invitado@correo.com
- Contraseña: DaC4p0

8. Trabajo a futuro

El diseño de la aplicación puede mejorarse en varios aspectos. Uno de ellos es la presentación de las lecciones, en lugar de mostrar toda la lección en una página, los contenidos se mostrarían de uno en uno en tarjetas similares a las de *Slang* 3.1. Otra mejora sería en el diseño de algunos apartados del sistema, esto tratando de dar más información a los usuarios, como qué porcentaje del módulo tienen cursado, qué materiales han visto, sus calificaciones, etc.

El desarrollo de Da Capo se puede continuar en:

1. Implementar administrador.
2. Implementar resto de agentes.
3. Implementar evaluación.

Todos estos apartados ya están diseñados, solo que por cuestiones de tiempo no fueron implementados. El administrador sería la vista que el experto en la materia (música) utilizaría para agregar contenido al sistema. Los agentes cumplirían el rol para el cual fueron diseñados y la evaluación es necesaria para que el alumno conozca su progreso.

El contenido educativo del sistema, podría mejorarse en los siguientes aspectos:

1. Implementar los ejercicios prácticos, los cuales se relacionan con identificar escalas, notas, y acordes.
2. Diseñar la medida de desempeño y la función de rendimiento para evaluar el desempeño del agente.
3. Diseñar un protocolo de evaluación del agente que involucre a un experto en el dominio de la enseñanza de la música.

9. Conclusiones

De acuerdo a la hipótesis establecida, podemos decir que es posible utilizar *agentes BDI* para enseñar música. El problema radica en la complejidad

que esto supone, ya que se necesitan expertos tanto en el área de agentes como en el área de educación musical. *OpenUP* y *Prometheus*, resultaron ser compatibles para llevar a cabo el proyecto. Una de las principales dificultades del proyecto fue la integración de sistemas, puesto que las plataformas no contaban con un antecedente (como una API). *Flask* y *Python-Agentspeak*, pese a que están hechas en el mismo lenguaje, presentaron varias dificultades, como la comunicación entre el agente y *Flask*, el llamado al agente desde *Flask*, a esto se le suma integrar: *MySQL*, *HTML*, *CSS*, *JavaScript*, *WSGI*. Respecto a *Python-Agentspeak*, al ser una biblioteca creada por una persona no está correctamente documentada y esto complicó el desarrollo.
Con este proyecto se espera sentar las bases para las personas interesadas en desarrollar un sistema web que utilice agentes BDI. Esto se puede aplicar en otras áreas de enseñanza e incluso se puede aplicar a otros enfoques no relacionados con la educación.

10. Publicaciones

Resultado de este proyecto se realizó un artículo el cual fue presentado en el 15th Workshop on Intelligent Learning Environments (WILE 2022) [https://wile.mozello.com]. Éste está en proceso de ser publicado. *A. Nieto Rocha y W. A. Luna Ramírez, Da Capo a proposal for a BDI-based tutorial for teaching music, WILE, 2022.*

11. Disponibilidad de código

Para solicitar el código del proyecto, ponerse en contacto con Alberto Nieto Rocha al correo electrónico `albertonr7@gmail.com` o con Dr. Wulfrano Arturo Luna Ramírez al correo electrónico `wluna@cua.uam.mx`.

Referencias

[1] C. Abromont, E. d. Montalembert, P. Fourquet, E. Oriol, and B. Pauset, *Teoría de la música: Una guía.* 2019. OCLC: 1176225198.

[2] L. R. W. Arturo, "Agentes racionales." 2020.

[3] N. Rojas, "Queremos contarte en qué consiste el adaptive e-learning de slang y cuáles son sus ventajas."

[4] L. R. W. Arturo, "Agentes cognitivos, intencionales y bdi." 2021.

[5] R. Balduino, "Introduction to openup(open unified process)," 2007.

[6] L. Padgham and M. Winikoff, "Prometheus: A Methodology for Developing Intelligent Agents," in *Agent-Oriented Software Engineering III* (F. Giunchiglia, J. Odell, and G. Weiß, eds.), Lecture Notes in Computer Science, (Berlin, Heidelberg), pp. 174–185, Springer, 2003.

[7] L. Barreto, P. Taele, and T. Hammond, "A Stylus-Driven Intelligent Tutoring System for Music Education Instruction," in *Revolutionizing Education with Digital Ink* (T. Hammond, S. Valentine, and A. Adler, eds.), pp. 141–161, Cham: Springer International Publishing, 2016. Series Title: Human–Computer Interaction Series.

[8] "Earmaster-music theory and ear training on pc, mac, ipad and iphone," *EarMaster*.

[9] S. Phon-Amnuaisuk and C. Siong, "Web-Based Music Intelligent Tutoring Systems," pp. 231–248, Jan. 2007.

[10] S. Seinfeld, H. Figueroa, J. Ortiz-Gil, and M. V. Sanchez-Vives, "Effects of music learning and piano practice on cognitive function, mood and quality of life in older adults," *Frontiers in Psychology*, vol. 4, 2013.

[11] L. Atiaja and R. S. Guerrero-Proenza, "Moocs: Problems and challenges in higher education," 07 2016.

[12] "Making music: Teaching, learning and playing in the uk,"

[13] D. Desarrollo, M. S.A., and S. de, "La jornada: Hay gran deseo por estudiar música, mas no escuelas públicas," *La Jornada*, 05 2010.

[14] W. A. Luna Ramírez, *Botzoologia un bestiario de agentes artificiales.* 2021.

[15] K. Apaydinli, "Intelligent Tutoring Systems in Music Education," May 2020.

[16] L. Peak, "The Suzuki Method of music instruction," in *Teaching and Learning in Japan* (T. P. Rohlen and G. K. LeTendre, eds.), pp. 345–368, Cambridge University Press, 1 ed., Feb. 1996.

[17] Natalie Sarrazin, *Music and the Child.* Open SUNY Textbooks., 2016. OCLC: 1125668082.

[18] S. Da, "Research on the Design of Online Intelligent Tutoring Agents," in *2010 International Conference on Electrical and Control Engineering*, (Wuhan, China), pp. 4755–4758, IEEE, June 2010.

[19] A. M. Latham, K. A. Crockett, D. A. McLean, B. Edmonds, and K. O'Shea, "Oscar: An intelligent conversational agent tutor to estimate learning styles," in *International Conference on Fuzzy Systems*, (Barcelona, Spain), pp. 1–8, IEEE, July 2010.

[20] F. Moncada García, *La más sencilla, util y práctica teoría de la música*. México: Ediciones Framong : Musical Iberoamericana, 1966. OCLC: 970580884.

[21] D. Garcıa, G. Simari, and A. Garcıa, "Planificacion en agentes BDI," p. 5.

[22] M. J. Wooldridge, *Reasoning about rational agents*. Intelligent robotics and autonomous agents, Cambridge, Mass: MIT Press, 2000.

[23] L. Padgham and M. Winikoff, *Developing Intelligent Agent Systems: A Practical Guide*. Wiley, 1 ed., June 2004.

[24] F. Niklas, "Python-agentspeak." https://github.com/niklasf/python-agentspeak, 2017. GitHub repository.