UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

MEJORA DE UN SISTEMA DE MESA DE AYUDA PARA ESTUDIANTES DEL DCC

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL EN COMPUTACIÓN

MARCELO ESTEBAN BECERRA ALARCÓN

PROFESOR GUÍA:
JOCELYN SIMMONDS

MIEMBROS DE LA COMISIÓN:
FELIPE BRAVO MÁRQUEZ
HUGO MORA R.

SANTIAGO DE CHILE
2022

Resumen

El Departamento de Ciencias de la Computación, de la Universidad de Chile (DCC), ha tenido un aumento sustancial en la cantidad de alumnos que ingresan a la carrera. Esto implica que se generen una serie de desafíos en torno a los procesos académicos. Estos desafíos generan un aumento en la carga de los funcionarios y docentes, a su vez que hacen que no se pueda responder con la velocidad necesaria que requieren los cambios. Esto aumenta la incertidumbre por parte de los alumnos, sobre todo en la forma de abordar los diferentes desafíos de los procesos académicos. Este proyecto busca continuar la extensión de un sistema de mesa de ayuda, pensado para modernizar el trato efectivo entre alumnos, profesores y funcionarios. Mejorando su infraestructura y añadiendo funcionalidades clave para la continuidad y mejora del proyecto.

Este proyecto se realizó en 4 fases. En primer lugar se realizó un nuevo levantamiento de datos para evaluar la percepción de los alumnos del sistema actual, y adquirir nociones que permitieran hacer un diseño centrado en el usuario de las nuevas funcionalidades. A partir de estos resultados se integraron los objetivos y preferencias de los alumnos de manera explícita en el sistema, y se agruparon en categorías funcionales y cualitativas las valoraciones de los alumnos.

Luego se realizó un proceso de diseño de alto nivel a través del lenguaje de modelación i*. Este se traspasó a un diseño basado en casos de uso a través de UML. Finalmente se produjo un diseño de funcionalidades de 3 partes que contempla: las preferencias y objetivos de los usuarios, las opciones en el sistema y, las funcionalidades específicas que permiten dar cumplimento a las dichas alternativas.

Después, se realizó un proceso de análisis y reestructuración del código actual, lo que permitió añadir nuevas funcionalidades, así como mejorar el sistema existente asegurando su extensibilidad. Se solucionaron problemas de compatibilidad y *bugs* en el código.

Finalmente se implementaron nuevas tecnologías como *Celery*, que permitieron la implementación de funcionalidades de suscripción personalizada. Permitiendo a cada alumno agregar recordatorios de los procesos habilitados.

Este trabajo extiende las funcionalidades anteriores y reestructura el código existente, de manera de crear un sistema extensible y personalizable, favoreciendo la continuidad de este servicio. Logrando así los objetivos planificados.

Tabla de Contenido

Índice de Ilustraciones

Capítulo 1

Introducción

La cantidad de alumnos del Departamento de las Ciencias de la Computación (DCC), ha aumentado considerablemente en los últimos 5 años, 200 alumnos aproximadamente (212) [4], [5], [3]. Pasamos de aproxiamadamente 300 a más de 500 en pocos años. Este aumento viene acompañado de una serie de desafíos, no solo en la docencia, sino en los procesos administrativos.

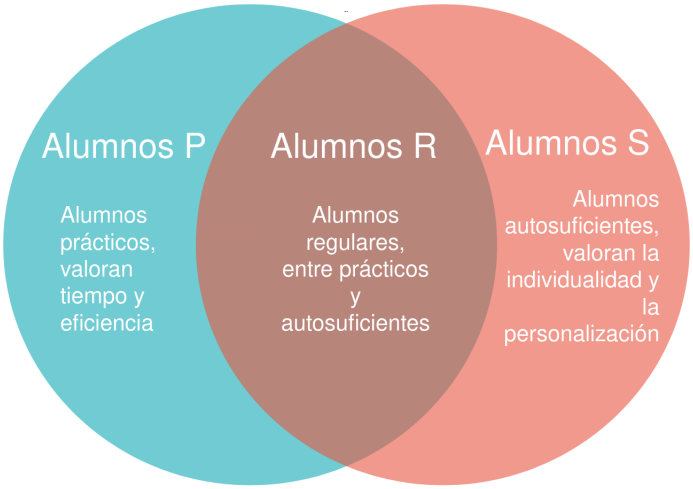

Figura 1.1: Diagrama de Venn que representa los tipos de estudiantes encontrados en el *Focus Group*.

1.1. Contexto

Uno de los focos de estos desafíos son los procesos de titulación, que también se han visto afectados por este aumento, casi el doble de alumnos en 7 años [2]. Asociados a ellos, encontramos procedimientos, reglas de formato, un tipo específico de contenido, que junto a otros elementos causan incertidumbre por parte de los alumnos. En relación con ellos hay varias fuentes de información que atañen a diversos aspectos de estos trabajos como la wiki de titulación, las páginas del centro de alumnos, publicaciones oficiales en U-Cursos, etc. A

pesar de tener estos recursos disponibles, muchas veces, la mayoría de los estudiantes suele preguntar directamente a sus profesores y, funcionarios como la secretaria del departamento suelen recibir la mayor cantidad de dudas no resueltas [2].

Una de las soluciones a considerar, podría ser ampliar personal, lo cual debe ser debidamente justificado, y consensuado [6]. Por otra parte, tampoco es lo que buscan los estudiantes, pues a la mayoría le gustaría un acceso directo a la información sin pasar por intermediarios [2]. Y es justamente en este ambiente, que el proyecto Mesa de Ayuda Virtual se posiciona como una buena alternativa para asistir a los estudiantes.

El proyecto Mesa de Ayuda Virtual, es un proyecto de soporte o asistencia digital, iniciado por Pablo Arancibia Barahona, egresado del DCC, que busca responder a la necesidad de mejorar el proceso de titulación de alumnos(as). Actualmente dividido en dos áreas: La primera es un *bot* de *Telegram*, con funcionalidades como preguntas frecuentes agrupadas por proceso, capacidad de feedback por parte del usuario y contactar a un asistente. La segunda, es una página web, que cuenta con un *front-end* para alumnos y otro para administradores, en ellos se puede categorizar preguntas, acceder y modificar los procesos vigentes con sus preguntas, dispone de chat directo con los estudiantes, entre otros

Este proyecto responde a las necesidades del estudiantado, pues fue diseñado a partir de la información levantada en el departamento y la consideración de las necesidades allí expuestas. Fue validado por la docencia y el centro de estudiantes y, ha finalizado su primera etapa de desarrollo. Actualmente, soporta más procesos como prácticas profesionales.

1.2. Problema

Como se habló anteriormente, la solución consta de dos partes vitales la web y el bot. Dentro de las conclusiones más relevantes del trabajo de Arancibia, destaca el hecho de que la mayoría de los encuestados, dijo que optaría por usar el servicio del *bot* de *Telegram*. Sin embargo, este proyecto no está implementado y presenta algunas falencias en su desarrollo que impiden hacerlo extensible, principalmente en el código del bot. En ese sentido, el que una parte vital del proyecto no pueda tener soporte es algo crítico.

Además la ausencia de personalización, en un sistema de información, podría hacer que los usuarios lo dejen de usar, ya que este es un factor clave en un sistema como este [22].

El presente trabajo busca aportar en la continuidad del proyecto, enfocándose en resolver los problemas descritos Objetivos

1.3. Objetivos

Objetivo General

El objetivo general es extender el sistema actual, agregando funcionalidades que permitan crear un sistema extensible, personalizado y confiable. El cual será actualizado principalmente por el CADCC, con miras de integrar otros actores posteriormente, favoreciendo la continuidad de este servicio.

Objetivos Específicos

1. Analizar la solución existente, con el fin de identificar qué modificaciones son relevantes, tanto para que el modelo de datos soporte las nuevas funcionalidades, como para saber la manera en que se debe reestructurar el código.
2. Rediseñar los modelos y funcionalidades, para que el sistema acepte las modificaciones necesarias.
3. Diseñar nuevos componentes de confiabilidad que permitan añadir información a las respuestas y mejorar el modelo de feedback del usuario, para que se pueda obtener más información como su vigencia.
4. Diseñar un modelo de subscripción personalizada: que contempla una búsqueda personalizada de procesos, un elemento suscriptor, variables relevantes de notificar y un modelo de notificaciones y data del alumno, entre otras *features* relevantes para los alumnos.
5. Implementar y probar las reestructuraciones, asegurando la integridad del modelo de datos, del código y permitiendo su extensibilidad a través de la modularidad lógica.
6. Implementar y probar las nuevas funcionalidades buscando que sean aquellas relevantes para los alumnos.

Evaluación

Para validar que el objetivo se cumplió, hay dos áreas que tomar en cuenta: La integridad del sistema y la validez de las nuevas funcionalidades.

En relación con la primera, se espera que las modificaciones hechas le permitan al sistema seguir cumpliendo con sus objetivos. Para lo cual se creará y usará una serie de tests que permitan asegurar sus funcionalidades críticas y la integridad del sistema. Por ejemplo, en el *back-end* probando los *end-points*, que los modelos permitan operaciones *CRUD*, entre otros.

En relación con la validez de las nuevas funcionalidades, se propone hacer tests de usabilidad, tanto entrevistas individuales como focus groups.

Adicionalmente, solo si es posible, se propone beta testing, porque permitiría integrar ambas áreas y acercar el producto al usuario. Pero esto depende de la disponibilidad del CADCC, así como de que el sistema ya este desplegado por ellos.

1.4. Solución Propuesta

Para poder llevar a cabo los objetivos anteriormente descritos, se propone agruparlos en 3 fases: Análisis, Diseño e Implementación. Pensadas en la lógica que abarca cada uno de estos objetivos.

El Análisis tiene el propósito de asegurar la mantenibilidad y rescatar las funcionalidades críticas. Para lo cual, se estudiará en detalle la solución actual. Con esto se busca identificar todos aquellos aspectos relevantes, así como aquellos que necesiten una modificación, ya sea una reestructuración o reimplementación. Entre ellos: las modificaciones en el modelo de datos, las refactorizaciones en el código para modularizar la solución actual y darle una estructura lógica, que permitan entender el sistema y hacerlo extensible.

A partir de este proceso, se diseñarán y usarán tests que sirvan para validar como el sistema actual responde a los objetivos originales y los planteados en esta propuesta. Luego con el conjunto de test diseñados, cuando existan modificaciones, se probará que no afectaron la funcionalidad original ya implementada.

Durante el Diseño, se buscará agregar elementos de confiabilidad y generar un modelo de subscripción.

Para mejorar la confiabilidad se agregará a las respuestas, la fuente de la información y su última actualización. Y junto con esto, la capacidad del usuario de retroalimentar al sistema, permitiendo ayudar a establecer métricas de vigencia reales, basadas en las calificaciones de los usuarios, así como proporcionar ayuda a los administradores del sistema, al poner atención a aquellos procesos que requieren realmente una actualización, lo que permite enfocar los esfuerzos.

En relación con el modelo de subscripción, este permitirá buscar y seleccionar un proceso de forma individual, más específicamente una instancia a un cierto proceso, ya que los procesos cómo tesis, memorias, funcionan como clases de los objetos individuales, que serían los procesos vigentes cada semestre (ver [2]). A partir de esta búsqueda, se debe permitir suscribirse no solo a procesos globales, sino que de forma más general, a información que sea relevante para el alumno, por ejemplo, deadlines, requisitos, consecuencias, entre otros. A estos elementos se les denominó en los objetivos elemento subscriptor.

Así mismo, hay que generar un modelo de notificaciones, que se ajuste a estas necesidades, y que sea lo suficientemente simple como para poder extenderlo sin problemas.

Al final de esta fase, se debe validar estos modelo con usuarios, y entidades relevantes para hacer un ajuste, antes de comenzar con la etapa de implementación. Estas validaciones deben contemplar todos los aspectos ya mencionados como nuevas funcionalidades, centrándose en la relevancia de cada una para del estudiante. Al mismo tiempo, el agregar nuevas funcio-

nalidades y personalización que dependa de datos del usuario, puede levantar problemas de privacidad que deben ser revisados junto a los encuestados.

La última fase es la implementación de las funcionalidades ya validadas. Este proceso busca integrar estas funcionalidades, tanto al *back-end* como al *front-end*. Por lo tanto, nuevamente se requerirá de un alto grado de cuidado para no romper las funcionalidades ya implementadas.

Estas *features* deben ser testeadas. Además se incluirá un proceso de validación con los usuarios que incluya demos. Opcionalmente, pero no depende del memorista, se plantea la posibilidad de incluir estas funcionalidades en un sistema de producción. Pero esto depende del CADCC y de la disponibilidad de sus miembros.

1.4.1. Documentación

Aunque la documentación no representa un desafío técnico elevado, se quiere destacar su rol, en la extensibilidad de un sistema, por lo que, se requiere tiempo, especial en cada iteración, para realizar una buena documentación del sistema.

1.4.2. Tecnologías

Si bien la decisión de qué tecnologías usar, puede verse modificada en un futuro, según los resultados del trabajo, en primera instancia se pretende continuar con el *setup* original del proyecto.
Dentro de los lenguajes a utilizar se destacan: Python, JavaScript y SQL. Estos están fuertemente ligados a las herramientas que se usan en los distintos aspectos del proyecto.

- **Servidor**: El servidor funciona principalmente en *django*, a través de la librería *django channels* se sincronizan los chats del administrador con el bot. También se pretende integrar *Celeri* para el menejo de proceso asíncronos.
- **Motor de Base de datos:** La base de datos, inicialmente pensada para preguntas repetitivas y frecuentes, usa *MongoDB*, que como herramienta *NoSQL*, está especializada en este tipo de *requests*. También, puede que sea necesario agregar otro tipo de motores en el transcurso del proyecto, ya que hay varias áreas sobre todo auditoría, en las que el modelo, no se ajusta completamente al enfoque *NoSQL*.
- **Cache sesión usuario**: Se usa *Redis*, para administrar las conversaciones a través de *Django Channels*.
- **Pagina Web**: La página web, esta principalmente diseñada en *React*, más algunas librerías para facilitar el desarrollo de componentes visuales. Se destaca, que idealmente esta no se encuentra en el scope del proyecto.

1.5. Metodologías

Para implementar lo propuesto se dividió el proyecto en 4 partes: Primero se realizó un levantamiento y análisis de datos, a partir de aquello un diseño basado en las preferencias de usuario, luego se analizó y reestructuró el código. Finalmente se realizó la implementación de las nuevas funcionalidades.

Para realizar el levantamiento de información, se convocó a dos *Focus Group*, cuyos resultados se pueden ver en 4.1. Luego se agruparon las respuestas de los estudiantes a las diferentes materias, en áreas de interés y tópicos críticos del sistema. Después, se elaboró un análisis sobre las opiniones conjuntas de los estudiantes. Con esto se hizo una clasificación cualitativa de los estudiantes, y se resumieron sus preferencias, objetivos y funcionalidades deseadas al final de la sección correspondiente.

A partir de los datos obtenidos y su análisis posterior, se generalizaron los objetivos de los estudiantes en 4 diagramas. Estos contienen sus percepciones en torno a la manera de cumplirse esos objetivos y lo que se complementa con las funcionalidades disponibles del bot, para esquematizar la manera en que se debe realizar el desarrollo de software. Las conclusiones obtenidas en el desarrollo de este análisis permitieron hacer una implementación centrada en los objetivos del usuario.

Luego se analizó en profundidad el código existente, se hicieron varias pruebas funcionales y se modificó parcialmente el código para albergar nuevas funcionalidades. A partir de este trabajo preliminar, se desarrolló una reimplementación profunda separando en capas lógicas el procesamiento de la información, lo que se dividió de las acciones que el sistema realiza a partir de las peticiones del usuario (ver5).

Finalmente se realizó la implementación de las nuevas funcionalidades de suscripción, se añadió documentación durante el trabajo y se realizaron una serie de correcciones de compatibilidad, para asegurar el resultado.

Capítulo 2

Marco Teórico

2.1. Estado del Arte

Para el desarrollo de este trabajo se hizo una investigación sobre los sistemas que solucionan problemas similares y a que objetivos responden. Al mismo tiempo como abordar el hecho de que el usuario es parte fundamental del flujo del sistema tanto en su funcionamiento como imponiendo restricciones dinámicas sobre el mismo, las que responderían a preferencias o valores de cada usuario. Por este motivo los temas abordados a continuación son *User Centered Design (UCD)*, *human in the loop*, *information systems* y *chatbots*.

2.1.1. Requisitos diseño centrado en el usuario

Este es un sistema que indudablemente se basa en el usuario y sus preferencias, es por esta razón que para que el desarrollo sea exitoso no basta con que se desarrolle de la manera correcta y asumiendo buenas prácticas, sino que debe hacerse en torno al usuario y rescatando no solo su input [16], sino sus objetivos y valores. Dentro de las maneras de enfocar un desarrollo centrado en el usuario encontramos 3 alternativas la especialista, generalista y mixto [10].

La manera especialista hace referencia a que hay 3 actores encargados del proceso, los usuarios, los encargados de UCD y el resto del equipo de desarrollo. Por otro lado, la generalista asume dos roles: el de usuario y el de desarrollador encargado de esta área.

En esta línea de *User Centered Design*, el diseño propiamente tal, se entiende como un proceso, en el que debe estar involucrado el usuario final, de forma constante. De modo que la personalización llega a ser üna pieza fundamental del diseño exitoso es aterrizar el desarrollo de *features* y herramientas sobre el valor del usuario final"[17].

De aquí se obtiene que el sistema debería ser personalizable, y rescatar lo que sea valorado por los usuarios. Dada la naturaleza del proyecto, se decide adoptar un enfoque generalista.

2.1.2. El usuario como input del sistema

Al ser un sistema que depende del usuario para determinar el curso de acción, el humano en este sistema no solo está en ciertos momentos críticos del flujo, sino que es el centro del flujo de información, modelos como este han sido validados con buenos resultados [28] recientemente.

Los sistemas de información asociados a chatbots, generalmente buscan la integración de información, este proceso es decir mostrarle al usuario una gran variedad de información de diferentes fuentes a través de una vista unificada viene asociado a varias dificultades técnicas y teóricas [18].

En este sentido, se propone que los sistemas de información que dependen de un humano son altamente complejos, porque los humanos, en este caso usuarios no solo son observadores, sino que tienen la capacidad de incidir en el ambiente no solo reaccionar al ambiente [20], sería iluso desarrollar un sistema para las personas sin considerar los objetivos de ellas de forma explícita en el diseño del sistema.

Estos descubrimientos e investigaciones ponen varias restricciones en cuanto al desarrollo de un sistema que depende los usuarios para funcionar y que al mismo tiempo está centrado en el usuario. Pero las podemos condensar en al menos 2: El diseño del sistema debe considerar de forma explícita los objetivos del usuario en cada interacción y debe integrar la información.

2.1.3. Otras aplicaciones de los chatbots en el ámbito académico

Los chatbots se utilizan desde hace varios años para variados y diversos fines, en múltiples áreas. En particular ahora los abordaremos desde la esfera académica, centrándonos específicamente en instituciones de educación superior. En estas se utilizan principalmente como un medio de apoyo al proceso de aprendizaje, como soporte para procesos universitarios, como ayuda para la gestión académica y cómo un sistema de consulta que responde a dudas de diversos entes de estas instituciones.

Como apoyo al proceso de aprendizaje los chatbots se han usado tanto dentro como fuera de la sala de clases.

En la *University of Salerno*, se creó un sistema para responder de manera efectiva las preguntas de los estudiantes, el contexto de *e-learning*. A grandes rasgos es un sistema que usa procesamiento de lenguaje natural como técnicas de ontología, para ligar las preguntas de los estudiantes al contenido disponible y de esta forma poder responder eso que los alumnos están buscando [8].

El Dr. Sherif Abdelhamid del *Virginia Polytechnic Institute and State University* también propuso un sistema de características similares [1], con las distinciones de que uso el sistema de reconocimiento de voz para traducir las dudas verbales de los estudiantes a texto y así procesarlas y que su objetivo estaba ligados a los elementos que acompañaban el estudio más que al aprendizaje académico en sí. Algunos ejemplos como cuál es la bibliografía pertinente al curso, que recursos tiene disponibles, que debería aprender, etc.

En este aspecto, investigadores del *Department of Mathematics and Computer Science, University of Kabianga, Kericho, Kenya* con miembros del *Department of Curriculum Instruction Educational Media, School of Education, Moi University, Kenya* analizaron la respuesta de los mismos profesores a la introducción de estas herramientas en la rutina de enseñanza [15].

Como soporte para procesos universitarios los chatbots se han usado para entregar el horario de clases o las clases siguientes así cómo ayudar a los alumnos a elegir ramos.

Desarrolladores de la *Bogdan Khmelnytsky Melitopol State Pedagogical University* en Ucrania, desarrollaron un servicio de 3 capas para poder implementar un servicio de ayuda al entregar el horario de clases a través de un chatbot, esto se hacía no solo con la finalidad de ayudar a alumnos que hayan olvidado o no un horario o su sala, sino también porque los horarios pueden sufrir cambios o modificaciones, además de que si bien hay alternativas simples como fotos estas se pierden en un gran volumen de elementos similares sin relación, lo que las hace difícil de volver a obtener, según la investigación [23].

En *The Open University of Hong Kong Hong Kong* desarrolladores implementaron un sistema para facilitar a los alumnos la decisión sobre qué cursos tomar a través de un chatbot [7].

En el área de los desarrollos de ayuda para la gestión académica, se encontraron desarrollos ligados a: Hacer disponibles servicios en horarios no hábiles, Mantener, monitorear y mostrar la información del alumno y Acercar la implementación de chatbots a usuarios de más alto nivel y con menos conocimiento técnico.

Los trabajos de Vaishnavi Ajay Inamdar1 y Shivanand R.D en el *BIET College* de la India, van enfocados en poder proveer servicios de gestión académica en cualquier momento, de este modo bajar la carga de los funcionarios y proveer de una *Efective GUI* [14].

Del *Engineering Rajalakshmi Institute of Technology* una profesora y sus alumnos, desarrollaron un sistema para administrar la información escolar de los alumnos y está ligado la base de datos de administración. Este sistema tenía el foco en el personal administrativo de la institución y dependía de la información entregada por el alumno.

Con el foco en los apoderados o tutores de un alumno, en la *Universitas Komputer Indonesia* A Heryandi, desarrolló un sistema de chatbots para poder desplegar las notas y el rendimiento de los alumnos a los padres usualmente menos versados en tecnología y más familiarizados con plataformas de mensajería. Por esto mismo se desarrolló un sistema con *Easy access*, es decir, con un login o un proceso de autenticación sencillo y con información conocida por los tutores [13].

En Indonesia en la *Universitas Sebelas Maret* investigadores desarrollaron un sistema con un bot de Telegram para incluir a funcionarios no expertos en el diseño de funcionalidades para la universidad. Este proveía funcionalidades como agregar botones, comandos, acciones, mensajes directos, transmitir mensajes, configuraciones y análisis [12].

Como un sistema de consulta se han desarrollado sistemas de preguntas frecuentes, algunos ejemplos son el sistema de Mesa de ayuda del DCC [2] y un trabajo de la *Manipal*

University en la India [25].

Aunque ambos son similares, responden a necesidades completamente distintas, el sistema del DCC es un proyecto centrado en el usuario, mientras que el trabajo de la India tiene un foco en la performance y el uso de IA.

2.1.4. Análisis de las aplicaciones actuales

Como se puede observar los bots cumplen variadas funciones dentro de la vida universitaria a través del mundo, apoyando a toda la comunidad escolar, tanto tutores, como funcionarios y alumnos, haciendo más accesibles, disponibles y eficientes diversos servicios, incluso apoyando procesos de aprendizaje virtual y presencial.

A pesar de la gran diversidad de funcionalidades que están presentes en estos sistemas, se logra rescatar que en la mayoría de los casos se diseña basándose en un modelo de inputs, procesamiento y respuestas. En la mayoría de los casos a pesar de ser desarrollos que buscan acercar o facilitar la vida de sus usuarios, suelen tomar un enfoque técnico y centrarse en aspectos como tener una , son desarrollos que en general carecen de un modelo que integre al usuario directamente, y se enfocan en solucionar los problemas de manera analítica o teórica. Esto podría ser una enorme dificultad para el proyecto e iría en contra de los descubrimientos en el área de *User Centered Design*.

Por estas razones se opta por tomar los buenos resultados funcionales de los chatbots, pero agregarles de alguna forma las preferencias de los usuarios de forma explícita en el diseño de los nuevos modelos de interacción.

2.2. Notacion i*

La notación i*, se usará como se explicó anteriormente para explicar no solo la interacción del usuario con el sistema, sino también que persiguen los mismos al realizar dicha interacción.

Las bases del lenguaje se pueden encontrar en la guía del lenguaje [9] y las imágenes en la wiki del lenguaje [21].

Esto se hace a través de la sintaxis de este lenguaje que detallaremos brevemente a continuación:

- Objetivos: los objetivos o *Goals* por su nombre en inglés, son en palabras simples un estado que desea alcanzar, por el usuario en relación a algo. Por ejemplo: «Tener los pasajes de avión comprados »

Figura 2.1: Objetivo i*

- Tareas: las tareas (*Task*) son acciones que el usuario puede realizar para lograr su objetivo. Ejemplo: «Buscar pasajes de avión », «Comprar pasajes en la aerolínea ».

Figura 2.2: Tarea i*

- Cualidades: o *Qualities* son maneras o elementos de valor que se quiere preservar u obviar al completar un objetivo. Ejemplo: «Rápido», en esta serie de ejemplo la frase completa se entendería como «Tener los pasajes de avión comprados rápido»o de manera «rápida ».

Figura 2.3: Cualidad i*, antes *softgoal* en i* 1.0

- Recursos: Los recursos son elementos ligados a las tareas, como una «tarjeta de crédito«en esa línea se podría «comprar los pases en la aerolínea con una tarjeta de crédito«.

Figura 2.4: Recurso i*

- Actores: Los actores son elementos principales del diagrama, realiza acciones, dispone los recursos y tiene objetivos. Pueden ser de tres tipos: un tipo general del cuando no se desea especificar, un agente que representa una instancia particular de un actor: como «Juan«o el «Bot«y un rol, que viene a ser una categoría o clase como «Estudiante«Las *boundaries* o límites relativos a un actor determinan que elementos le «pertenecen«.

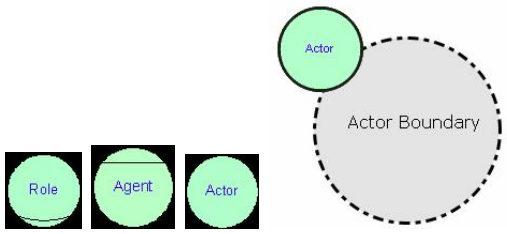

Figura 2.5: Actores i*

- Enlaces: o *Links* son lo que representa las relaciones entre los diferentes elementos del sistema, pueden representar dependencias, contribuciones, especificaciones o incluso notar una contribución. En el caso de este trabajo se usan principalmente 3 tipos de enlaces:
  - Dependencias: Se indican con una letra D y expresan que el elemento a la izquierda de la dependencia, depende del elemento a la derecha, de manera un poco simplista.

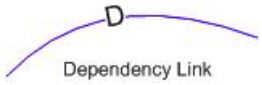

Figura 2.6: Dependencia i*

  - Contribuciones: Indican que cierto elemento contribuye a una Cualidad, puede ser de 4 formas, satisfaciendo completamente lo que se nota con *make* sobre el enlace, Imposibilitando su realización lo que se denota con *break*, o ayudando un poco o debilitando un poco lo que se denota con *help* y *hurt* respectivamente.

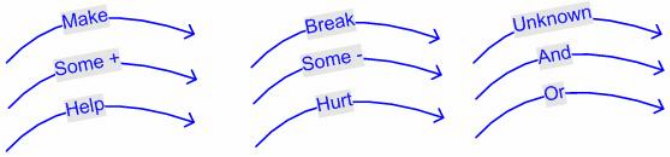

Figura 2.7: Contribuciones i*

  - Refinamiento: Expresan que un objetivo o tarea tiene otros elementos que lo definen. Pueden ser O (*OR*) o Y (*AND*), lo que determinan es que para que el objetivo se satisfaga se debe satisfacer a su vez todos los elementos que lo refinan (*AND*) o alguno de ellos (*OR*).

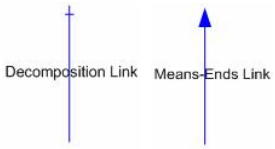

Figura 2.8: Refinamiento i*

Capítulo 3

Trabajo Previo

En este capítulo, se resumen los alcances del proyecto mesa de ayuda iniciado por Pablo Arancibia y acogido por el CADCC. Se divide en los detalles relevantes de la solución existente y los alcances de esta plataforma.

3.1. Solución existente

El proyecto iniciado el 2020, cuanta esencialmente con un diseño basado en una arquitectura de 4 partes: Database, back-end, aplicación web y el bot. El Modelo de datos se basa principalmente en la información que el sistema consume, credenciales de usuarios y feedback de los usuarios de bot. Este sistema es una gran mejora a las vías actuales de información de los alumnos, principalmente porque está diseñado para responder las dudas de los estudiantes de manera satisfactoria, mejora la comunicación entre alumnos y funcionarios, además de estar validado por la comunidad del DCC [2].

3.1.1. Arquitectura Previa

Cómo se puede observar en la figura 3.1, la capa de datos cuenta con dos bases de datos: Redis y Mongodb. Redis se usa para guardar mensajes en caché en tiempo real, enviados por ejemplo desde un estudiante a través del bot a un asistente en la plataforma de administración web. Mientras que Mongo se usa para almacenar la información relevante de los procesos, las cuentas de los usuarios, los mensajes de los usuarios de forma persistente, etc.

El back-end hecho en Django, sirve tanto al bot, la plataforma web de estudiantes y administración. Aquí se utiliza la tecnología de django channels para sincronizar mensajes desde el bot a la vista web. Django principalmente se usa para manejar las consultas a la base de datos por información, cómo el procesamiento y traspaso de mensajes desde el bot.

El front por otra parte se divide entre la plataforma de telegram y el front diseñado en react, para la vista de alumnos y administración. El Bot utiliza la API de Telegram

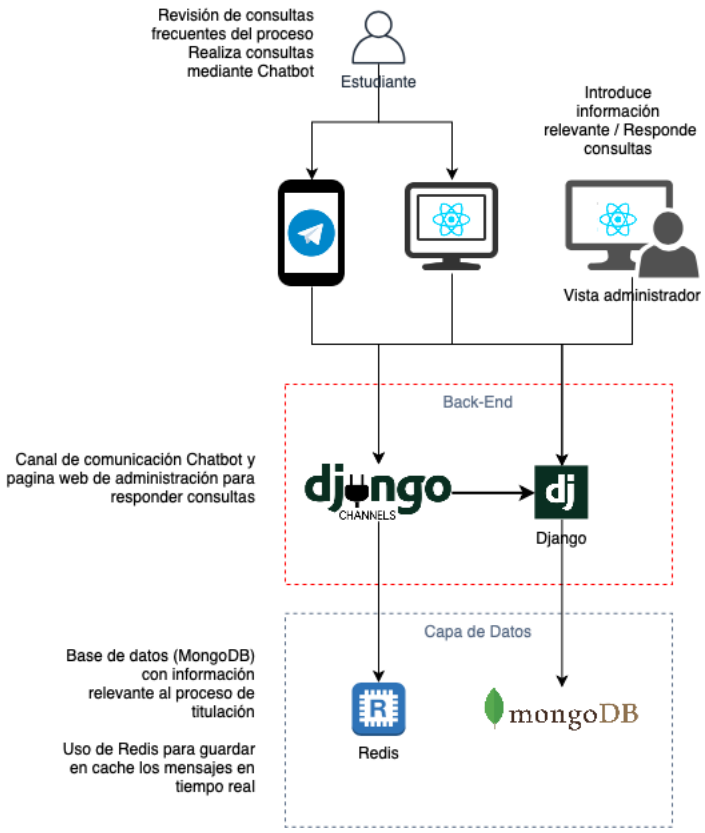

Figura 3.1: Esquema de arquitectura del trabajo realizado por Pablo Arancibia. Obtenido desde [2]

para generar vistas e interactuar con el usuario. La plataforma web por otra parte es una aplicación completa que muestra información sobre el proceso además de proveer el módulo de administración, para contestar mensajes, editar la información, entre otras.

3.1.2. Modelo de Datos Previo

El modelo de datos se puede estudiar en detalle en [2], pero a grandes rasgos. Tiene las credenciales de los usuarios de administración [1] y la información de los procesos.

La información de los procesos académicos, esta distribuida en 2 grandes áreas, aquella que es semi estática[2] en el tiempo y la que cambia continuamente.

La información relacionada con un proceso, cómo por ejemplo «Proceso de Titulación »o «prácticas profesionales »es el nombre, la descripción, las preguntas frecuentes y las categorías de estas preguntas frecuentes. Por otro lado, una instancia del proceso, cómo por ejemplo «Proceso de Titulación 2022 », tiene asociadas las novedades y las etapas de dicho proceso, ya que estas van a variar de instancia a instancia.

[1] tabla user

[2] Se dice semi estática y no completamente estática, porque la información en la base se puede modificar, sin embargo, es transversal a la fecha de consulta o de realización de un proceso.

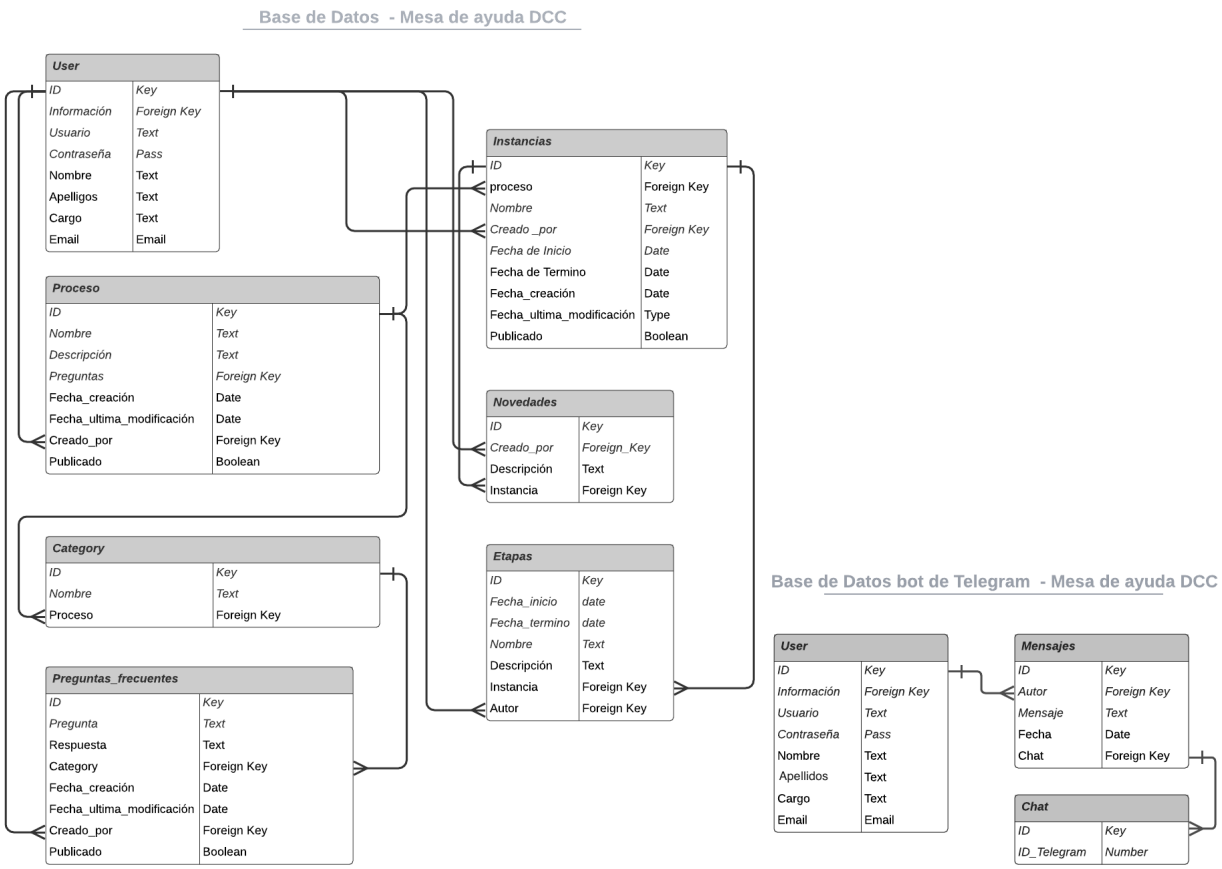

Figura 3.2: Modelo de datos del trabajo realizado por Pablo Arancibia. Obtenido desde [2]

3.1.3. Módulos lógicos del bot

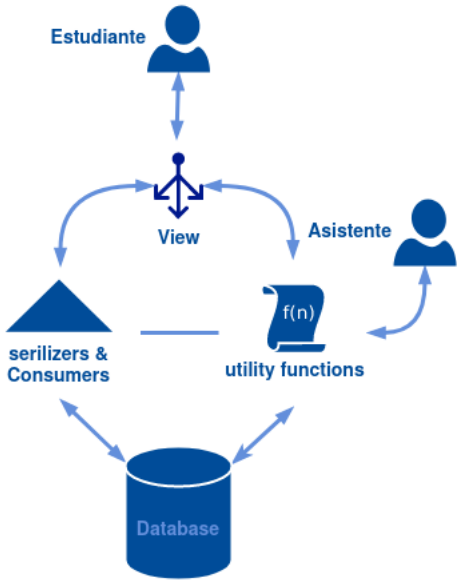

Figura 3.3: Diagrama que esquematiza el funcionamiento del bot.

la aplicación del bot, actualmente cuenta con los scripts default de Django más routing utilziado por django channels, además de *consumers* y *serializers* que se usan para seguir un estándar REST.

los *scripts consumers* y textitserializers engloban la lógica de serializar los datos y procesar las *requests* y la data anexa de json para traducirlas a Django. Por otra parte routing es utilizado en caso de que sea necesario establecer un websocket entre la aplicación web y el servidor para mantener una comunicación en tiempo real entre el administrador y el alumno.

en el modulo view, se hace el procesamiento de los *requests*, actualmente engloba la mayoria de las operaciones y funciones del bot como eviar mensajes, procesar input del usuario, extraer data para el chat desde la base de datos y guardar inputs en ella.

3.1.4. Problemas del trabajo previo

Falta de modularización

El bot, cómo estaba desarrollado, incluía la mayoría de las funciones nucleares en la *view* de Django (ver 1). Esto implica que no es claro dónde se podría modificar el código para agregar nuevas funcionalidades. Si se desea mantener el diseño, la única forma de extender el código es añadiendo nuevas funciones o modificando las existentes, sin mejorar el orden lógico. Esto tácitamente implica que para cada cambio grande se debe hacer un *refactoring* del código actual. Además lo hace poco legible, hace que trabajar simultáneamente en el proyecto implique hacer *branch* y *merges* en un sistema de versionamiento por ejemplo, a pesar de que dos personas trabajen en áreas similares. Estas deficiencias hacen que el proyecto no sea escalable, no por arquitectura física o capacidades del sistema, sino porque no porque la única forma de lograr extender el sistema de forma ordenda y manteniendo la integridad del mismo, es modificar la disposición del código existente.

Ausencia de delegación de responsabilidades

Ya que la mayoría de las funcionalidades del bot, se encuentran todas bajo el mismo *script* y además bajo el *scope* de la función principal de la view. La única forma de procesar distintos protocolos de comunicación o formas de procesar mensajes, es través de condicionales sobre el contenido del código, esto implica que la ramificación y extensión de la función principal puede crecer indefinidamente a medida que se deseen agregar nuevas formas de entendimiento, cómo comandos de telegram o procesamiento de lenguaje natural.

También implica que la lógica de acción a realizar depende de la misma capa en el sistema, por lo tanto, al ir añadiendo funcionalidades se crea una sobrecarga sobre esta parte de la API. Asi mismo resulta difícil agregar nuevos comportamientos sin extender el código existente, lo que hace aún menos claro cómo funciona el procesamiento de mensajes.

Ausencia de Documentación

El sistema actual solo cuenta con un *README*, que tiene la información esencial para ejecutar el proyecto. Pero hay nula información sobre cómo funciona el sistema, sobre cómo se pueden extender los módulos actuales, cómo se comunican las diferentes partes del sistema. Esto hace que el proyecto dependa de que las personas que trabajan en él, deban conocerse y comunicarse continuamente. En el caso de perder esta comunicación el proyecto queda virtualmente desechado, porque la configuración es lo suficientemente compleja, como para hacer difícil que alguien la tome y la extienda sin apoyo de un desarrollador del proyecto.

Por otro lado muchas de las librerías utilizadas en el sistema están bajo cambios constantes, al punto de que durante el trabajo de esta memoria cambiaron su comportamiento, incluso la documentación de lugar. Algunas se volvieron proyectos con una versión de pago, y eso hace que si alguien que no conoce estas librerías toma el proyecto, le tomará un tiempo largo entender los errores que el sistema puede arrojar. Incluso, pueden iniciarse errores inesperados o *dataraces*, lo que harían al sistema inutilizable.

Cómo uno de los objetivos es que este sistema sea adoptado por los estudiantes del departamento y extendido por ellos, estas faltas de documentación no pueden existir, porque este material es necesario para mantener, extender y mejorar el proyecto.

Finalmente, las tecnologías utilizadas son variadas y diversas, por lo tanto para alguien no familiarizado con ellas, tienen una curva de aprendizaje media alta. Cada una de estas herramientas es extensa tanto en funcionalidades como en documentación, por lo tanto, esto sumado a los cambios constantes, hacen que el proyecto no sea extensible por terceros.

3.2. Alcances de la solución existente

Durante la finalización del trabajo anterior, el trabajo con *Focus Group*, y en conversaciones con el Centro de Alumnos del Departamento de Ciencias de la Computación. Los estudiantes se han visto favorables a adoptar la plataforma [2]. Han habido varios comentarios acerca de las funcionalidades que debiera incluir, que mejoras se les podría hacer, pero en general se podría decir que la solución tendría una aceptación tal cual esta, solo que aún le faltarían funcionalidades para reemplazar la mayoría de los otros canales de información.

Si bien la solución existente es viable, aún no ha sido puesta en producción de forma definitiva, aunque está disponible en una versión de desarrollo por el CADCC.

```python
class BotView(View):
    def post(self, request, *args, **kwargs):
        try: # access field chat_id
        except Exception as e:
        try:# access field label
        self.message_processing( ... )
        return JsonResponse({"ok": "POST request processed"})

    def message_processing( ... ):
        messages = []
        if label is None:
        if message == '/start': messages = # [list of messages]
        elif message == '/preguntasFrecuentes':
        else: # contactar a un asistente
        elif label == 'Process':  # [list of messages]
        elif label == 'Category':  # [list of messages]
        elif label == 'Question':  # [list of messages]
        elif label == 'Feedback':  # [list of messages]
        elif label == 'Helper':  # [list of messages]
        for msg in messages:
        self.send_message(msg['text'], t_chat["id"], msg['keyboard'])

    # all functions below are static functions
    def get_process_keyboard():
    def get_name_process(id):
    # Similar for category, other functions
    def send_message_website(message, t_chat):
    def send_message(message, chat_id, keyboard_button={}):

```

Listing 1: Resumen del código contenido en la *View* de Django

Capítulo 4

Análisis y Diseño

Cómo se ha expresado, la tarea de la mesa de ayuda es actualmente ayudar a responder las inquietudes de los alumnos, con foco en las preguntas más frecuentes. Esto con el fin de mejorar el viaje por los procesos académicos, sobre todo mejorando la comunicación entre las diversas fuentes de información y el alumno. Esto ayuda a disminuir la ansiedad, a mejorar los tiempos de respuesta y escalar en la cantidad de alumnos que se puede atender. Este proyecto no solo abarca las preguntas frecuentes, sino que establece una comunicación con actores reales de ser necesario, además de fomentar la centralidad de la información para el alumno, y de esa forma establece como paradigma general, acompañar al estudiante de forma eficaz. Por ende, pretende ser una plataforma que se pueda consultar de forma genérica, ayudando a resolver la mayoría de las inquietudes de los alumnos, priorizando la información que requiere ser contestada por autoridades oficiales y así disminuyendo la carga sobre todo en las preguntas frecuentes que cada funcionario debe responder.

Hay variados desafíos en torno al diseño del sistema, la implementación y uso del mismo. Una de las grandes preocupaciones al desarrollar el sistema, era diseñar una infraestructura tanto lógica como de software, que permitiera abarcar no solo las necesidades de los alumnos, sino también la manera en que ellos esperan que estas necesidades sean resueltas. Ya que esto es un elemento clave en la adopción de tecnologías como los chatbots (). Además de ser el paradigma central de User Centered Design

Para ello el proceso de análisis y diseño tomó como input las valoraciones de: los alumnos, potenciales usuarios del sistema, o de exalumnos [1]. Y partir de estos resultados, se esquematizan sus necesidades para desarrollar manteniendo no solo las funcionalidades y contenidos deseados, sino también la manera en que se quieren obtener los resultados.

Entonces, para lograr este diseño holístico se realizaron: Dos Focus Groups (su metodología y resultados se detallan a continuación), un diseño basado en i* de necesidades y objetivos, y un diseño basado en UML para la capa de software. Además se llevó a cabo un proceso de refactoring del código del bot para: ser extensible, aceptar nuevas funcionalidades y separar las capas lógicas de la solución.

[1] Usuarios que ya hubieran pasado por la mayoría de los procesos académicos de pregrado

4.1. Exploración a través de Focus Groups

Una de las problemáticas del trabajo realizado, era validar con usuarios del DCC las propuestas obtenidas de la investigación teórica. Con el fin de recoger sus preferencias de manera explícita en la interacción con el sistema. Para esto, se llevaron a cabo dos *Focus Group*. En ellos se realizaron consultas en torno a la información a la que se puede y podría eventualmente acceder a través del bot, información sugerida a través del comportamiento social, el acompañamiento de terceros, la motivación para obviar el contacto directo con personas, privacidad y opciones de notificación.

4.1.1. Caracterización de los participantes

Los participantes del primer FG, son alumnos y ex-alumnos del DCC. De estos, ocho de los nueve han presentado problemas relativos a información académica, es decir, la falta de información o información errónea les ha dificultado el avance en algunos procesos académicos. Cinco de nueve, no han cursado ningún ramo relativo a la memoria, sin embargo, todos han participado de procesos con plazos como las prácticas profesionales o CC5402 Proyecto de software. En el caso del segundo *Focus Group* las características son similares, pero son solo cuatro alumnos. Adicionalmente se llevaron a cabo conversaciones con el CADCC, para evaluar las temáticas, respuestas y soluciones alternativas al sistema.

4.1.2. Respuestas de los estudiantes

A continuación, se detallan los resultados obtenidos para las categorías anteriormente descritas. Lo obtenido en cada consulta, se muestra de forma conjunta para ambos *Focus Group*, con el fin de analizar los mismos tópicos de forma única. Por lo tanto, de aquí en adelante cuando refiramos este término (*Focus Group*) nos estaremos refiriendo al proceso de consulta completo.

- **Información a la que se puede acceder a través del bot:** El contenido al que se puede acceder actualmente a través del bot, son solamente las preguntas frecuentes. En relación con las nuevas funcionalidades de suscripción se les preguntó a los estudiantes cuál era la información relevante para ellos, sobre qué cosas querían un recordatorio o sugerencias.

  La mayoría está de acuerdo en que las fechas, ya sean de apertura o cierre de procesos, incluso tareas o avances parciales serían un contenido que ellos desearían del bot, cómo una notificación. También los cambios en la información relativa a un proceso, se deberían comunicar oportunamente, por otro lado, no se debería notificar un cambio de plazo, sino ajustar los recordatorios a los nuevos plazos. Cómo sugerencias, se admiten temas relacionados con los procesos en los que se esté suscrito o se haya preguntado reiteradamente ya sea por parte del usuario o por usuarios «similares».

  Aparte de las notificaciones, se sugirió dar acceso a información relevante del proceso

en cuestión, tales como: el contacto público de personas relacionadas con el proceso[2] y ejemplos de válidos de los requisitos para cierta etapa (preinscripciones de práctica, ejemplos de propuestas de memoria).

- **Información Social:** Sobre este punto, la consulta era en torno a si estarían dispuestos a recibir sugerencias de contenido, a partir de otros usuarios con perfiles similares. Aquí las opiniones estaban divididas. Aunque en el alumnado se es reticente a compartir información con terceros, no habría problemas en que fuera información exclusivamente académica, relacionada con fines laborales o información que otros organismos ya tengan.

- **Usuario de Apoyo:** La idea de esta pregunta, era saber qué tan dispuestos están en involucrar un tercer actor que pueda ser de ayuda en un proceso académico. A este ente le denominaremos usuario de apoyo. El ejemplo, era que el profesor guía recibiera un aviso cuando el alumno no respondiera ni interactuara con el bot, durante el proceso de titulación. Las respuestas dan cuenta de que no hay un acuerdo claro en cuanto a esto. Para algunos era algo viable, para otros era algo opcional, mientras que unos pocos preferían descartar de plano la alternativa.

  Finalmente a pesar de los cuestionamientos a su funcionalidad y a la forma de usar los bots de cada alumno. Se pudo consensuar que: Fuera una opción habilitada por el usuario y con posibilidad de deshabilitar en cualquier momento, coloquialmente «cómo hacer un sudo». Así mismo, los parámetros asociados a este tercero como el contacto e información a la que accede también deberían ser definidos por el usuario. De esta forma, se elige al usuario de apoyo que cada usuario decida y se asocia al aspecto que él decida.

- **Motivación para obviar el contacto:** Esta pregunta busca profundizar en las razones de por qué los alumnos podrían preferir el bot, por sobre contacto directo con otra persona. En este caso, las razones son diferentes para cada alumno, pero concuerdan con la literatura de lo que se busca en un sistema de información [29]. Algunas de ellas son: ansiedad producto de la interacción, para no molestar, porque las preguntas son repetitivas, algunas mejoras funcionales como la rapidez, sugerencias a temas que tal vez no surjan de una pregunta directa a un encargado humano, la libertad de acudir cuando se desee y la sensación de control sobre la interacción.

- **Privacidad:** Dentro de este ítem, se preguntó específicamente por la privacidad de la información que el bot pudiera almacenar, y que *trade-off* estarían dispuestos a realizar por más funcionalidades. Aquí hay acuerdo en que debería ser la mínima información posible y en caso de ser más, solo información académica. Por otro lado, la mayoría estuvo de acuerdo en conceder permisos temporales y sin guardar información de forma permanente, en caso de que alguna funcionalidad requiera más información. De este modo se vuelve a repetir la configuración y permisos personalizados a cada alumno.

- **Opciones de notificación:** En el caso de las opciones de notificación, se puede observar que las notificaciones deberían ser: cortas con poca información y personalizables. Cada notificación debería mostrar: el área a la que pertenece o una forma de saber qué temática o relacionado con qué es esa notificación. Al mismo tiempo, debería ser capaz

[2] ejemplos serían: el correo el coordinador del CC6908, contacto de la secretaria docente, etc.

de explicar en breves líneas, que es la información nueva. En el caso de que se requiera más espacio o se necesite mostrar más contenido, debería agregarse un link o un enlace en el que se pueda desplegar a través de una página o un blog la información necesaria pero no directamente en el Chat.

4.1.3. Análisis y conclusiones

A pesar de que en el *Focus Group* se puede ver una ligera tendencia a ciertas características a partir del grado de avance de los alumnos, en general lo que se puede observar es que no hay una correlación directa entre el grado de avance y las preferencias personales de cada alumno. Así que se propone una caracterización alternativa[3], asociar a los estudiantes en tres grupos: Primero, aquellos que prefieren definir absolutamente todos los parámetros de su interacción. Segundo, los que prefieren que se les sugiera o lo que se les facilite la interacción. Tercero, sería el usuario mixto que prefiere cierto grado de personalización mientras que prefiere una simplificación de otras tareas.

Generalmente los alumnos que prefieren una alta especificación o definir cada una de las características, están también ligados a un alto grado de valoración de la privacidad y configuración manual. Se perciben a ellos mismos como autosuficientes.

Por otro lado, los alumnos que suelen tener un interés más práctico, valoran entablar la menor cantidad de interacciones posibles y valoran mucho más la eficiencia del flujo de trabajo y su tiempo. Generalmente se perciben a sí mismos como no tan atentos a las tareas y algo desconcentrados.

A partir de esto, vamos a caracterizar los tres grupos de estudiantes con tres nombres: el primero se denomina un alumno auto suficiente (*Alumno S*), el segundo vendría ser un alumno regular (*Alumno R*) y el tercero se nombra como un alumno práctico (*Alumno P*).

También podemos caracterizar la forma en que persiguen sus objetivos, los distintos actores en el sistema, esta puede variar dependiendo del proceso en el que se esté, la experiencia previa y las preferencias personales. Cabe destacar que no responden específicamente las caracterizaciones de cada uno de los alumnos es decir un *Alumno P* aun así puede querer valorar o perseguir la privacidad al consultar información y también un *Alumno S* puede perseguir la funcionalidad de los recordatorios.

Como conclusión se puede ver una tendencia a la personalización de la mayoría de los elementos funcionales, a sí mismo, una actitud selectiva en torno al contenido. En el caso de la privacidad, es más complejo. Se debería poder elegir tanto cuáles permisos son entregados y su duración. En el caso de las notificaciones, poder establecer su frecuencia en cualquier momento. En el caso de las sugerencias, se accede a entregar más información o recibir sugerencias siempre y cuando esto sea establecido por el usuario y que la información dada tenga el carácter de input temporal y no almacenado. Así mismo se establece que el usuario de apoyo debe ser definido por el estudiante. Todo esto va en línea con la investigación preliminar

[3]Esta caracterización es arbitraria y tiene el propósito de agrupar las preferencias de los alumnos. Esta agrupación se hace a partir de las cualidades que sus valoraciones generarían en el sistema. Ej: Eficiente (pocos pasos), parametrizable, entre otros.

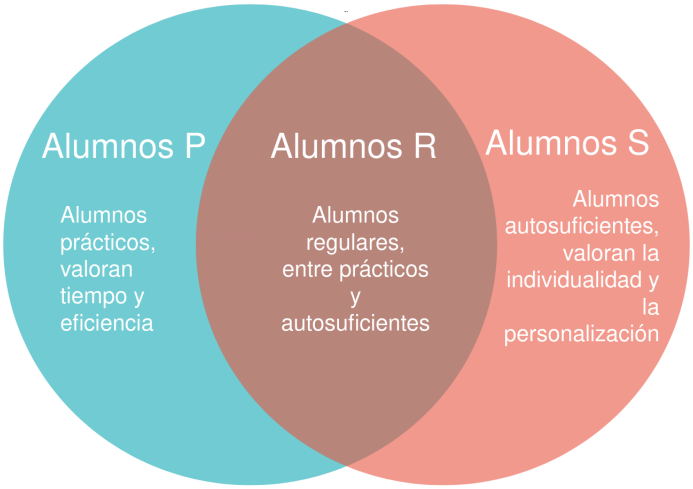

Figura 4.1: Diagrama de Venn que representa los tipos de estudiantes encontrados en el *Focus Group*.

que sugiere que los usuarios valoran la personalización y lo relevante de la información para cada usuario suele ser completamente diferente. Todo lo anterior, hace que este trabajo esté orientado en la dirección de las necesidades y preferencias del estudiantado, al mismo tiempo que va en línea con los descubrimientos y los trabajos relacionados.

4.1.4. Nuevos Modelos de Interacción

En esta sección se detalla las problemáticas de diseño de los nuevos modelos, luego se condensan los tipos de contenido que debería tener el sistema, las funcionalidades y las modalidades de interacción que surgen de las preferencias de los usuarios

Problemáticas de diseño de los nuevos modelos

A partir de la diversidad de opinión y de preferencia que se extrae de la investigación a través de *Focus Group*, podemos ver que la complejidad del modelo subyacente a tales impresiones es alta. Por la misma razón, el sistema va a tener una manera diferente de concretar los objetivos de estos actores dependiendo de cuáles sean estos objetivos y cuáles sean las tareas o requisitos asociados a tales objetivos. A raíz de esto, se hace extremadamente necesario poder especificar con claridad cada uno de los requisitos y sin embargo poder englobarse en funcionalidades que respondan de manera unilateral a las necesidades de las personas, es decir a través de la misma interfaz, la del chatbot. Por eso es que antes de involucrarnos en el diseño funcional o computacional o generar algún diagrama como el de arquitectura o de clases. Se hace necesario plasmar de manera clara las diferentes necesidades de los alumnos en el sistema, las características de estos objetivos, sus relaciones y como pueden ser llevados a la concreción.

Estas diferencias de parecer y de uso, hacen que el sistema a crear deba ser altamente configurable, es decir que cada usuario puede definir que quiere y cómo lo quiere. El cómo

responder a las necesidades de los distintos usuarios determina las funcionalidades del sistema, así mismo el qué o la propia necesidad por un contenido específico determina cuál será la información del sistema. A partir de esto, podemos resumir lo que buscan los diferentes usuarios en el sistema en dos grandes categorías contenido, es decir fechas, definiciones, actores relacionados, requisitos, consecuencias, etc. y funcionalidades, tales como recordatorios, consultas, entre otros.

Desde esta separación preliminar se proponen diferentes definiciones y diagramas, los que ayudan a encapsular y explicar de manera concisa y clara el modelo de interacción de alumnos y alumnas, sin perder el foco en los objetivos de los mismos.

Para lograr esto se propone la adopción de la simbología del lenguaje gráfico i* [9], con el fin expresar de forma correcta a los actores involucrados, sus preferencias y asimismo las tareas subyacentes que el sistema debería realizar para completar los objetivos que cada uno de estos actores tiene.

A partir de la investigación del estado del arte, la exploración a través de *focus groups* y el análisis posterior, se define lo siguiente:

Contenidos del Sistema

Los contenidos que buscan los alumnos a través del bot los podemos agrupar en cuatro categorías Definiciones, Requisitos, Fechas y Actores.

las Definiciones son la descripción o detalle del proceso, a grandes rasgos, es una explicación de se trata el proceso, como su nombre lo expresa muy bien, cumplen un rol *educativo*, que es informar al usuario de que es lo que se trata este proceso. La idea de este tipo de información es que el usuario puede entender en una sola consulta que es el proceso, de que trata, que consigue con él, de forma eficiente y amigable.

Los Requisitos son todas aquellas cosas que serán solicitadas durante el proceso, y que permiten el avance durante el mismo. El alumno busca enterarse de cuáles son estas cosas, que tareas o asignaciones debe realizar para lograrlo y ejemplos válidos de una tarea completada. Por ejemplo, un ejemplo de requisito para el proceso de práctica sería realizar la preinscripción de práctica. Al consultar en el sistema el alumno debería obtener el requisito junto a una explicación de en qué consiste y un ejemplo válido de cómo realizarla.

Las fechas son todas aquellas fecha relevantes en un proceso, en general definen plazos o hitos durante el proceso. Por ejemplo la fecha de entrega del primer avance en el CC6908

Los actores son todos aquellos entes relacionados con un proceso, tienen cuatro características que serían: el rol que cumplen dentro del proceso (el cargo o responsabilidad dentro del proceso), el contacto (medio para localizarlos), proveen de un servicio dentro del proceso (las tareas asignadas a su rol) y tendrían una asociación o un grupo al que pertenecen, por ejemplo dentro de una categoría "tipo profesor.^{es}te podría pertenece al rol de .^{A}cadémicos.^{o} Üniversidad", mientras que una institución que provee de prácticas podría pertenecer a la categoría de .^{Ex}ternos.^{o} "Lugares de práctica".

Funcionalidades relativas a la información

Las funcionalidades buscadas se puede agrupar también en 4 grupos: Consultas, Sugerencias, Intervenciones y Recordatorios.

Las Consultas son un tipo de interacción ya presente y que consiste en acceder al bot para realizar alguna consulta sobre la información de un determinado proceso, esta información podría ser cualquiera de los contenidos listados arriba.

Las Sugerencias son de un formato similar a las consultas, pero son entregadas sin ser solicitadas directamente, debe poder activarse y desactivarse, además de que no deben ser entregadas en cualquier momento, sino ojalá con una periodicidad predeterminada por el usuario, buscan ayudar al usuario a encontrar o enterarse de contenido que podría serle de utilidad sin ser buscado directamente, es decir a partir de su interacción y usuarios con características similares.

Los recordatorios son un tipo especial de notificación que es gatillada expresamente por el usuario quien les fija una periodicidad o frecuencia, buscan ayudarlo a programar y efectuar los trabajos de manera efectiva. Tienen la opción de requerir feedback si el usuario así lo desea, para motivar el compromiso.

Las intervenciones son habilitadas por el usuario y permiten al bot enviar notificaciones que alteren o podrían alterar el curso conocido del proceso, estas notificaciones pueden ser enviadas tanto al estudiante cómo al estudiante, cómo a un tercero que estudiante defina, en variados casos. Por ejemplos, los cambios en los requisitos serían una intervención, y esa modificación sería enviada al alumno. Por otro lado un cambio de fechas no siempre se categorizan como intervención, dependiendo del alumno, y del plazo. Por esta razón las intervenciones tienen un periodo de vigencia, en el que son relevantes y deberían proveer idealmente suficiente marco de acción para tomar medidas pertinentes al caso. También podrían ser provocadas por el estudiante al no responder a una interacción pactada como los recordatorios, siempre y cuando esto haya sido habilitado y parametrizado por el alumno.

Modalidades de Interacción

la mayoría de reticencias o preferencias de los alumnos, respecto a como se deben llevar a cabo las tareas asociadas a los objetivos de contenido o funcionalidad, se pueden agrupar en 5 categorías Privacidad, Parametrización, Eficiencia, Individualidad y Confiabilidad.

la Privacidad hace referencia a la información que el alumno debe entregar para obtener cierta funcionalidad o contenido. En general se busca que exista la mayor privacidad posible, pero algunas interacciones contienen un *trade-off* entre privacidad y eficiencia por ejemplo, que algunos alumnos están dispuestos a aceptar.

La Parametrización tiene que ver con la configurabilidad de la funcionalidad en cuestión, dependiendo del estudiante es más o menos valorada, y esto implica que el sistema debe dar las opciones de parametrización al tiempo que simplificarlas dependiendo del alumno.

La Eficiencia hace referencia a que tan bien puedo cumplir la tarea en cuestión, usualmente relacionada a obtener más recursos o simplificar las parametrizaciones de una funcionalidad determinada.

La Individualidad propone la autonomía del usuario en esa funcionalidad, por ejemplo, buscar contenido de memoria o de práctica sin avisar a otros actores. O no avisar inmediatamente al profesor guía de una omisión sin que el alumno haya consentido esto.

La Confiabilidad por último hace referencia tanto a la validez de la interacción como la confianza que alumno puede descargar en tal interacción. Por ejemplo, que los contactos consultados de un determinado actor estén vigentes.

Resumen conceptual del modo de Interacción

este resumen pretende expresar de manera breve, los diferentes contenidos, funcionalidades y modalidades a través de un diagrama jerárquico de clases o de conjuntos. Las funcionalidades determinan de qué forma se quiere obtener un determinado recurso del sistema, son la interfaz funcional del sistema. Por otro lado los contenidos, son los datos que el sistema debe almacenar para posteriormente servir, estos a su vez definen modelos de datos de la información del sistema. Finalmente las Modalidades hacen eco de los valores presentados en la interacción alumno-bot.

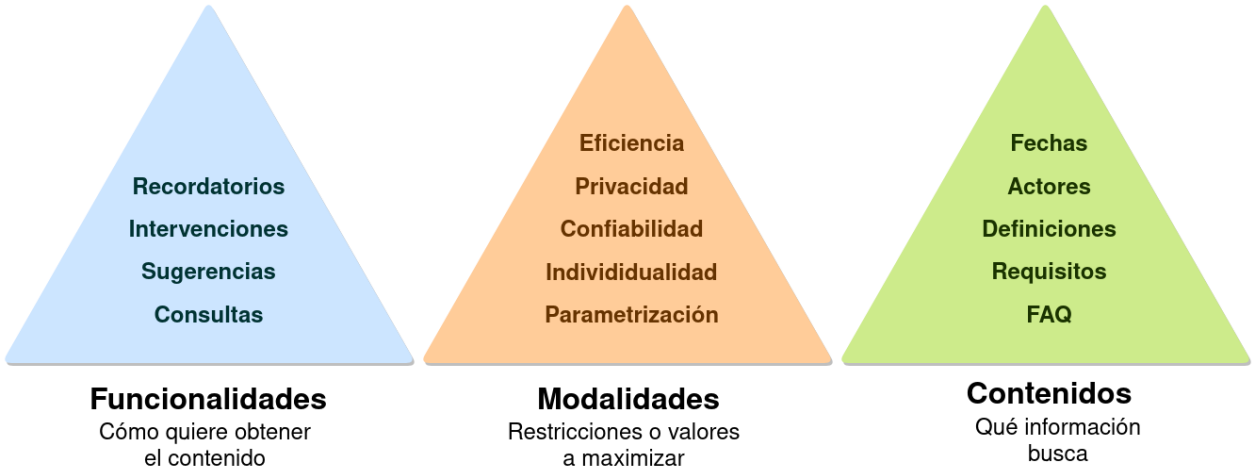

Figura 4.2: Diagrama que busca plasmar las superclases en las que se agrupan los objetivos de los alumnos a sí como las funcionalidades del sistema. Todo agrupado en categorías

4.2. Análisis de requisitos

Para poder plasmar nociones como seguridad, individualidad y personalización, es necesario interactuar varias veces con un cliente. Además, para que un diseño que integre visiones opuestas de lo que se requiere, es necesario plasmar alternativas diferentes de un mismo servicio. Estas complejidades se expresan y se detallan a través de diagramas i* (ver 1.5).

Por lo tanto los siguientes diagramas plasman las preferencias de los usuarios obtenidas de los *Focus Group*, y permiten el diseño y discusión del proceso cómo un todo. Este proceso simplifica el diseño general y permite que la discusión se centre en lo que se requiere lograr y cómo lograrlo, y no en cada camino para llegar a ello específicamente. Es decir se mantiene un alto nivel de diseño.

A continuación se explican las funcionalidades y objetivos esperados por los usuarios, a través de 4 diagramas Consultas, Intervenciones, Recordatorios y Sugerencias.

Consultas

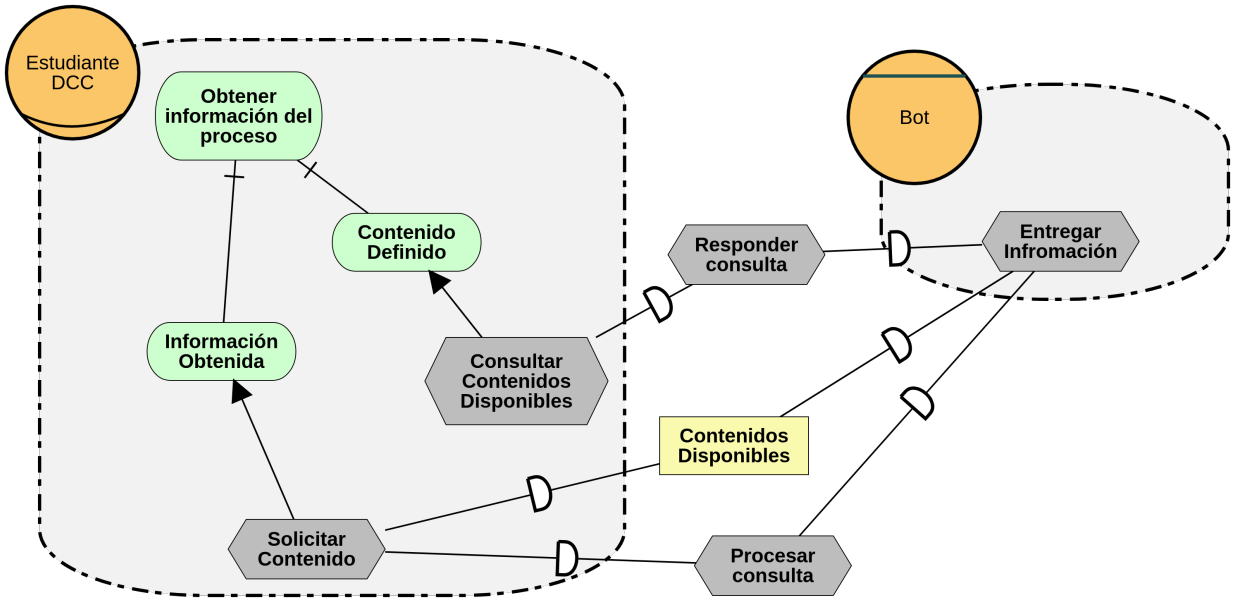

Figura 4.3: Diagrama que representa la consulta por información

Una de las principales necesidades de los estudiantes, en torno a los procesos académicos y administrativos, es obtener información. Ahora, ¿cómo se obtiene información a través del bot? En el diagrama esquematizamos este proceso.

Primero, hay que identificar el objetivo principal del estudiante del dcc (en esta interacción): Tener información del proceso (en el diagrama el objetivo superior).

Ahora para lograr esto, el usuario debe solicitar un contenido al bot, el que procesara la consulta y luego entregará el contenido disponible. Luego, el alumno podrá seleccionar qué contenido desea conocer. Cuando tiene el contenido definido, se cumple el sub-objetivo: Tener el contenido definido. Entonces puede solicitar lo que desee y el bot entregará la información del proceso seleccionado, entonces cumplirá el segundo sub-objetivo: Información Obtenida [4].

[4]En esta interacción el alumno es informado dos veces: Una de los contenidos disponibles y otra de la información en específico. Esto porque actualmente el bot no procesa leguaje natural, por lo tanto debe proveer de pasos al usuario para llegar a la información deseada.

Esta interacción simple en 3 pasos, engloba varias funcionalidades específicas. Por ejemplo los contenidos se entregan y consultan de formas diferentes. También hay ciertos contenidos que se requieren con más o menos frecuencia. Sin embargo como se puede observar, este diagrama representa lo que se quiere lograr y cómo lograrlo a modo general.

El esquematizar el problema de esta forma, permite que el diseño lógico posterior, de todas las funcionalidades que se comportan de manera similar, sean entendidas bajo una categoría común. Esto da la posibilidad de diseñar los diferentes mecanismos de una misma categoría, bajo una misma lógica, manteniendo así siempre el foco en el problema, y promoviendo una buena modularización del código. Por otro lado, permite que la comunicación y extensión de funcionalidades similares se pueda explicar de forma sencilla a nuevos desarrolladores uno de los objetivos del proyecto.

Por ejemplo el FAQ, es una forma de consultar información. Cualquier otra modalidad de consultas también recae sobre este modelo. Y a pesar de que un método de consulta, use una vía de comunicación distinta, incluso ejecuten procesamientos de datos diferentes, todos ellos responden a la misma funcionalidad. Esto permite extender esta área del sistema, añadiendo nuevas formas de preguntar y mantenerse bajo la lógica de retribuir información que el usuario pregunta dentro de un catálogo.

Recordatorios

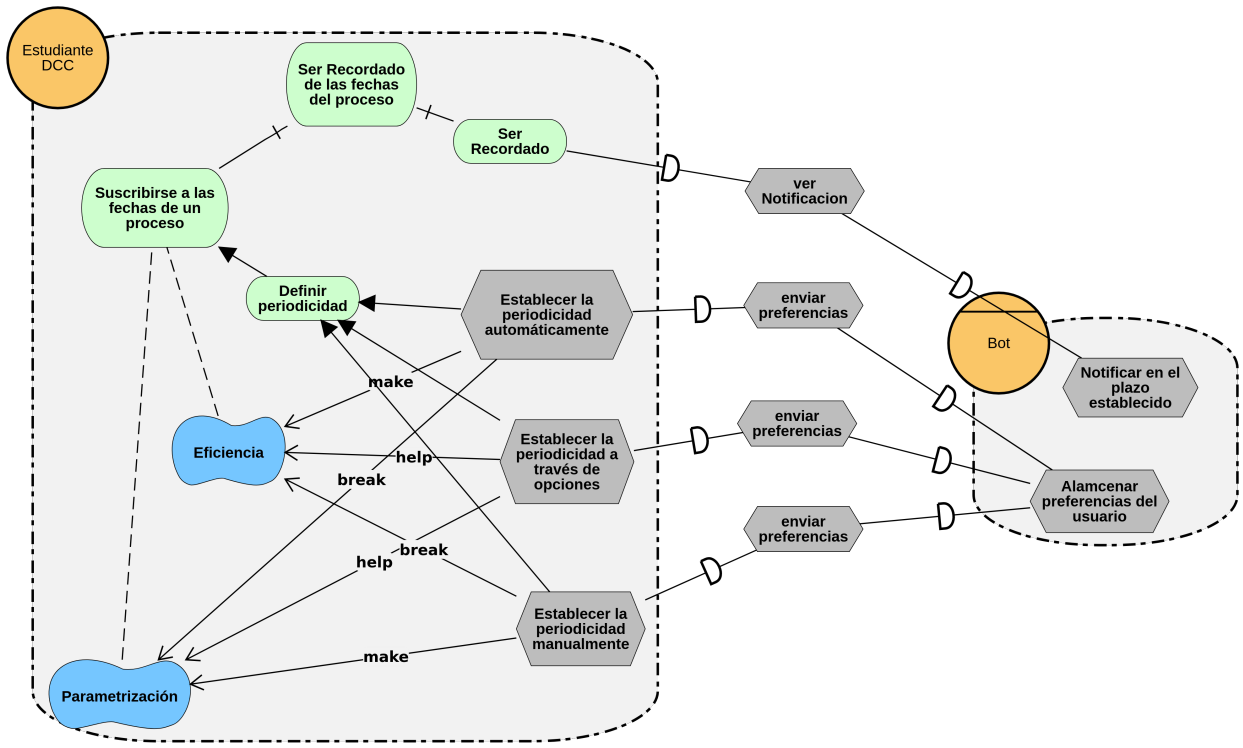

Figura 4.4: Diagrama i* que da cuenta de los objetivos y cualidades valoradas en una interacción que permita al alumno obtener recordatorios sobre fechas o plazos de un proceso académico

Otra de las funcionalidades que obtuvimos, fue la de ser recordados sobre las fechas de algún proceso académico. Cómo, por ejemplo, durante el proceso de titulación: la fecha para definir el tema, definir profesor guía, entregar el informe de propuesta, etc.

A partir del análisis también se obtuvo que existían dos cualidades o características que se valoran en cuanto a ser recordados, una es la eficiencia y otra es la parametrización [5]. La eficiencia en este caso, hace referencia a que si se realiza una suscripción a un recordatorio, se desea que el sistema sepa la manera de ser recordado. Simple, rápido y por sobre todo obteniendo el recordatorio a tiempo. La parametrización, por otro lado, tiene que ver con establecer todos los parámetros que se consideren relevantes, cómo: Cuando soy notificado, cuantas veces, etc.

En este sentido, los alumnos propusieron 3 maneras de establecer la periodicidad de una subscripción o recordatorios: De forma manual, automáticamente (el sistema define predeterminadamente cuando), a través de opciones predefinidas: 2 semanas, 1 mes, 1 día, etc. Algo que aplicaciones como outlook o google calendar han recogido y permiten en sus sistemas.

En el diagrama, si bien se puede observar que existen las tres alternativas. Cada una favorece de manera diferente las cualidades valoradas por los alumnos. Si bien, por un lado, establecer la periodicidad manualmente hace que la aplicación sea parametrizable, la hace menos eficiente. El caso contrario ocurre al establecer la periodicidad automáticamente.

Una de las conclusiones importantes que se saca de este diagrama, es que tener las 3 alternativas es la única forma de favorecer a todos los usuarios de sistema en cuanto a sus preferencias. Sin embargo, eso implica desarrollar 3 vías adecuadas especialmente para ese objetivo.

Otra conclusión interesante que se extrae de diagrama es que el bot, debe guardar una configuración específica para cada estudiante, en orden de favorecer las cualidades que son importantes para él. Si bien esto podría eventualmente ser un riesgo de privacidad, tanto desde lo extraído de los alumnos como lo la forma en que está creado el bot, hacen que se puedan guardar estas preferencias de forma anónima.

Y finalmente, como es de esperarse, el bot debe ser capaz de notificar también en una fecha particular a cada alumno.

Este diagrama, si bien en pocos pasos, da cuenta de una complejidad subyacente en torno a diseñar e implementar una funcionalidad puede hacer que un grupo de alumnos se vea favorecido y otro perjudicado, estas conclusiones son relevantes a la hora de priorizar o establecer metas de desarrollo.

Sugerencias

Las sugerencias son un tipo de funcionalidad pensada en alumnos que no tienen claro que deben hacer, se adelanta a las posibles dudas del alumno y entrega recomendaciones sobre alguna de las alternativas disponibles.

[5] la definición de cada una se puede ver en la sección anterior

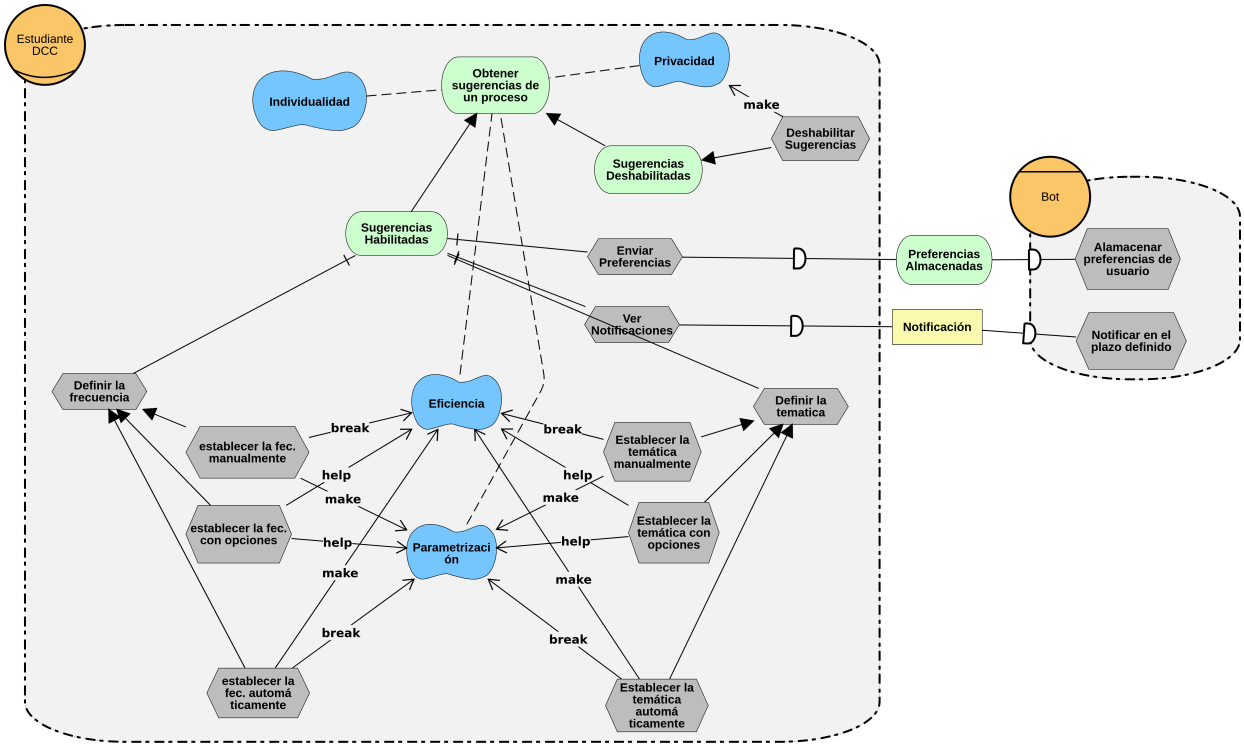

Figura 4.5: Diagrama i* que da cuenta de los objetivos y cualidades valoradas en una interacción que permita al alumno obtener sugerencias sobre alguna temática de proceso académico

El gran objetivo de Tener sugerencias de un proceso, depende de Tener las sugerencias habilitadas o no tenerlas. El último caso es sencillo, ya que el alumno partiría con las preferencias deshabilitadas. Habilitarlas por otro lado, implica enviar sus preferencias, definir la temática y definir la frecuencia.

Para definir la frecuencia y la temática se encontró que el *Alumno P* deseaba que fuera de forma automática, mientras que el *Alumno S*, prefería que fueran de forma manual. Las opciones son un término medio.

Esta fue una de a las funcionalidades más contradictorias en términos de lo valorado por los alumnos y esto se puede observar en las cualidades adjuntas a esta funcionalidad, Individualidad, Privacidad, Parametrización y Eficiencia.

Individualidad. En primer lugar, este es un sistema particularmente individualizado, esto quiere decir que las sugerencias son específicas a cada alumno, por ejemplo, uno quieren recibirlas de forma automática, otros prefieren configurar algunas cosas y algunos de lleno no quieren recibir sugerencias.

Privacidad. Esta funcionalidad es recibida con un poco más de recelo por parte de los mismos estudiantes. Porque para que el sistema pueda dar sugerencias, debe tener más información de los alumnos, la que podría hacerlos identificables y con ellos se derivan las problemáticas que podrían surgir a partir de exponer las preferencias de cada uno. Esto incluye al alumno auto suficiente y al alumno regular.

Eficiencia. Por otro lado, esta capacidad del sistema se prefiere muy automatizada por los alumnos que deseaban más guía y asesoría durante el proceso (*Alumno P*). Varios prefieren que alguien se adelantara a los posibles requisitos de cada etapa.

También se observa una discusión similar a los recordatorios, en cuanto a la parametrización y eficiencia. Sobre todo entre los alumnos tipo *Alumno P* y *Alumno S*.

Ahora algo interesante, aunque muy similar a establecer los plazos. Sería establecer la temática de las sugerencias. Uno de los grandes problemas que el bot busca resolver, es el hecho de que a pesar de tener mucha información disponible, los alumnos continúan con dudas e incertidumbres. Esto debido a dos razones principales, una es el exceso de información no relevante y la otra el desconocimiento de los canales existentes. Un sistema que agrupe la información sería valioso para solucionar el problema de tener múltiples canales, y una mejora significativa en la experiencia se daría si los alumnos reciben información que es importante para ellos y no cualquier cosa.

Esta es una discusión particularmente especial, porque es similar a las disyuntivas que presenta el desarrollo de redes sociales, y este aspecto del bot se ve muy influenciado por las percepciones que los alumnos tienen de las políticas establecidas en cada una de ellas.

Intervenciones

Las intervenciones, soporte o tutelaje tienen el objetivo de incluir a un tercero en el recorrido del alumno por un proceso, y que este pueda ser una voz amigable para revisar el estado del proceso. Este "Tutor" cómo se denomina, es notificado cuando el alumno no interactúa con los elementos a los que el mismo se suscribió, esta funcionalidad, busca que el alumno pueda recibir apoyo cuando por diversas razones podría no estar dándole la atención necesaria al proceso. Esta funcionalidad, al igual que las sugerencias, se adelanta a que el alumno pudiese tener un problema o atrasarse en los plazos.

Cómo se puede ver en el diagrama, las cualidades de individualidad y privacidad son factores determinantes a la hora de decidir si usar o no esta funcionalidad. Por otro lado la tranquilidad que puede brindar al estudiante el tener un soporte externo que puede ser cualquier persona, es uno de los mayores incentivos para agregar esta feature.

Otra de las razones por las que puede ser controversial es porque debe almacenar interacciones del alumno en el sistema, se deben guardar contactos de terceros, etc.

4.3. Conclusiones generales de Análisis

A modo general, desde los diagramas se pueden obtener varias nociones que pueden guiar el trabajo. Se presentan las funcionalidades a realizar por el sistema en las tareas que deben realizar los distintos agentes involucrados. Y, por otro lado, las cualidades vienen a ser restricciones y alternativas a las mismas funcionalidades. También algunos de los contenidos quedan explícitos como recursos. Así mismo todos los actores relevantes para una funciona-

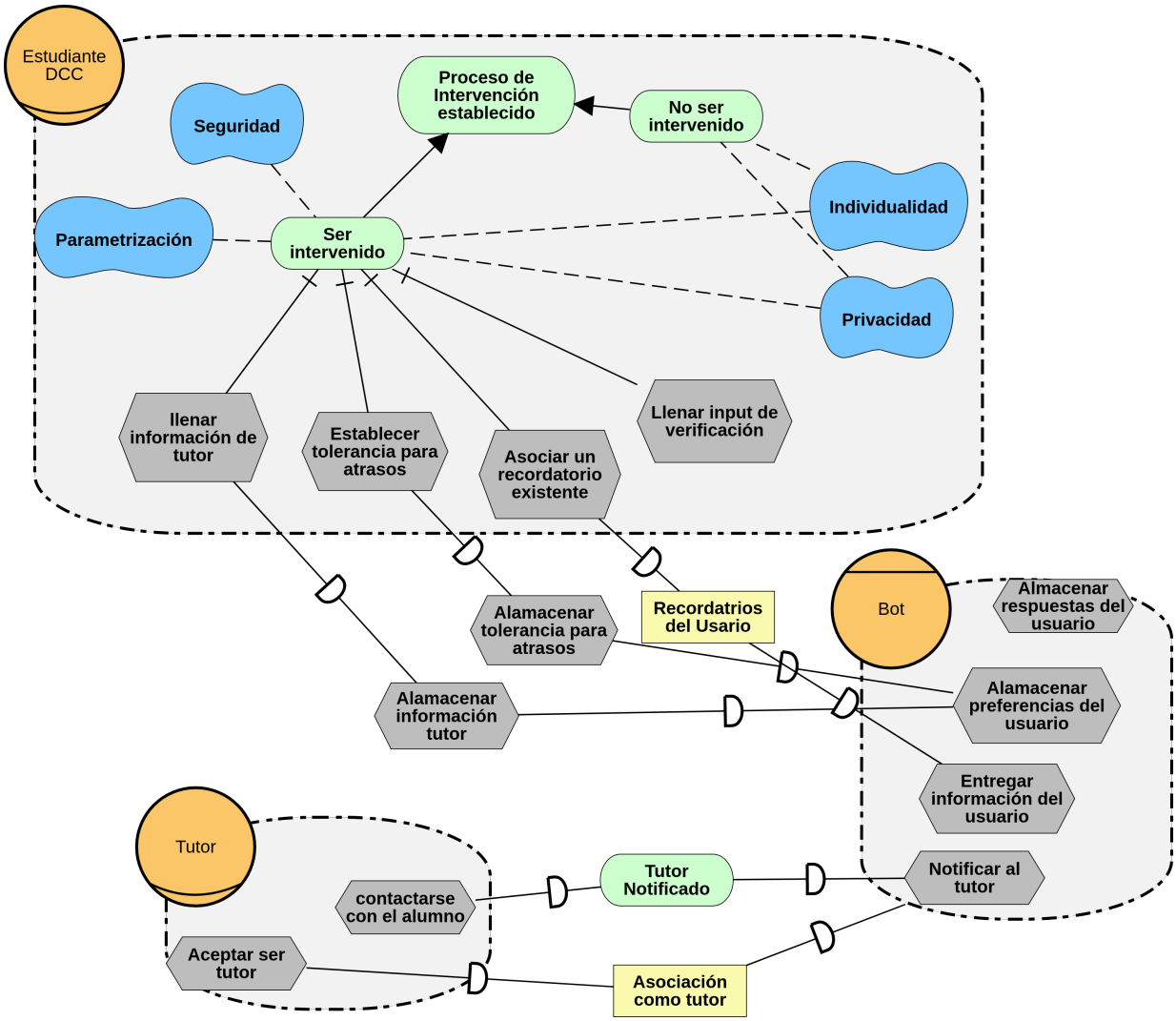

Figura 4.6: Diagrama i* que da cuenta de los objetivos y cualidades valoradas en una interacción que permita al alumno incluir a un usuario de apoyo durante el periodo de un proceso académico.

lidad. Siempre teniendo a la vista cuál es el objetivo final de cada una de las funcionalidades globales.

Algo, es que en un proceso de análisis normal, usualmente se plantean como objetivos «tareas »a lograr, en vez de estados a los que se quiere llegar, por lo tanto, es extraño definir un objetivo como por ejemplo: Tener el Contenido definido. Sin embargo, esto ayuda a visualizar que la tarea en sí, no es el objetivo final, sino un medio para un fin[6].

El objetivo no es la creación de un sistema que resuelva una tarea, sino que es el estado al que se quiere llegar resolviendo una tarea de software. Es la motivación de un sistema y no sus componentes.

Usualmente, este proceso es automático y/o heurístico en el ser humano, por lo tanto es

[6](Este es un problema que usualmente traspasamos al diseño de software)

fácil tratar funcionalidades del sistema, como objetivos del usuario. Eso en general hace que se pierda de vista, la manera en que deben ser logrados estos objetivos. Cómo expresa [26] «Un aparato heurístico garantiza una solución, pero probablemente el aparato óptimo para llegar a una solución no es una heurística».

Este tipo de análisis, permite distinguir entre los requisitos de usuarios (objetivos y cualidades) y los requisitos de software (tareas, recursos y cualidades) de manera global, muy útil en la dirección estratégica de proyectos. Mantener este tipo de nociones resulta fundamental en el desarrollo orientado a usuarios, porque es muy fácil que el quehacer del proyecto se puedan perder los horizontes originales y perder de vista el objetivo por el cual se desarrolla el sistema.

Por otra parte, hay que notar que estos diagramas representan un escenario ideal. En el que se extraen las necesidades, objetivos y restricciones de manera explícita en el planteamiento de una solución. Pero como todo desarrollo de software, se deben priorizar las funcionalidades de mayor importancia y también las que sean alcanzables durante la implementación. Respetando las reglas tiempo de todo desarrollo.

Desde del análisis del *Focus Group*, de los diagramas, y el traspaso del trabajo anterior, Se puede establecer a modo general, cuáles son los principales requerimientos. Estos debieran serán los que se alinean mejor con los objetivos de: «extender el sistema actual, agregando funcionalidades que permitan crear un sistema extensible, personalizado y confiable.»Considerando la complejidad de implementación se opta por: Las consultas y recordatorios.

Ambos agregan valor deseado por los usuarios, permiten personalizar su experiencia en cuanto a la información que desean conocer y son los que presentan la mayor aceptación en torno a privacidad y otras cualidades valoradas por los usuarios.

También se hace necesario analizar y modificar el sistema existente, para que sea capaz no solo de cumplir con la funcionalidad esperada, sino ser extensible de manera lógica y garantizar el desarrollo continuo de la plataforma.

4.4. Diseño de plataformas

A partir de los requerimientos recabados en el trabajo anterior, se diseñaron varios diagramas UML, que permitieron esquematizar las principales funcionalidades y cambios a realizar en el sistema. Eso sí, es fundamental decir que a estos diseños sufrieron modificaciones durante el transcurso del proyecto y algunos como el diagrama de objetos obtiene su versión final en el modelo de datos. La versión de los diseños que se presenta aquí es la obtenida a partir del análisis de los requerimientos y el planteamiento inicial.

4.4.1. Diseñar los casos de uso a partir de los objetivos de Usuario

Los casos de uso, son posibilidades que el sistema da a los usuarios. En nuestro diseño, representan funcionalidades de las que los estudiantes pueden hacer uso. Traspasando la

notación i*, los casos de uso vendrían a ser cada una de las maneras en que se cumple un objetivo, en otras palabras funcionalidad = Tarea a realizar.

Los casos de uso considerados son los siguientes.

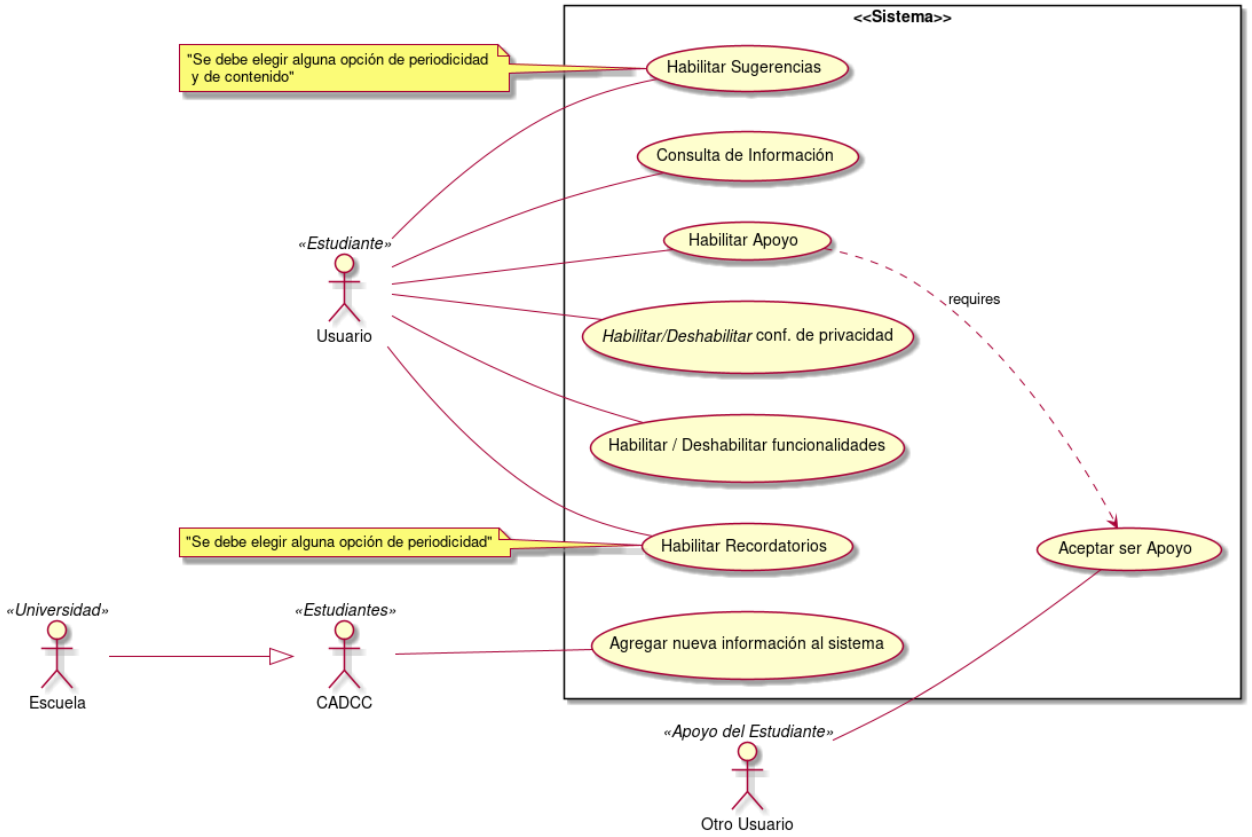

Figura 4.7: Casos de Uso extrapolados desde el análisis de requisitos.

Durante el planteamiento de los casos de uso, se agruparon varias de las tareas. Sobre todo porque desde los diagramas se extrae que: si bien el contenido puede variar, la funcionalidad deseada es la misma. En resumen del diagrama se puede extraer que el alumno podrá hacer 3 cosas principalmente en el sistema: Buscar información, suscribirse a información y habilitar permisos.

Cada una de estas funcionalidades clave, tiene muchos pasos y diseños asociados, pero mantienen una lógica similar, por eso se mostrará a continuación el diagrama de estados de una de ellas, a modo de ejemplo. Se eligió el diagrama de estado del proceso de susbcripción porque es una de las interacciones más abarcantes y representativas del sistema, y permite ver cómo interactua el usuario con la información y cómo responde el sistema.

En la figura anterior, se puede observar que el alumno debe seleccionar primero un proceso vigente y a partir de ahí, seleccionar alguno de los contenidos disponibles. Definido esto, se procede a elegir la funcionalidad deseada en torno al contenido y finalmente se solicita al alumno aceptar o no el almacenamiento de los datos, equivalente a permitir o rechazar la funcionalidad.

Una salvedad es que si los permisos hubieran sido aceptados en el pasado o configura-

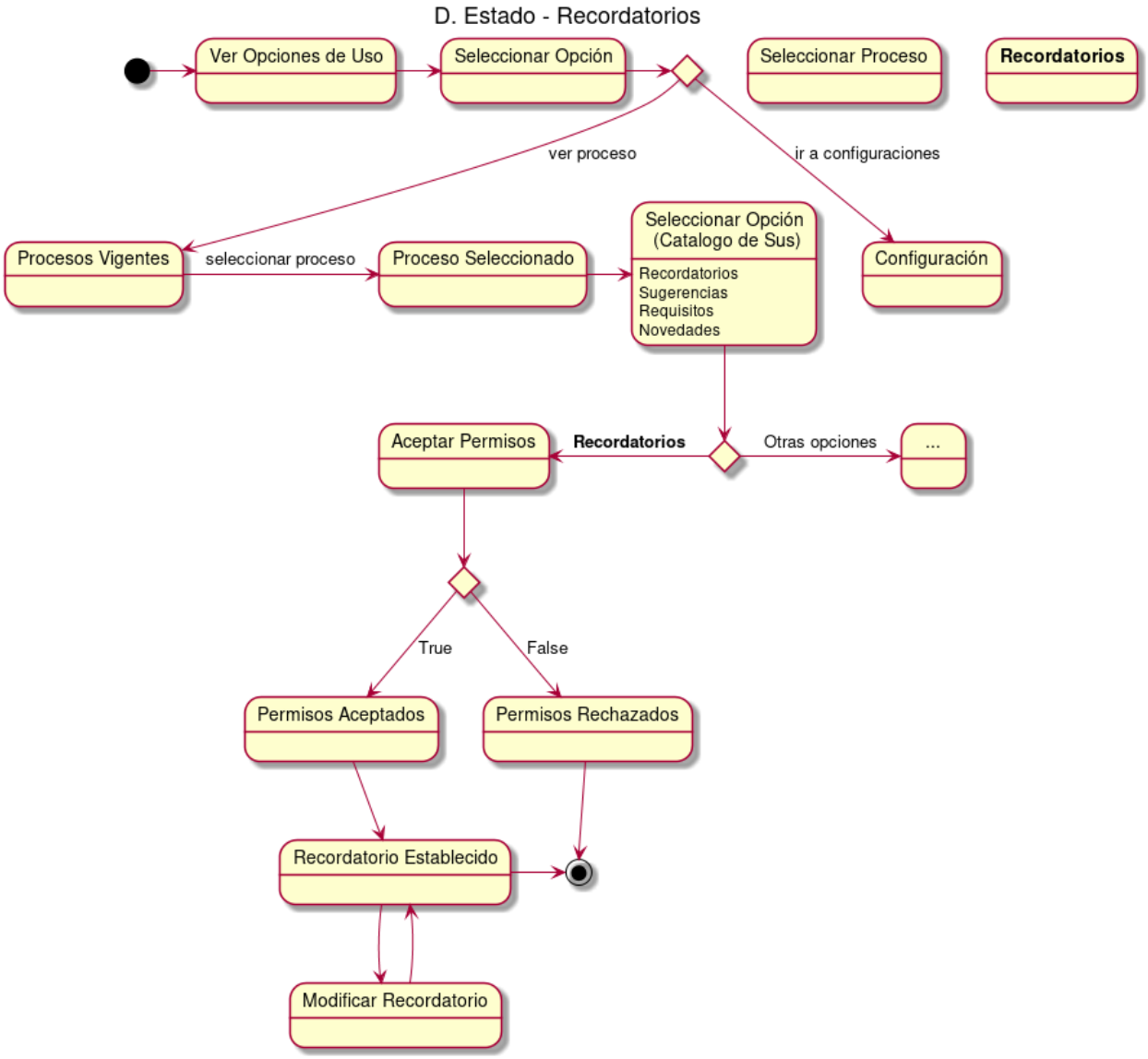

Figura 4.8: Diagrama de estado del proceso de suscripción.

dos globalmente, se puede pasar directamente a la funcionalidad esperada en este caso los recordatorios.

Una de las principales conclusiones de este diseño (que es que un diseño basado en una interfaz), es que necesariamente consta de varios pasos. Los cuales se pueden simplificar y trabajar, pero añade muchas permutaciones en torno a los posibles viajes del usuario, y también tiene un alto costo de diseño. A sí mismo, si además de establecer este viaje *default* agregamos la opción de que el usuario pueda parametrizar la frecuencia, se añaden muchas opciones de interfaz, lo que termina haciendo al sistema engorroso y lento de usar. Algo que no buscan los usuarios.

Es por esto, que luego del diseño de varias de las plataformas, se diseñaron dos formas paralelas de acceder a las funcionalidades, para mantener la simplicidad y la eficiencia, por un lado, mientras que la parametrización y control sobre otro.

4.4.2. Cambio a diseño dual de interfaces gráficas y comandos

Uno de los grandes hallazgos del proceso de diseño, es que las interfaces no se podían comportar de forma eficiente y parametrizable a la vez. Ya que se habrían tenido que añadir una gran cantidad de opciones de ajuste, esto al final redunda en un viaje de usuario muy largo, lo que claramente perjudica a los alumnos de tipo P.

A partir de esto, se propuso un nuevo sistema que funciona de manera paralela a las interfaces basadas en botones[7] y es una de comandos. Los comandos son una función bastante común en Telegram, ya que la mayoría de los bots ofrece un funcionamiento a partir de comandos, esto permite no solo identificar una opción a realizar, sino que agregar en el mismo comando texto y valores que pueden servir cómo parámetros para dichas funciones. Este tipo de interfaz, más parecido a una shell o terminal, es bastante familiar para los alumnos del dcc y además usuarios de Telegram, en ese mismo sentido proporcionan una enorme flexibilidad y extensibilidad, dado que a partir de un mismo identificador se pueden efectuar muchos ajustes y configuraciones.

4.4.3. Nuevo Modelo de datos

A partir de los casos de usos se creó el siguiente diagrama de objetos que da cuenta de las principales relaciones entre los objetos del sistema. Uno de los grandes cambios al modelo de original es la creación de un Usuario (o perfil) que debe servir como el centro para las suscripciones, los permisos y el usuario de apoyo. Así mismo se identifica que deben agregarse tablas de contenido para las instancias. Finalmente, una de las grandes modificaciones es que se necesita una forma de mantener un catálogo de las cosas a las que los usuarios pueden acceder, particularmente suscribirse, las que no pueden quedar simplemente en el código, ya que por naturaleza son mutables de instancia a instancia, eso implicaría agregar muchas líneas de código o quitar muchas para cada instancia de cada proceso en particular.

El problema principal con este catálogo, es que se debe diseñar alguna forma de mantenerlo actualizado, por ahora el diseño es manual, pero en un futuro podría ser de forma automática.

- **User:** Información de acceso al sistema de administración.
- **Proceso:** Contiene los procesos que existan. Ejs: Proceso de Titulación, Prácticas, etc.
- **Categorías:** Categorías de las preguntas frecuentes.
- **FAQ:** Preguntas Frecuentes.
- **Instancias:** Instancias de un proceso, por ejemplo: Proceso de Titulación 2022.
- **Novedades:** Son las noticias, pero relacionadas con una instancia en particular del proceso.

[7] A pesar de que no se diseño una GUI propiamente tal, los *keyboards* de Telegram, proveen una especie de interfaz gráfica que se puede generar a partir de los intercambios de mensajes entre el usuario y el BOT.

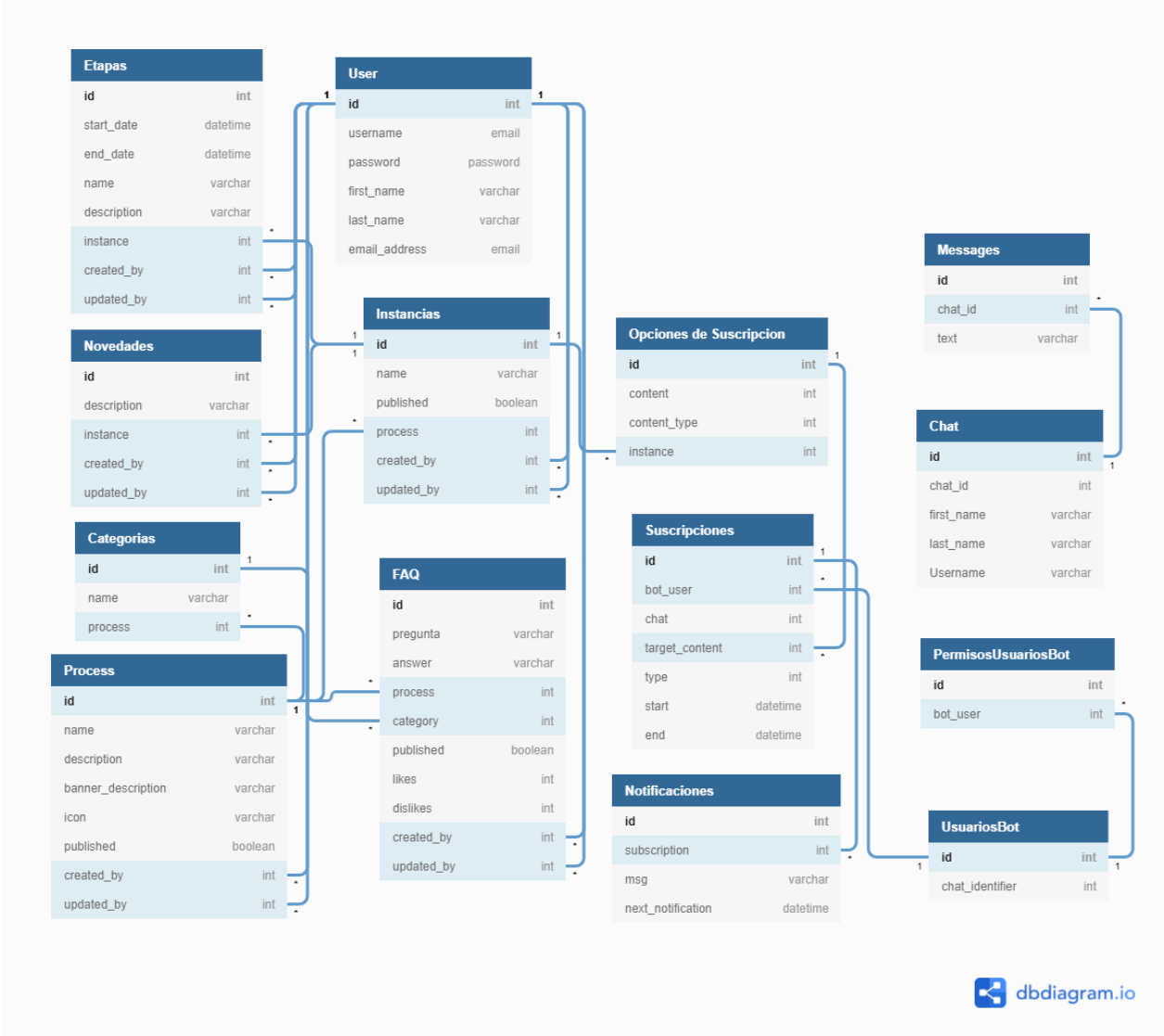

Figura 4.9: Nuevo Modelo de datos del sistema

- **Etapas:** Fechas importantes relacionadas con una instancia particular del proceso.
- **Usuarios del Bot:** identificador del chat.
- **Permisos de los Usuarios del Bot:** Configuraciones de funcionalidad y seguridad de los usuarios.
- **Suscripciones:** Suscripción creada por un usuario para algún contenido disponible.
- **Opciones de suscripción:** opciones de suscripción habilitadas para los usuarios.
- **Notificaciones:** Notificaciones generadas por las suscripciones de los usuarios.
- **Chat:** Campos de un chat de telegram para conectarlos con el administrador
- **Mensajes:** Mensaje de un chat

Capítulo 5

Implementación

A partir del análisis y diseño, se realizó la implementación del sistema, los pasos expuestos en la metodología son Reestructuración, Implementación de la nueva Arquitectura, Implementación de nuevas funcionalidades, y un análisis de los problemas

5.1. Reestructuración

El código existente para el bot contenía varios problemas de diseño que hacían compleja su extensibilidad y se pueden revisar detalladamente en la sección de trabajo previo.

Para solucionar estos problemas primero se extrajeron las funcionalidades clave del bot: Obtener información desde la base de datos, recibir request, procesar request y enviar mensajes. La extracción de información, la recepción y envío de mensajes en sí no presentaba mayores problemas de diseño.

5.1.1. Cambio en el procesamiento de las Requests

Cambio de procesamiento condicional a parser

Cómo se vio en la sección de trabajo previo, la API de telegram permite varias formas de comunicación. De ellas se utilizan los websockets. A partir de aquí, los bots pueden recibir todos los mensajes enviados por el usuario, sin embargo, telegram permite agrupar los mensajes en más o menos 4 modalidades: primero texto, archivos o media adjunta, comandos y los keyboards.

En el diseño anterior, como se explicó, se seleccionaron principalmente una manera de comunicación a través de inline keyboards. (sección anterior: En este tipo de comunicación se presentan al usuario una serie de botones que el puede apretar y a partir de los se envia un mensaje al servidor, cuando este es procesado se envía la respuesta al usuario de manera asincrona). Para poder desarrollar las nuevas features, se añadieron comandos, en miras

```
def message_processing(self, t_chat, message,
    label=None, question=None ):
    messages = []
    if label is None:
        if message == '/start':
        messages = [ {"text": 'Hola ... ',
        "keyboard": {}}, ... ]
        elif message == '/preguntasFrecuentes': ...
        elif message == '/asistente': ...
    Else: ...
        elif label == 'Process': ...
        elif label == 'Category': ...
        elif label == 'Question': ...
        elif label == 'Helper':  ...

    for msg in messages:
        self.send_message(msg['text'],
        t_chat["id"], msg['keyboard'])
```

Listing 2: Sistema de procesamiento basado en ifs sobre el tipo de comunicación y la expresión de texto

tambien de procesar texto. Se amplió el diseño para pasar de una decisión condicional a un parsing con double dispatch.

```
def post(self, request: AsgiRequest, *args, **kwargs):
    update = request.body
    parser.decode_update(update)
    return JsonResponse({"ok": "POST request processed"})

    # Parser
    def decode_update(self, update: bytes):
            # Se detecta que el update es un mensaje
            if update.match(message-regex) :
            return self.parse_expression(message)
            # Se detecta que el update es una call_backK_query
            elif update.match(call_backK_query-regex):
            self.parse_expression(label, chat_id)
            # No see detecta que es el update
            else:
                return Updates.UpdateProccesingResult()
```

Cómo se puede apreciar en este nuevo sistema, el parser se hace cargo de determinar que tipo de update o actualización fue recibida desde telegram y a partir de esa detección se pasa la expresión recibida a un handler, este determina que acción debe realizarse y luedo llama a la instancia determinada de la acción para que se procese la request. Este nuevo modelo es util principalmente por 5 razones. Mantiene la integridad del procesamiento sin importar si el tipo mensaje de telegram es conocido o no. Separa el parsing de la estructura de la api de Telegram del procesamiento de la acción requerida por el usuario en el sistema. Dualidad Parser-Handler.

Permite escalar de forma ordenada el código para recibir muchas nuevas funcionalidades siguiendo un diseño estándar de procesamiento. Esto porque separa la identificación de la acción de su código en particular, estandariza el llamado de las acciones y permite reutilizar patrones de código sin duplicar código. Utiliza un parsing basado en regex y no en comparaciones de strings, lo que permite una mayor flexibilidad y robustez del procesamiento.

Da la posibilidad de añadir nuevos tipos de procesamiento de acciones sin cambiar la lógica de traducción de la API del bot. En otras palabras, para añadir procesamiento de lenguaje natural, no hay que hacer grandes modificaciones al parser, solo hay que añadir el handler correspondiente.

Creación del Handler

En el sistema anterior todos los comandos tenían su propio bloque de código, principalmente porque solo enviaban mensajes de texto de vuelta al sistema. En el nuevo sistema se deja la tarea de determinar que acción tomar a un Handler. Hay dos grandes diferencias entre el parser y handler (porque ambos funcionan como un parser).

La primera es que el Handler se hace cargo de la lógica específica de cada acción de manera estándar, es decir una vez que identifica la expresión de texto, llama a un objeto que contiene la acción a realizar. Uno de los mayores beneficios es que se puede ver claramente cuál es árbol de decisión o el camino de la acción sin entrar en la acción particular, a la vez que permite que las acciones se puedan modificar o extender sin cambiar el llamado predeterminado de los objetos. También hace sencillo extender la funcionalidad.

La segunda, es que cada tipo de mensaje en la API tiene su propia forma de procesarse dada la naturaleza de la comunicación en la API de Telegram (objetos de JSON). Por dicha razón cada modo de comunicación tiene su propio Handler, esto separa lógicamente la comunicación mediante comandos del texto plano o las callback queries [1] . Esto implica que para extender las maneras de comunicarse basta con agregar un handler que represente esa manera de comunicación. Actualmente se procesan comandos y callback queries por lo tanto solo existen esos handlers.

La estructura de un Handler es simple, es un objeto que contiene instancias de las diferentes acciones a realizar y define a si mismo, una función handle, que hace el match entre una expresión y la expresión regular que identifica a la misma. Si el match se cumple se llama a la función `do_action` de la Acción en Cuestión.

A continuación se muestra como ejemplo el comand Handler

```
1 class CommandHandler(Handler):
2     start_command = StartCommand()
3     help_command = HelpCommand()
4     unknown_command = UnknownCommand()
```

[1]Las formas de comunicación responden a diferentes maneras en que un usuario puede enviar mensajes en telegra, telegram diferencia estas maneras entregando objetos JSON con propiedades distintas, hay varias pero en la memoria se usan 3: texto plano, callback queries y comandos, que se procesan de forma similar al texto plano

```
    ...

    def handle(self, expression: str, for_id: int) -> None:

        # expression <-> /start
        if self.start_command.re.match(expression):
            self.start_command.do_action(chat_id=for_id)

        # expression <-> /help
        elif self.help_command.re.match(expression):
            self.help_command.do_action(chat_id=for_id)
```

Creación de un modelo OO para respuestas

Un tercer aspecto es que se extrapolo el concepto de acción a una clase abstracta Action, utilizando el modulo abc de python. El único compromiso de esta clase es realizar una acción. Y a esta clase la extiende dos clases una es Command y la otra es CallBackQuery. Que agrupan las funcionalidades tipicas de cada forma de comunicación, esto permite al handler instanciar varios Commands o CallBackQueries por ejemplo y llamar `do_action` en cada una para representar la acción buscada por el usuario. Esto también permite diseñar la respuesta a cada acción de manera individual ocultando la complejidad del modelo de interacción.

La última modificación fue, que se creó una clase llamada Objects con el objeto de imitar el comportamiento de los objetos de la API basados en JSON a objetos de python. El gran problema lo presentan los objetos compuestos sobre todo en la distinción de tipos, python ha ido agregando varias herramientas de tipado, pero el estándar de JSON carece de ellas y complica modelos de herencia muy extendidos. Por ejemplo un mensaje contiene dentro un keyboard y el keyboard a su vez contiene una callback query a realizarse y ser procesada de vuelta por el sistema. Las detecciones a través de match y regex son muy útles a la hora de procesar el mensaje, pero no tienen el mismo valor a la hora de construirlos, es por eso, que se diseñó un modelo de objetos que pudiera interactuar de manera más sencilla con los objetos de la API de Telegram. Sin embargo, todavía no se logra un modelo completamente funcional que además sea lo suficientemente robusto, como para usarse. Para la creación de estos objetos, se usaron Typed Dictionaries (TypedDict) una nueva libreria de python 3.10.

Estos objetos buscan 3 simplificar 3 procesos: la detección de objetos compuestos y sus funcionalidades correspondientes a cada objeto. La transformación de JSON a python de objetos compuestos. Y por ultimo de python a JSON.

Se propone investigar si usar el modelo de serializers de API rest u otros similares. El principal problema no es la traducción de uno a otro lenguaje, sino la traducción y detección de objetos compuestos.

Como resumen, se pasó de un modelo que contenía toda la lógica de procesamiento de mensajes en la view a uno que separa en capas lógicas cada parte de la interacción, aprovecha modelos de herencia y estandariza el comportamiento esperado en cada fase del procesamiento de información.

Imagen vs modelo anterior actual

Se migró de un modelo no extensible (según la lógica de diseño modular) a uno extensible. Se ordenaron en capas la resolución de mensajes y se modularizó y estandarizó la manera de tomar la acción debida en cada situación.

5.2. Nuevas Funcionalidades

5.2.1. Adopción de un framework para taras asíncronas

Algunas de las funcionalidades comprometidas requerían la ejecución de tareas de forma asíncrona. Particularmente las funcionalidades de subscripción, tales como los recordatorios. Ya que se debía levantar una notificación en el momento indicado para cada usuario. Con este fin, se agregó la librería "celery" para el manejo de tareas en python.

Celery, es un gestor de tareas o cola de tareas y trabaja en base a un broker, eso significa que puede recibir tareas desde diferentes instancias o servicios, y procesaralas de forma eficiente.

La gran ventaja es que para realizar esto se configura una aplicación de celery, esta "entidad" se encarga de enlazar las configuraciones hechas en el código por el desarrollador a un worker. Un worker es una instancia de celery que funciona como un thread o proceso. Entonces celery envía a través del broker las tareas que esten configuradas en una APP a un proceso que las ejecuta. En celery se pueden ejecutar tareas directamente, de forma asíncrona o programadas.

A continuación se explica como se usan tareas para lograr hacer funcionar los recordatorios.

- **Subscripciones:** Una suscripción cómo se explicó en el proceso de diseño es la capacidad que le da el sistema al usuario de elegir de entre una colección de opciones, una o más a las que suscribirse par obtener información relevante en una cierta frecuencia. Para poder entregar estos mensajes en el momento especificado se crean notificaciones y se requiere un proceso de actualización.
- **Notificaciones:** Una Notificación, en este sistema es un mensaje que debe ser enviado a un usuario en algún momento

5.2.2. Sistema de Suscripciones

Las susbscripciones son un sistema de 3 pasos: primero se crea una suscripción, más una notificación asociada y luego el sistema revisa periódicamente las notificicaciones almacenadas para decidir cuáles corresponden a ser enviadas. Finalmente modifica la notificación para que almacene el próximo envío. Para revisar periódicamente las notificaciones, se establecieron

tareas recurrentes con celery que revisan la base de datos manera constante y en el caso de encontrar una colección de notificaciones que deben enviarse hoy, se envían. De este modelo el envío de las notificaciones se realiza por parte del gestor de tareas. Esto da un comportamiento dual en el que las suscripciones se realizan de manera síncrona (por el usuario) y las notificaciones se envía de forma programada (por el sistema).

La configuración de celery es simple, ya que se realiza una breve configuración inicial y las tareas simplemente se identifican con @decorators[2] de Python El código queda más o menos así:

```python
app = Celery('API')
app.config_from_object('django.conf:settings', namespace='CELERY')
app.autodiscover_tasks()

app.conf.beat_schedule = {
    # Executes every Monday morning at 7:30 a.m.
    'add-every-minute': {
        'task': 'tasks.send_today_notifications',
        'schedule': crontab(minute=1)
    },}

@shared_task
def send_today_notifications():
    notifications = Notification.get_today_notifications()
    messages = [] #[{"text": '', "keyboard": {}}]

    for notification in notifications:
        msg = markup_clearner(f'{notification.msg}')
        try:
            notf_susc = notification.subscription
            notf_chat = notf_susc.chat
            print(notf_chat)
        except:
            notf_chat = None
        notf_message = {'text': msg, 'keyboard': {}, 'chat': notf_chat}
        messages.append(notf_message)

    for msg in messages:
        if msg['chat'] is None:
            continue
        async_send(msg['text'], msg['chat'], msg['keyboard'])
```

Algunas notas sobre el sistema de Subscripciones. Para lograr programar tareas se utiliza un módulo de celery que se ejecuta como un proceso paralelo llamado celery beat. Además los resultados de las tareas y las tareas programadas se guardan en la base de datos. De esta forma hay al menos 2 procesos paralelos en ejecución un worker de la aplicación y beat el programador que hace uso de la db.

[2]Los @decorators de python, son azúcar sintáctico, en el que nombre del wraper se corresponde con una función del mismo nombre, esta última recibe cómo parámetro la función que el @decorator acompaña.

5.3. Cambios en Diseño

5.3.1. Cambios en la arquitectura

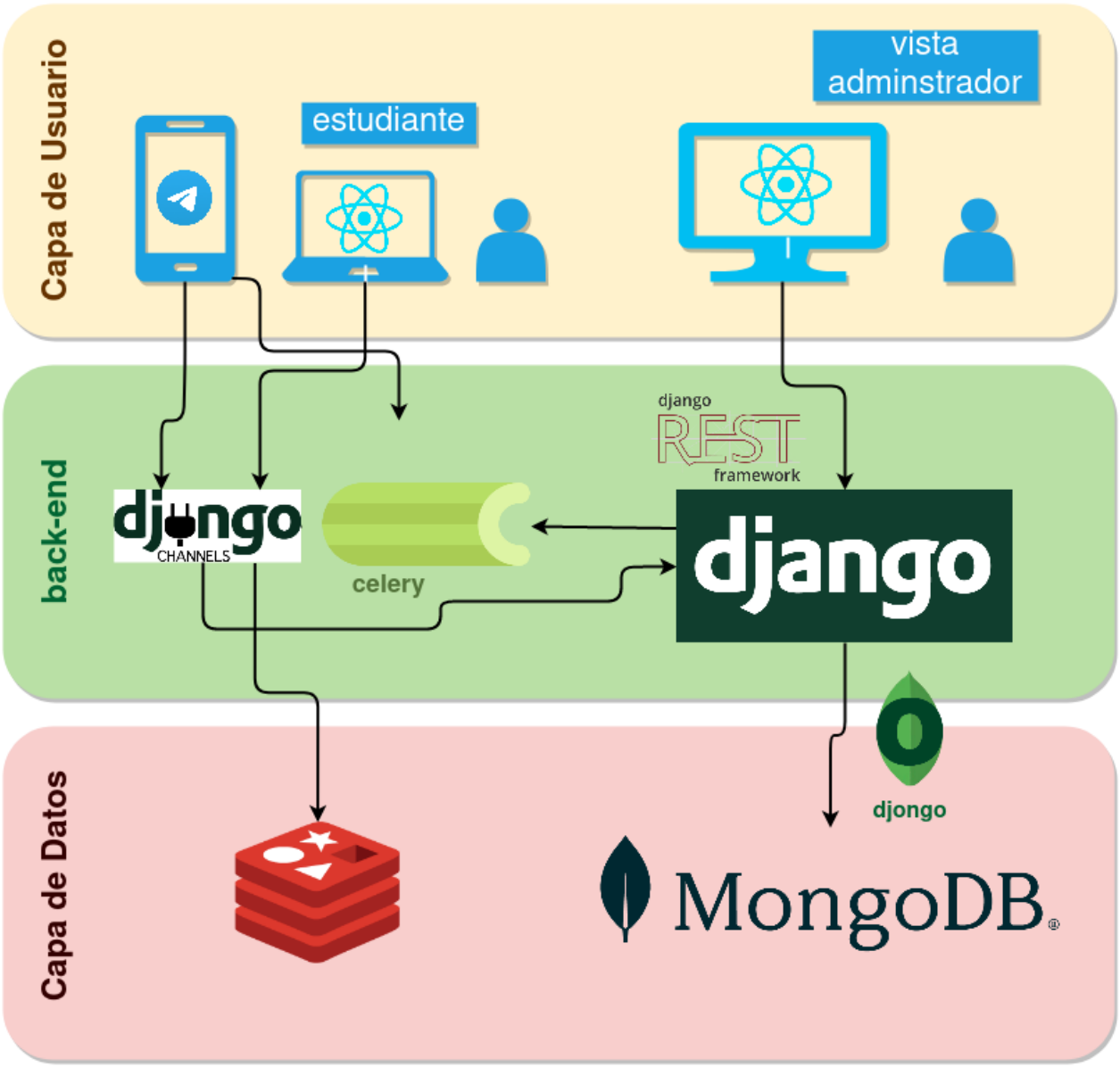

Figura 5.1: Diagrama de arquitectura del nuevo sistema

La nueva arquitectura cómo se puede apreciar en el diagrama integra:

- **La Base de Datos (Mongodb):** En la base de datos se almacenan los datos de autenticación, la información de la aplicación, el contenido de subscripción, las preferencias de los usuarios y las tareas.
- **Backend Django:** Contiene la lógica del servicio de contenido, también configura y programa las tareas (de manera manual una vez). Se comunica a través del broker con la cola de tareas.

- **Broker de Mensajes (Redis):** Sincroniza a través de mensajes diferentes módulos de la aplicación
- **API Telegram:** Servicio externo proporcionado por telegram que permite la comunicación entre la app y los usuarios del bot (que son por defecto usuarios de telegram).
- **El Front web:** Sirve tanto como plataforma de información como centro de control de para los staf de la mesa de ayuda.
- **Celery:** Maneja las tareas programadas, asíncronas y directas desde la aplicación.

5.3.2. Cambios en los módulos

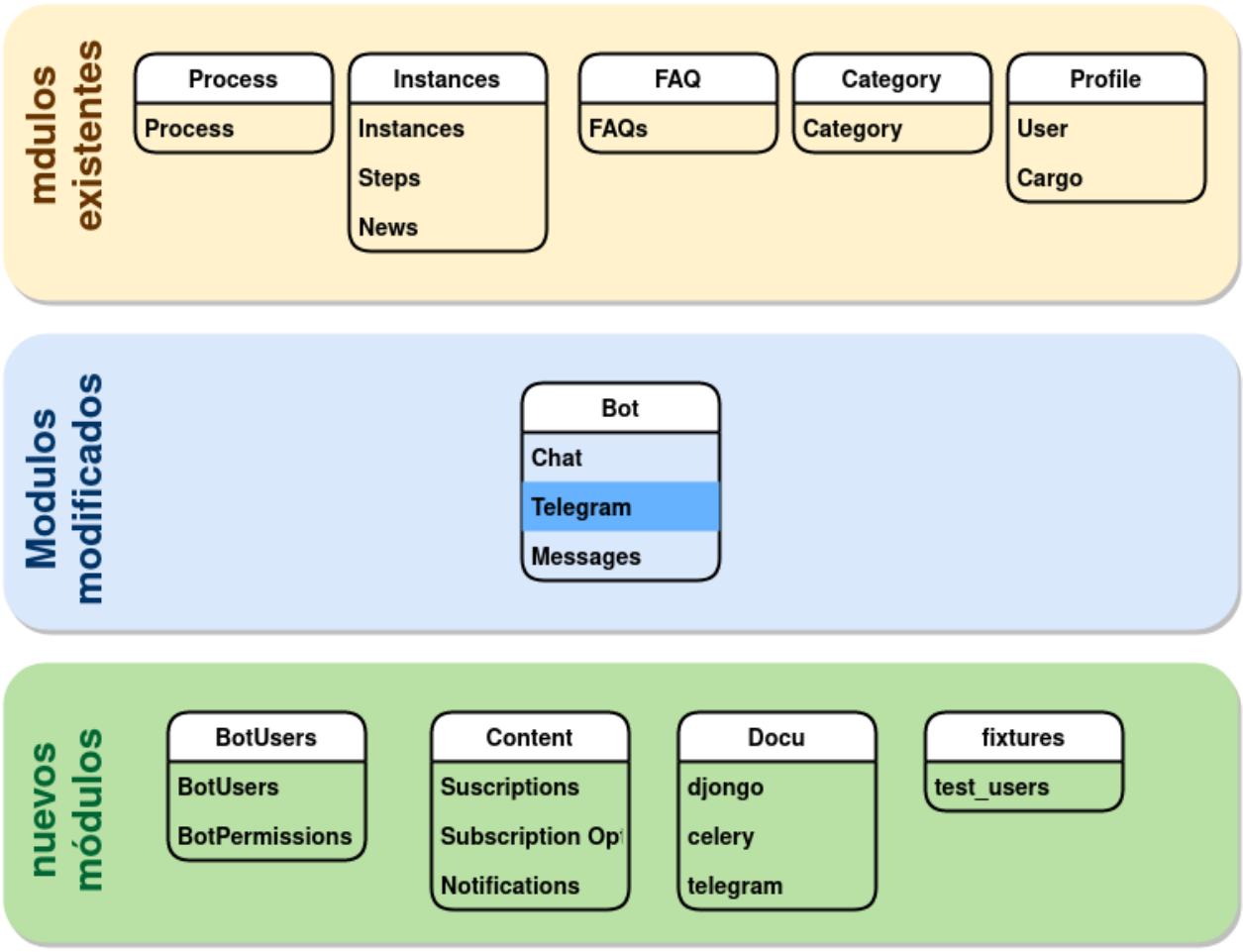

Figura 5.2: Diagrama de arquitectura del nuevo sistema

Módulos ya existentes

- Process, correspondiente al módulo de procesos descritos anteriormente.
- Instances, correspondiente a las instancias de cada proceso.

- FAQ, el cual aborda las preguntas frecuentes asociadas a un proceso.
- Category, asociado a las categorías de una pregunta. Esta aplicación es utilizada por el bot y se describe con más detalle en la sección 5.3.
- Profile, correspondiente al manejo de usuarios(as) de la aplicación Web.

Módulo modificado

- Bot, correspondiente a la implementación del bot de Telegram de esta solución. Se añaden los submodulos de: Telegram y task. En tasks, se almacena las nuevas tareas periodicas del sistema, aunque pueden agregarse en cualquier parte del backend. En el modulo de Telegram se integra la nueva lógica de parser, handler, acciones y objetos.

Modulos nuevos

- BotUsers, corresponde al manejo de los usuarios(as) del Bot.
- Content, correspondiente al manejo de las suscripciones y otras funcionalidades de contenido.
- Docu, que añade la documentación del sistema y los procesos realizados
- Fixtures, que añade datos de prueba para poblar el sistema.

5.4. Problemas de compatibilidad

Dentro del desarrollo del sistema, dadas las tecnologías heredadas, y decisiones de diseño anteriores, al implementar nuevas tecnologías hubo varios problemas de compatibilidad, los más relevantes se detallan a continuación.

Django que es el framework de desarrollo web en el que está desarrollado el proyecto, funciona con un sistema llamado Object Relationship Mapping, que lo que hace es traducir operaciones de objetos de django a queries en la base de datos relacional. Esto en general era parte de desarrollo de las aplicaciones en frameworks como sping de java, aun es necesario realizar este tipo de adecuaciones, para trabajar con los resultados salientes de la base de datos. Esta funcionalidad clave de django da una enorme simplicidad y versatilidad al código. Por otro lado Mongo es una base de datos no relacional, si bien se puede integrar a python con varias librerías integrarla con el ORM de django no es tan sencillo. De hecho, en general hay conectores a base de datos pero no algo similar al ORM. En ese sentido djongo es un proyecto que logra soslayar esta dificultad, con lo que ellos llaman un ODM un Object Document Mapping. Ahora como todas las operaciones en Mongo son operaciones de documentos y no de tablas, la traducción no siempre es perfecta, y eso implica que cosas como una migración para cambiar el tipo de una tabla por ejemplo no solo no sean necesarias sino imposibles. Ya que mongo no necesita ni tiene un método para cambiar el tipo de tabla, en realidad, nada asegura que un documento dentro de la misa colección tenga el mismo formato que otro, por

esto se puede modificar los modelos de django en vivo y el cambio se ve reflejado de manera directa sin migraciones.

Por otro lado para lograr la mayor compatibilidad posible djongo crea un súper set al módulo models de django para incluir algunas estructuras interesante como modelos embebidos, arreglos entre otros.

Uno de los grandes problemas es que django por default crea un identificador de tipo int, que se puede reemplazar manualmente por cualquier otro, pero en el caso de djongo se agrega usualmente por temas de compatibilidad, sin embargo, esto no ocurre el 100 % de las veces, de hecho, para todos los modelos nuevos que se crearon en el proyecto para algunos funcionaba y para otros no aleatoriamente, esto causo un error muy complejo de resolver porque fue muy difícil de identificar. Al mismo tiempo complicó la administración de módulos o librerías que funcionaban con el ORM de django como la vista de Admin y Celery.

Para resolverlo se probaron muchas soluciones, pero finalmente, lo que se hizo fue extender la clase Models por default de djongo-django, para que incluyera automáticamente el identificador id que ahora es de tipo uuid. Esto solucionó, los problemas de compatibilidad con el administrador y otros servicios similares.

Para celery y otros servicios, la solución fue no pasar por django, celery se conectó directamente a Mongo como base de datos a través de una librería experimental, pero la forma de guardar los elementos la hace imposible de manejar con el ORM de django o con las vistas de administrador. Por lo tanto para poder administrar y revisar las tareas se requirió instalar flower. Esta librería permite hacer el seguimiento a los eventos que celery expone (lo que se debe configurar) y eso permite hacer un seguimiento de las tareas de la sesión, para las tareas programadas en otro arranque de flower y celery, se hace necesario revisar directamente en la base de datos, ya que tampoco tienen permanencia los mensajes en el Message Broker en este caso Redis.

La suma de todos estos problemas hizo que fuera difícil configurar el proyecto, hacer trazabilidad de los errores, y una vez configurador entender por qué ciertas líneas de código funcionaban para ciertos objetos en la base de datos y para otros no.

Sumado a esto Celery y djongo son librerías bastantante activas, de hecho Celery cambió completamente su documentación de dominio hace solo unos meses, las últimas releases son del 2021, aunque ha habido convenciones anuales por Varios años. Por otro lado djongo se actualizó de ser una librería libre, a tener una versión de pago para empresas, a dar soporte, incluso ahora se ofrecen otros servicios cloud usando mongo Atlas entre otros.

Esto hace que mantener el proyecto requiera de estar en constante revisión de los recursos utilizados, ya que las dependencias están en constante actualización. Esta fue una de las lecciones más importantes del proyecto, a la vez que uno de los mayores desafíos.

5.5. Resumen

Inicialmente se propusieron el análisis y reestructuración de código existente para soportar las nuevas funcionalidades. Además dentro de las nuevas funcionalidades se propuso creer un sistema que permitiera ser personalizado y confiable, a partir del análisis del *Focus Group*, se añadieron variables sobre la privacidad, parametrización y eficiencia.

A partir de diseño y de las evaluaciones se priorizaron las funcionalidades más complejas en términos de implementación, menos controversiales y con más apoyo de los usuarios. Esto derivó en que no se realizaran cambios para agregar información a las respuestas y mejorar el modelo de feedback del usuario. Esto sumado a las *features* que requieren análisis de privacidad usualmente requieren especificar más de una vía de acción.

A modo general se cumplió con lo planificando priorizando los cambios de mayor envergadura. Se corrigieron bugs y se mejoró el estilo de procesamiento del bot, para permitir dejar un sistema extensible.

Capítulo 6

Validaciones

6.1. Diseño escalable del Bot

La nueva arquitectura (ver figura 5.1),es un sistema de 3 capas: La capa de usuario, el back-end, la capa de tareas y la capa de datos, mediadas por un *broker* de mensajes. En su conjunto ellas crean un sistema capaz de servir información, de recibir y procesar tareas de manera síncrona y asíncrona.

La nueva forma de procesar información y de segmentar las acciones del sistema para responder individualmente a cada necesidad del usuario, sumado a la capacidad añadida de separar la forma de comunicación del sistema, permite añadir: Nuevas formas de procesar información como lenguaje Natural, extender las vías de comunicación existentes como botones y comandos. Al mismo tiempo permite extender las capacidades de procesamiento de información añadiendo la oportunidad de planificar tareas en el sistema, lo que serviría para realizar mantenciones de datos, obtener estadísticas, hacer sistemas de aprendizaje automático, entre otros.

Todo el trabajo de *refactoring* apuntaba a hacer el sistema extensible, a ordenarlo de forma lógica, y a permitir diferenciar entre las distintas capas de procesamiento del sistema, permitiendo su continuidad cómo proyecto para el alumnado del dcc.

A sí mismo la modularización creada para manejar la API de Telegram, permite ir añadiendo nuevas funcionalidades en torno a las vías de interacción que facilita la plataforma.

6.2. Diseño estratégico para proyectos

Este proyecto busca la adopción por parte de la comunidad del DCC, por tanto todo su diseño y capacidades iniciales y nuevas, apuntan a crear un sistema que resuelva las reales necesidades de la comunidad. Sin embargo, su mantención y mejora, dependen de alumnos que no necesariamente estarán en contacto con los desarrolladores anteriores del proyecto. Para esto es imprescindible que el proyecto por sí mismo pueda ser entendido, reparado y extendido

por personas que pueden o no conocer las tecnologías del sistema. Las que no conocerán las motivaciones iniciales del proyecto, que no necesariamente están al tato del marco teórico que envuelve al sistema, así cómo de las dependencias que este tiene de librerías y trabajos de terceros.

Para que un proyecto pueda continuar vigente, es necesario que su código sea extensible y mantenible. Pero otra de las contribuciones del trabajo actual, es proveer un marco de diseño y evaluación de las funcionalidades en torno a las preferencias del usuario, la mayoría de las interacciones que el sistema puede proveer al usuario se pueden esquematizar con la sintaxis i*, y los diagramas actuales generalizan las interacciones posibles. Sin embargo, aunque no fueran adoptados los diagramas, estos proveen de conceptos claves a la hora de evaluar la forma de interactuar de los estudiantes con el sistema. También, cómo estos afectan la manera de diseñar la solución y, la necesidad de mantener formas alternativas de cumplir los objetivos de cada alumno. Y por sobre todo apuntan a mantener de forma explícita en el diseño los objetivos de los usuarios.

Estas contribuciones están en línea con los hallazgos actuales y con la literatura existente. Permiten por tanto, tomar decisiones estratégicas en torno a este proyecto para la comunidad. Esto es de ayuda para quienes continúen el proyecto, y tiene la intención de servir de apoyo para las generaciones futuras.

En el área de la extensibilidad, se hicieron logros importantes en la documentación del código, que permiten a su vez de forma práctica, levantar el proyecto, configurar los módulos, agregar datos de prueba y usar el proyecto, sin necesidad de conocer todas las tecnologías y plataformas utilizadas.

Todo esto en su conjunto permite que nuevas funcionalidades sean implementadas. Manteniendo un ecosistema favorable tanto para el desarrollador como para la comunidad, y desarrollar piezas de software que no solo cumplan con realizar una acción, sino que se amolden a las preferencias de la comunidad, y de cada usuario particular, proveyendo de valor, para la diversidad de estudiantes del dcc.

Todo esto crea un marco conceptual, que permite hacer evaluaciones estratégicas del proyecto, provee una metodología de diseño y promueve que el sistema pueda ser extendido y mejorado de forma efectiva por los estudiantes que continúen el trabajo.

6.3. Valor agregado para Usuarios

A continuación se muestran solo los flujos vía comandos porque los flujos en botones ya existían en el trabajo anterior.

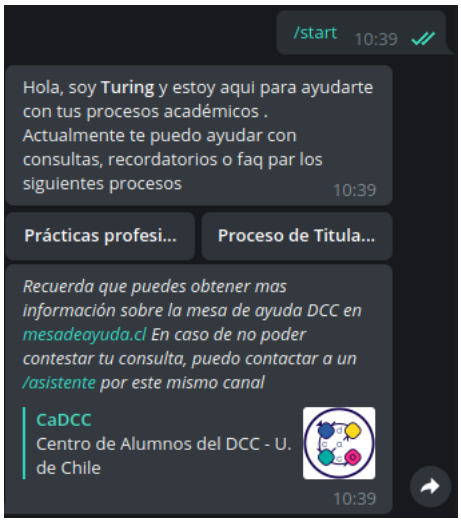

Figura 6.1: Captura de pantalla del Bot Respondiendo el nuevo comando start

En el nuevo sistema de procesamiento el bot procesa los comandos en 3 fases. Primero entiende que el usuario ingresa un comando de tipo `r'\/command'`. Luego se pasa al handler, el handler determina si es comando aceptado y envía un mensaje como el escrito arriba.

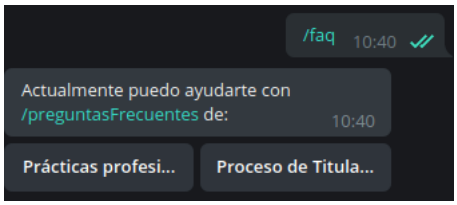

Figura 6.2: Captura de pantalla del Bot Respondiendo el nuevo comando faq

Complementando los flujos de botones se añadieron versiones de comando de las mismas acciones (ver figura 6.2). Además se pueden agregar otras palabras. Pero ambas se procesan de forma completamente diferente (ver sección 5.1)

Además las configuraciones de seguridad y preferencias permiten a los usuarios estar al control de las *features* que requieran, y pueden ser activadas o desactivadas en cualquier momento (ver figura 6.3)

Este tipo de configuraciones y funcionalidades, permiten al alumno escoger de que ser informado, segmentar y priorizar la información, y así, se añade valor a este sistema de información.

Se añadió el módulo *BotUsers* a la API, permitiendo que cada alumno pueda escoger la información que es relevante para él. Además ahora puede ser notificado por ejemplo de los plazos de cada proceso. Este módulo contempla que se puedan añadir otras funciones de

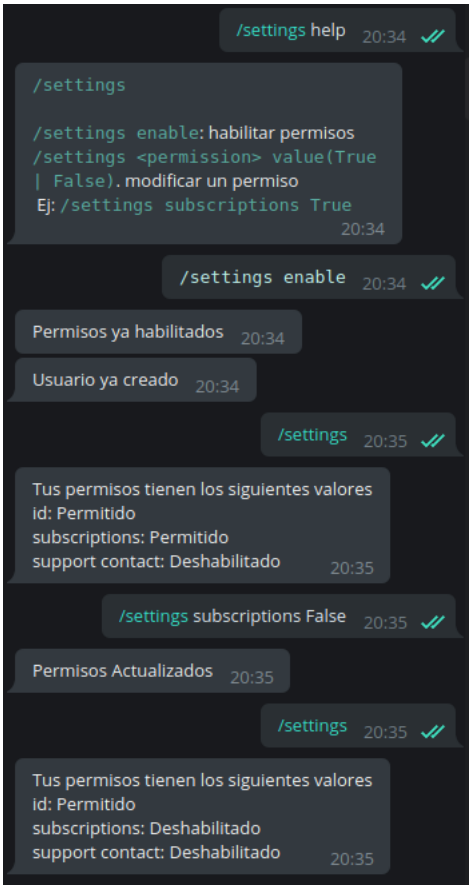

Figura 6.3: Captura de pantalla del Bot Respondiendo el nuevo comando settings

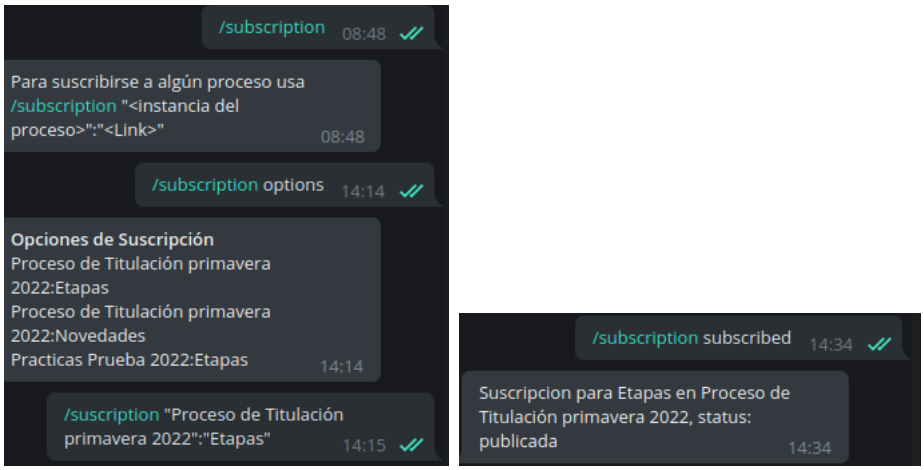

Figura 6.4: Captura de pantalla del Bot Respondiendo el nuevo comando subscription

subscripción como las sugerencias en el futuro. Esto permitiría seguir añadiendo un valor personalizado para cada alumno (ver figura 6.4).

6.4. Resumen

De lo expuesto en este capítulo, se puede ver que se satisfacen los requisitos analizado así los cambios necesarios para extender y modificar el sistema, se agregó documentación técnica y lógica al proyecto, se rediseñaron las funcionalidades existentes, y se agregaron nuevas funcionalidades que permiten hacer un sistema extensible, personalizado y confiable.

Además, durante el transcurso del proyecto se sostuvieron conversaciones con las directivas del Centro de Alumnos del Departamento de Ciencias de la Computación. Y se les presentaron los cambios en el bot, así cómo los problemas de compatibilidad entre otras cosas. El centro de alumnos en conjunto con el memorista anterior, fueron validando los cambios en el código, las nuevas funcionalidades logradas y también apoyaron en la resolución de algunos errores específicos.

Además, cómo se explicita en la memoria, el diseño de las funcionalidades se hizo tomando de forma explícita los objetivos de los alumnos y las cualidades que valoran en estos objetivos.

Capítulo 7

Conclusiones

7.1. Discusión Final

Con esta memoria se busca seguir mejorando el sistema de mesa de ayuda, y de este modo mejorar la experiencia de los alumnos durante los procesos académicos, particularmente el proceso de titulación. El objetivo general era «extender el sistema actual, agregando funcionalidades que permitan crear un sistema extensible, personalizado y confiable. El cual será actualizado principalmente por el CADCC, con miras de integrar otros actores posteriormente, favoreciendo la continuidad de este servicio». Para esto se dividió el objetivo principal en 6 objetivos específicos:

En concordancia con el objetivo 1, se analizó la solución existente y se determinaron que las modificaciones necesarias eran: Cambio de procesamiento condicional de la *request*, la separación en capas de procesamiento de mensajes, así cómo la estandarización y modularización de las respuestas del sistema a través de la creación del *Parser*, el *Handler* y los objetos asociados a las acciones.

Se reestructuró el código existente, lo que permitirá la adición de nuevas funcionalidades en el futuro, así cómo extender y mejorar las ya presentes.

Se rediseñaron los modelos actuales, así mismo como las funcionalidades, para que el sistema fuera capaz de recibir las mejoras presentadas en concordancia con el objetivo 2.

Cumpliendo el objetivo 4, se desarrolló un modelo de suscripción personalizada: El que permite para cada proceso agregar opciones de subscripción a las que el usuario puede suscribirse, así mismo se creó un modelo de notificaciones, y data mínima del alumno para habilitar o deshabilitar estas funcionalidades. Se creó la opción de generar recordatorios a partir de la suscripción a las etapas de una cierta instancia de proceso académico

A partir del diseño, la moficiación e implementación de nuevas funcionalidades, y resolviendo los problemas de compatibilidad de las librerías, se asegura la integridad de los modelos de datos, del código y se permite la extensibilidad a través de la modularidad lógica, cumpliendo el objetivo 4

Además se incluyeron explícitamente en el diseño las preferencias de los alumnos, y buscando funcionalidades relevantes para ellos, a través del diseño de los diagramas i* y su posterior conversión a UML, y funcionalidades del sistema. Por esta misma razón se creo un sistema dual que permite tanto la interacción a través de botones cómo de comandos.

Sin embargo, los problemas de compatibilidad, sumado a lo extenso de las reestructuraciones, no permitieron agregar más funcionalidades de suscripción para los otros contenidos valorados por los alumnos y tampoco añadir componentes de confiabilidad en la información. Aunque se creó la segmentación suficiente como para que sea sencillo añadir estas funciones en el futuro.

Se añadieron además al proyecto, documentación necesaria para levantar el proyecto, tener nociones clave de los usuarios y sus preferencias, establecer un marco conceptual que permita darle una dirección estratégica al proyecto en el futuro. Permitiendo la mantención, la mejora y el desarrollo continuo por alumnos del dcc.

En resumen se extendió el sistema actual agregando los módulos de *parser*, *handler*, y *actions*, se modificaron y crearon modelos de datos para proveer funcionalidades de suscripción, se corrigieron errores de compatibilidad en la base de datos, se agregó un módulo para el procesamiento de tareas asíncronas. Se añadieron opciones de configuración, además de recordatorios programables. Además de documentación y nociones clave en desarrollo y mantenibilidad del proyecto, permitiendo crear un sistema, extensible, personalizado y confiable.

A partir del trabajo se lograron varios aprendizajes. Primero, extender el trabajo de otra persona, si este esta mal documentado es inviable el trabajo práctico dónde los tiempos son acotados y se esperan resultados en plazos más o menos fijos. Por eso se hacen necesarias buenas prácticas de programación para extender los sistemas existentes, sobre todo cuando las personas que van a ir rotando constantemente.

Las soluciones que dependen de librerías o diseños de terceros son muy vulnerables a quedar deprecadas en el corto o mediano plazo. Sobre todo si son tecnologías de uso extensivo. Esto hace que los encargados de un proyecto necesariamente deban conocer las dependencias del sistema y su rol en el mismo, y mantenerse al tanto de los cambios que estas implican. Por otro lado, mantenerse aislado de las actualizaciones que las librerías externas puedan tener, hace que el código se degrade aceleradamente, porque se va quedando sin recursos para interactuar con nuevas tecnologías. Esto implica que una empresa se puede quedar atrás muy rápido si no hace un adecuado balance entre mantenerse al día con las tecnologías usadas y reestructurar el código para ir asegurando su integridad cada cierto tiempo.

La verdad es que un balance complejo, que la muchas de las instituciones no sabe llevar, porque si algo funciona, no se cambia no se toca, el problema es que en el mundo del software, algo rara vez permanece estático por mucho tiempo, a menos que haya sido abandonado. Son las constantes y crecientes mejoras las que van haciendo a los sistemas más robustos. En ese sentido la conclusión más importante que se obtiene de este trabajo, en el área de la mantención de software, es que se necesita adquirir un balance entre las herramientas que se usan y la mantención que estas requieren, y muchas veces este balance depende de un equipo tan calificado como en el área de innovación. Esta es una de las razones por las que muchas empresas prefieren externalizar todo aquello que no es de forma directa parte de su negocio,

porque se vuelve muy difícil y costoso mantener las soluciones existentes.

Se hace muy útil, conocer herramientas diversas a la hora de desarrollar una solución, porque esto puede reducir los tiempos de desarrollo considerablemente. Aunque siempre que se incluya una solución externa, es muy necesario evaluar el impacto de aprendizaje, mantención y capacidad de desarrollo que serán necesarias para mantener la solución vigente.

Otra de las conclusiones que se saca del trabajo realizado, es que el software basado en usuarios es complejo, y requiere una gran cantidad de diseño, además de una gran cantidad de datos. Realizar una tarea de la forma en que los usuarios esperan es muy costoso para la mayoría de las empresas pequeñas. Sin embargo, también se da cuenta, que si se hace un análisis lo suficiente claro respecto a los objetivos y valores del usuario es posible obtener buenos resultados de diseño e implementación, lo que puede ser una alternativa para soluciones de menor escala.

Finalmente, para que un trabajo tenga continuidad es necesario que el proyecto de software desarrollado tenga su propia lógica, que pueda ser continuada por personas que no necesariamente están en contacto con los creadores. Así mismo, se ve la importancia del trabajo en equipo para desarrollar soluciones complejas y completas, no solo para realizarlas más rápido, sino porque el resultado es más duradero en el tiempo. Es por eso, que a juicio del alumno, el mundo opensource es todavía mayoría en lo que se refiere a software, porque para que las soluciones permanezcan y evolucionen en el tiempo deben evolucionar las comunidades junto con ellas.

7.2. Pasos a seguir

Queda pendiente una validación con los usuarios de las nuevas funcionalidades. Aunque se añaden conceptos clave para traer al diseño del sistema sus preferencias y objetivos. Así como un marco de decisión para la evaluación de nuevas funcionalidades del sistema.

A partir de la memoria anterior, aún queda pendiente la puesta en marcha del sistema. Y su expansión a otros actores del departamento. Este punto es clave, y se espera que a través de los acuerdos llegados con el CADCC, se puedan pasar a producción este año.

El *refactoring* del bot y su separación en capas de procesamiento a través del *parser*, permite añadir nuevas formas de comunicación y procesamiento de información como el procesamiento en lenguaje natural propuesto en el trabajo anterior [2]

la adición del módulo de tareas permite la adición de otros modelos de aprendizaje automático no supervisado. Uno de los propuestos en esta memoria es el desarrollar un sistema de notificaciones que pueda tomar inputs del usuario, inputs sociales anónimos y cambios en la información para determinar el mejor momento para interactuar con el alumno. Y se pueda adecuar al flujo de trabajo individual de cada usuario.

También tanto la interfaz de Telegram como la plataforma web, abren la posibilidad de investigaciones sobre de *UX* y UI.

por otro lado, la puesta en marcha del sistema es el paso clave, y se sugiere que cualquier nuevo proyecto, tenga como objetivo inicial asegurar la puesta en marcha del sistema.

Así mismo se recomienda extender la documentación y crear un equipo de soporte para la plataforma, que permita su funcionamiento independiente de los trabajos de título que puedan mejorarlo. Actualmente el CADCC es un aliado clave en este proceso, pero se recomienda que más actores pudieran verse involucrados en la mejora del sistema.

Apéndice

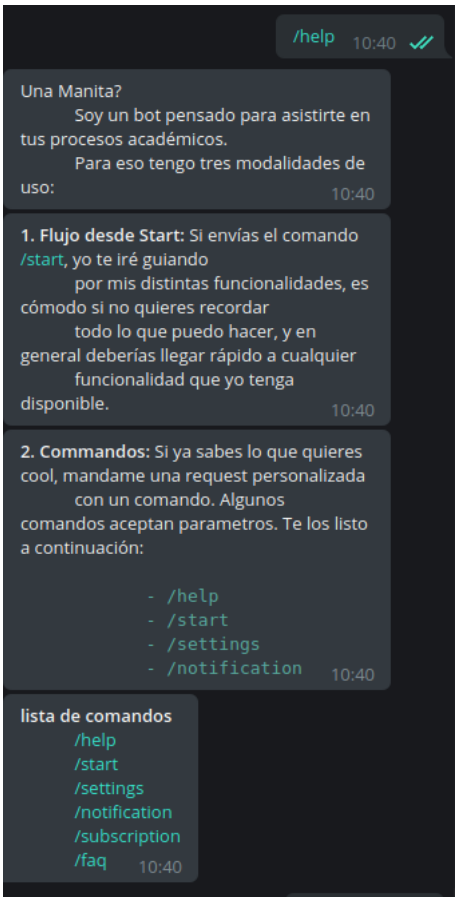

Figura 7.1: Captura de pantalla del Bot Respondiendo el nuevo comando help

Bibliografía

[1] Sherif Abdelhamid y Andrew Katz. «Using Chatbots as Smart Teaching Assistants for First-Year Engineering Students». En: *ASEE American Society for Engineering Education* (2020), pág. 7. URL: https://peer.asee.org/using-chatbots-as-smart-teaching-assistants-for-first-year-engineering-students.pdf.

[2] PABLO IGNACIO ARANCIBIA BARAHONA ARANCIBIA. *Creación de un sistema interactivo de información para mejorar el proceso de titulación en el DCC*. Inf. téc. Santiago de Chile: Universidad de Chile, 2021.

[3] CADCC. *Elección directiva CaDCC 2021*. 2021.

[4] CADCC. *Padrón 2016-2 Elecciones CaDCC 2017.xlsx*. Santiago, 2016. URL: https://www.u-cursos.cl/uchile/2008/0/COMCADCC/1/material_docente/detalle?id=1619365.

[5] CaDCC. *Padrón Oficial Votaciones CaDCC 2019*. 2018. URL: https://www.cadcc.cl/tricel-padron-oficial-votaciones-cadcc-2019/ (visitado 03-05-2021).

[6] Rectoría de la Universidad de Chile. *Reglamento de Presupuesto - Universidad de Chile*. 2014. URL: https://www.uchile.cl/portal/presentacion/senado-universitario/reglamentos/reglamentos-aprobados-o-modificados-por-el-senado-universitario/104521/reglamento-de-presupuesto (visitado 16-05-2021).

[7] Chan Chun Ho y col. «Developing a Chatbot for College Student Programme Advisement». En: *2018 International Symposium on Educational Technology (ISET)*. IEEE, jul. de 2018, págs. 52-56. ISBN: 978-1-5386-7209-9. DOI: 10.1109/ISET.2018.00021. URL: https://ieeexplore.ieee.org/document/8456189/.

[8] Fabio Clarizia y col. «Chatbot: An Education Support System for Student». En: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 11161 LNCS. Springer Verlag, oct. de 2018, págs. 291-302. ISBN: 9783030016883. DOI: 10.1007/978-3-030-01689-0_23. URL: https://link.springer.com/chapter/10.1007/978-3-030-01689-0_23%20http://link.springer.com/10.1007/978-3-030-01689-0_23.

[9] Fabiano Dalpiaz, Xavier Franch y Jennifer Horkoff. «iStar 2.0 Language Guide». En: (2016), págs. 1-15. arXiv: 1605.07767. URL: http://arxiv.org/abs/1605.07767.

[10] David Fox, Jonathan Sillito y Frank Maurer. «Agile methods and user-centered design: How these two methodologies are being successfully integrated in industry». En: *Proceedings - Agile 2008 Conference* (2008), págs. 63-72. DOI: 10.1109/Agile.2008.78.

[11] Elizabeth Hansen y Emily Goligoski. *Tow Center for Digital Journalism A Tow/Knight Report GUIDE TO AUDIENCE REVENUE AND ENGAGEMENT*. Inf. téc. 2018. DOI: 10.7916/D8BG410W. URL: https://academiccommons.columbia.edu/doi/10.7916/D8BG410W%20https://doi.org/10.7916/D8BG410W.

[12] M W Hasyim, S Pramono y Sutrisno. «Web-Based Telegram Chatbot Management System: Create Chatbot Without Programming Language Requirements». En: *IOP Conference Series: Materials Science and Engineering* 1096.1 (mar. de 2021), pág. 12075. DOI: 10.1088/1757-899x/1096/1/012075. URL: https://doi.org/10.1088/1757-899x/1096/1/012075.

[13] A. Heryandi. «Developing Chatbot for Academic Record Monitoring in Higher Education Institution». En: *IOP Conference Series: Materials Science and Engineering* 879.1 (ene. de 2020), pág. 12049. ISSN: 1757899X. DOI: 10.1088/1757-899X/879/1/012049. URL: https://iopscience.iop.org/article/10.1088/1757-899X/879/1/012049%20https://iopscience.iop.org/article/10.1088/1757-899X/879/1/012049/meta.

[14] Vaishnavi Ajay Inamdar y Shivanand R.D. «IRJET-DEVELOPMENT OF COLLEGE ENQUIRY CHATBOT USING SNATCHBOT DEVELOPMENT OF COLLEGE ENQUIRY CHATBOT USING SNATCHBOT». En: *International Research Journal of Engineering and Technology* 6.7 (jul. de 2019), págs. 1615-1618. ISSN: 2395-0072. URL: www.irjet.net%20https://www.irjet.net/archives/V6/i7/IRJET-V6I7525.pdf.

[15] Bii P K, J K Too y C W Mukwa. «Teacher Attitude towards Use of Chatbots in Routine Teaching». En: *Universal Journal of Educational Research* 6.7 (2018), págs. 1586-1597. DOI: 10.13189/ujer.2018.060719. URL: http://www.hrpub.org.

[16] John Karat. «Evolving the Scope of User-Centered Desing». En: *COMMUNICATIONS OF THE ACM* 40.7 (1997).

[17] Joseph Kramer, Sunil Noronha y John Vergo. «A User-Centered Design Approach To Personalization». En: *Communications of the ACM* 43.8 (2000), págs. 44-48. ISSN: 15577317. DOI: 10.1145/345124.345139.

[18] Guoliang Li. «Human-in-the-loop data integration». En: *Proceedings of the VLDB Endowment*. Vol. 10. 12. Association for Computing Machinery, ago. de 2017, págs. 2006-2017. DOI: 10.14778/3137765.3137833. URL: https://dl.acm.org/doi/abs/10.14778/3137765.3137833.

[19] Stephen Marsh, Stephen Marsh y Mark R. Dibben. «The Role of Trust in Information Science and Technology». En: *Annual Review of Information Science and Technology (ARIST)* 37 (2003), págs. 465-98. ISSN: 0066-4200.

[20] Neil McBride. *Human in the loop*. Mar. de 2021. DOI: 10.1177/0268396220946055. URL: https://journals.sagepub.com/doi/abs/10.1177/0268396220946055?journalCode=jina.

[21] No-mention. *Summary of i* Notation*. 2011. URL: http://istarwiki.org/tiki-index.php?page=Summary+of+i%2A+Notation&structure=i%2A+Guide (visitado 02-08-2021).

[22] María Paz y col. «Liderazgo transformacional: su impacto en la confianza organizacional, work engagement y desempeño laboral en trabajadores millennials en Chile». En: *Revista de Psicología* 30.1 (mar. de 2021), págs. 1-17. ISSN: 0719-0581. DOI: 10.5354/0719-0581.2021.55066. URL: www.revistapsicologia.uchile.cl.

[23] Andrii O Priadko, Kateryna P Osadcha y Bogdan Khmelnytsky. *Development of a chatbot for informing students of the schedule*. Inf. téc. Melitopol: Bogdan Khmelnytsky Melitopol State Pedagogical University, feb. de 2019, págs. 129-137. URL: http://ceur-ws.org/Vol-2546/paper08.pdf%20http://elibrary.kdpu.edu.ua/handle/123456789/3744.

[24] DrGeetha M Professor y col. «CHATBOT FOR EDUCATIONAL INSTITUTION». En: *Journal of Xi'an University of Architecture & Technology* 8.5 (ene. de 2020), págs. 2919-2913. ISSN: 1006-7930. URL: http://xajzkjdx.cn/gallery/306-may2020aszw.pdf.

[25] Bhavika R. Ranoliya, Nidhi Raghuwanshi y Sanjay Singh. «Chatbot for university related FAQs». En: *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*. Vol. 2017-Janua. IEEE, sep. de 2017, págs. 1525-1530. ISBN: 978-1-5090-6367-3. DOI: 10.1109/ICACCI.2017.8126057. URL: http://ieeexplore.ieee.org/document/8126057/.

[26] Marc H.J. Romanycia y Francis Jeffry Pelletier. «What is a heuristic?» En: *Computational Intelligence* 1 (1 ene. de 1985), págs. 47-58. ISSN: 14678640. DOI: 10.1111/J.1467-8640.1985.TB00058.X. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-8640.1985.tb00058.x.

[27] Ms Akshata Sarang y Mr Shital Agrawal. «STUDENT INFORMATION CHATBOT SYSTEM FOR COLLEGES». En: *International Research Journal of Modernization in Engineering Technology and Science @International Research Journal of Modernization in Engineering* 2.6 (jun. de 2020), págs. 2582-5208. ISSN: 2582-5208. URL: www.irjmets.com%20https://irjmets.com/rootaccess/forms/uploads/student-information-chatbot-system-for-colleges.pdf.

[28] Alison Smith y col. «Closing the loop: User-centered design and evaluation of a human-in-the-loop topic modeling system». En: *International Conference on Intelligent User Interfaces, Proceedings IUI*. Association for Computing Machinery, mar. de 2018, págs. 293-304. ISBN: 9781450349451. DOI: 10.1145/3172944.3172965. URL: https://doi.org/10.1145/3172944.3172965.

[29] David A Thurman, Jeffrey S Tracy y Christine M Mitchell. *Design of an Intelligent Web-Based Help Desk System*. Inf. téc. URL: http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=337433F8ECB944A5FD6CC08B0191DE20?doi=10.1.1.34.1442&rep=rep1&type=pdf.

[30] Universidad de Chile. *Vicerrectoría de Tecnologías de la Información (VTI) - Universidad de Chile*. URL: https://www.uchile.cl/vti (visitado 17-05-2021).

[31] Dieter Wallach y Sebastian C. Scholz. «User-Centered Design: Why and How to Put Users First in Software Development». En: *Maedche A., Botzenhardt A., Neer L. (eds) Software for People. Management for Professionals. Springer, Berlin, Heidelberg.* (2012), pág. 294. DOI: 10.1007/978-3-642-31371-4.

Apéndice

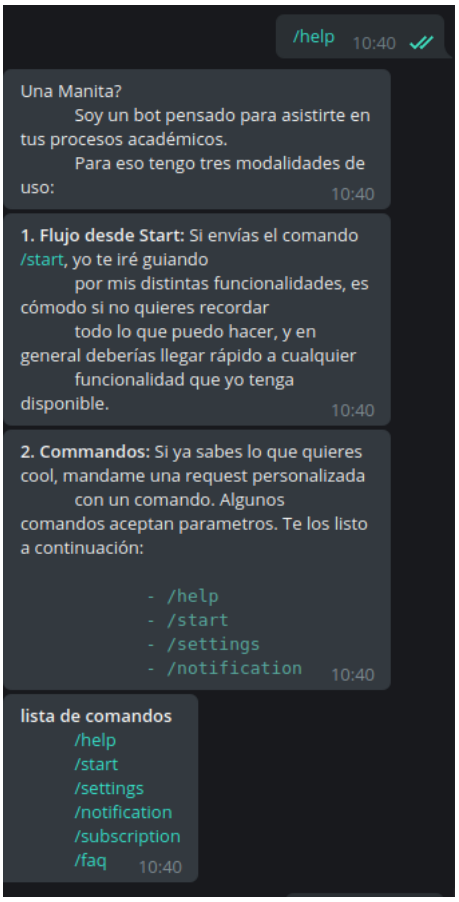

Figura 7.2: Captura de pantalla del Bot Respondiendo el nuevo comando help

Glosario

Django Telegram es un software gratiuto, multiplataforma, basado en la nube, de mensajería instantanea. El servicio además provee video encriptado de principio a fin, VoIP, envío de archivos y varias otras funciones. Fue lanzado para iOS en Agosto 14 2013 y Android en Octubre 2013 . 16, 18

Efective GUI Cuando se habla de *Efective GUI* o interfaz gráfica efectiva para el usuario, se habla de una interfaz que está diseñada para simular o reemplazar una interacción humana, se acuña principalmente en el área de IA relacionada con los chatbots. 9, 10

i* El lenguaje de modelamiento i*, fue introducido como un marco de referencia orientado a modelar actores y objetivos. Consiste en un lenguaje de modelamiento junto con una serie de tecnicas que permiten analizar estos modelos. [9]. i, v, 10–12, 24, 34, 50, 55

REST Representational State Transfer, architectural style for distributed hypermedia systems. 16

Telegram Telegram es un software gratiuto, multiplataforma, basado en la nube, de mensajería instantanea. El servicio además provee video encriptado de principio a fin, VoIP, envío de archivos y varias otras funciones. Fue lanzado para iOS en Agosto 14 2013 y Android en Octubre 2013 . 13, 36, 49, 56

Siglas

Alumno P alumno práctico. 22, 30, 31

Alumno R alumno regular. 22, 30

Alumno S alumno auto suficiente. 22, 30, 31

CADCC Centro de Alumnos del Departamento de Ciencias de la Computación. 5, 13, 17, 20, 53, 54, 56, 57

CC5402 CC5402 Proyecto de software. 20

CC6908 CC6908 Introducción al trabajo de título. 21, 24

DCC Departamento de Ciencias de la Computación, de la Universidad de Chile. i, 13, 20, 49

FG *Focus Group*. 6, 17, 20, 22, 23, 27, 33, 48

UCD User Centered Design. 7, 10, 19