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

# DESARROLLO Y PERFECCIONAMIENTO DE SISTEMA DE COEVALUACIÓN DE EQUIPOS PARA CURSOS DEL DCC

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERA CIVIL EN COMPUTACIÓN

CAROLINA ESCILDA CONTRERAS RODRÍGUEZ

PROFESORA GUÍA:
JOCELYN SIMMONDS WAGEMANN

MIEMBROS DE LA COMISIÓN:
CLAUDIO GUTIÉRREZ GALLARDO
ADOLFO CARRASCO ACOSTA

SANTIAGO DE CHILE
2021

# RESUMEN

Como parte del desarrollo personal de un individuo, la percepción de uno mismo y de otros resulta fundamental. El poder evaluar a otro con plena objetividad, dejando de lado relaciones interpersonales y objetivos propios puede llegar a ser una de las cosas más difíciles de hacer, pero aún más difícil que eso, puede ser el evaluarse a sí mismo, pero el lograr hacerlo conlleva beneficios tanto personales como colectivos, pudiendo reconocer fortalezas y debilidades y potenciar de acuerdo a eso. Estas capacidades dentro de un grupo ayudan a enfrentarse a desafíos, ya que se sabrá de antemano en qué se puede aportar con mayor capacidad y en cuáles tendrá que pedir ayuda o entregar más tiempo.

En función de esto, es que los cursos de ingeniería de software del Departamento de Ciencias de la Computación de la Universidad de Chile incorporaron coevaluaciones a sus programas, mas su forma de hacerlo manualmente podía producir vacíos de información entre evaluaciones, dejándose olvidados los resultados al arribar una nueva evaluación, sin la opción de analizar un progreso o cambio en comportamientos. Además, la logística de hacer una coevaluación manualmente puede ser altamente complicada, ya que entre recabar las evaluaciones y los cálculos de puntajes los errores humanos pueden ocurrir y perjudicar a alguien.

Como solución a esto es que en el Departamento de Ciencias de la Computación existe el Sistema de Coevaluaciones, un sistema que permite realizar las coevaluaciones en línea, quedando un registro de las evaluaciones anteriores contestadas por los alumnos. Sin embargo, este sistema está en uso desde su primera versión en 2014 y su construcción ha hecho desde difícil a imposible su mantenimiento.

Para solucionarlo, este documento propone y desarrolla la creación de un nuevo sistema de coevaluaciones, robusto, útil, fácil de usar y mantener, creado bajo la misma arquitectura que el sistema original y actualizando sus funciones y vistas. Con él, se hace posible la gestión de cursos, alumnos, profesores, coevaluaciones y equipos, la implementación de herramientas de seguimiento de desempeño y el poder contestar tanto auto como coevaluaciones. Implementado para los usuarios Administrador, Alumno y Profesor, posee además una solución temporal para Ayudantes, de tal forma que el sistema pueda ser usado en un contexto de curso real.

Este nuevo sistema fue evaluado por distintos actores: profesores, administradores y alumnos, además de ser usado en el contexto real de una coevaluación de un curso de 42 personas. Los resultados de estas evaluaciones obtuvieron promedios sobre el 80 % en encuestas de usabilidad y comentarios generales positivos, de los cuales se pudo a la vez obtener *feedback* para mejorar el sistema.

La implementación de este sistema permitió obtener una alternativa usable para el sistema de coevaluaciones original, con las funciones ya existentes y otras nuevas que aprovechan el sistema y su potencial como herramienta de evaluación y análisis.

*Dedicado a todos quienes me inspiraron y ayudaron a llegar a cumplir mis sueños.*

# Agradecimientos

Agradezco a mis padres, Oscar y Patricia, por apoyarme en mi camino y darme todos los recursos para poder sobrellevarlo. También, y aunque no lo puedan leer, le agradezco a mis perros, Chimuelo, Pillina y Peluca, por su amor incondicional y las noches que se desvelaron conmigo.

A Renata, Rocío, Camila y Andrea, por las risas y buenos momentos brindados desde mi primer día en la carrera, recorrer este viaje con ustedes ha sido maravilloso.

A Erenata, por ser mi amiga y consejera de la vida adulta y espiritual, por estar siempre ahí cuando la necesito, apañando como ninguna otra, brindándome su apoyo e infinitas risas.

A Carolina Mondaca por ser mi partner de nombre, colegio, universidad, carrera y amiga incondicional.

A Gonzalo, por estar para apoyarme siempre, haciéndome reír en cualquier momento.

A la profesora Jocelyn, por toda su ayuda y apoyo durante la realización de esta memoria.

Y finalmente a Illary, mi modelo a seguir, por acompañarme en cada paso y ayudarme en mi camino, gracias.

# Tabla de Contenido

# Índice de Tablas

# Índice de Ilustraciones

# Capítulo 1

# Introducción

La coevaluación se distingue como un proceso en que el estudiante forma parte activa de la toma de decisiones para la evaluación, en donde logra conseguir una opinión más realista de sus propias capacidades, incentivando y favoreciendo la construcción de los conocimientos, elevando a la vez la responsabilidad personal y colectiva de los alumnos [38].

Para la carrera de Ingeniería Civil en Computación, el Departamento de Computación de la Universidad de Chile define como objetivo general "Formar ingenieros civiles **altamente capacitados para desarrollar e integrar nuevas tecnologías** de la información y comunicación, así como dirigir proyectos o empresas informáticas" y en particular, "Formar profesionales **capaces de aplicar tecnologías de frontera** y emprender actividades de innovación tecnológica; **de trabajar en forma independiente o dentro de un equipo** de especialistas o multidisciplinario" [33]. De aquí se puede observar la importancia del trabajo en equipo en la carrera, siendo un pilar fundamental para la formación de los estudiantes del departamento; y por tanto el desarrollar estas capacidades durante la formación académica resulta indispensable.

Dentro de la malla de la carrera, existen tres cursos con enfoque en desarrollo de software en equipo: *Ingeniería de Software I*, *Ingeniería de Software II* y *Proyecto de Software*. En ellos se forman grupos que durante el semestre deben desarrollar un proyecto de software aplicando tanto conocimientos técnicos como habilidades blandas. En sus evaluaciones generales cuentan con una sección de coevaluación, en donde los estudiantes deben calificarse entre ellos de forma objetiva con respecto a distintos aspectos de su trabajo.

Esta coevaluación se hace usando el Sistema de Coevaluación desarrollado por Riquelme [25] y Sánchez [31], al que se le denominará "sistema en producción" durante esta memoria, implementado como portal web y que principalmente permite que los profesores puedan agregar coevaluaciones, sus estudiantes responderlas y luego poder revisar los resultados. Aunque el sistema funciona, posee ciertas falencias en situaciones de grupos dinámicos, como por ejemplo, cuando alumnos son cambiados de equipo durante el semestre. En tal caso, se pierde el registro de su equipo anterior, haciendo complicado seguir su desempeño en el curso y verificar sus cambios de comportamiento durante este.

El sistema además clasifica las preguntas según dimensiones del trabajo en equipo (o áreas de desempeño) definidas por Luis Silvestre [28], y dependiendo de estas es cuanto contribuyen las calificaciones de cada pregunta a la nota final de la coevaluación. Actualmente, el sistema esconde la dimensionalidad de las notas y tampoco las muestra gráficamente, lo que podría ayudar a recabar las fortalezas y debilidades de los alumnos y los grupos.

El año 2019 se comenzó una actualización del sistema como trabajo dirigido por Galvez y López, integrando un nuevo modelo de datos y cambio de interfaces. A través de esto, es que se decide seguir con la actualización del sistema de coevaluaciones, extendiendo el desarrollo del trabajo dirigido hacia un programa sostenible que permita responder y revisar coevaluaciones, agregando funciones que hagan posible la gestión de estudiantes, equipos y profesores, el seguimiento de alumnos en los cursos y un *dashboard* para mostrar de forma gráfica los resultados obtenidos por los estudiantes.

Mediante un proceso iterativo incremental, y luego de analizar el sistema en producción y el trabajo dirigido mediante encuestas y entrevistas, diseñar los mockups del sistema y validarlos con profesores involucrados y de estudiar las distintas opciones de librerías gráficas, de manejo de archivos e interactivas, se comenzó el desarrollo del sistema, implementando con Django, Python y JavaScript un sistema para usuarios del tipo Alumno y Profesor, con el cual pudieran, desde el lado de los alumnos, responder y ver encuestas; y para profesores, administrar y gestionar todo el proceso relacionado a cursos, alumnos, evaluaciones y resultados.

Lo anterior fue validado durante 4 instancias, usando elementos como un programa en Python, entrevistas, demostraciones, encuestas SUS y el uso del sistema en el curso Ingeniería de Software I. Todas las pruebas con usuarios alumnos incluyeron tanto a novatos como avanzados en el uso del sistema, obteniendo resultados positivos con comentarios de mejora, y mediante las cuales también se pudo descubrir otros puntos de vista respecto al sistema, aportando estos al descubrimiento de futuras mejoras al sistema.

Como conclusión se considera que el sistema desarrollado cumple con los objetivos planteados, implementando todas las funciones esperadas y obteniendo un sistema autónomo de coevaluaciones, con posibilidades de ampliarse a otros ramos y mejorar las funcionalidades ya incluidas.

## 1.1. Objetivos

### Objetivo general

Completar el sistema del trabajo dirigido, agregándole robustez y usabilidad a través de la integración de funciones presentes en el sistema en producción y de otras nuevas, para que faciliten la gestión de cursos, seguimiento del desempeño de estudiantes y análisis del progreso de los alumnos con respecto a las coevaluaciones de sus compañeros.

### Objetivos específicos

Para cumplir el objetivo general, se establecen los siguientes objetivos específicos:

1. Analizar el sistema en producción, analizando las opciones de mejora disponibles.
2. Analizar el sistema del trabajo dirigido, encontrando sus fortalezas y debilidades, entendiendo el código y el modelo de datos.
3. Diseñar las nuevas vistas y los posibles cambios a aquellas que ya existen, siguiendo el diseño del trabajo dirigido.
4. Implementar funciones de gestión, seguimiento y análisis, tales como la gestión de profesores y alumnos, gráficos de los resultados de los estudiantes, sistema de seguimiento de estudiantes que han cambiado de grupos y avance a través de los cursos inscritos.
5. Validar el sistema implementado a través de entrevistas a profesores de los cursos que utilizan el Sistema de Coevaluación y a estudiantes que lo hayan utilizado, recabando niveles de satisfacción y oportunidades de mejora de la solución propuesta.

## 1.2. Solución Propuesta

La solución propuesta es continuar la actualización del sistema de coevaluación para los cursos *Ingeniería de Software I, Ingeniería de Software II y Proyecto de Software* ya comenzada con el trabajo dirigido de Galvez y López. De acuerdo a este desarrollo, se decide seguir con el *framework* de desarrollo web Django y el modelo de datos ya existente, haciendo una revisión a este para realizar las modificaciones necesarias para adaptarlo a esta continuación del proyecto.

Este sistema se enfocará principalmente en los usuarios profesores y alumnos, dejando al administrador usando directamente el sistema de administración integrado en Django y para los ayudantes una solución temporal, convirtiéndolo en una combinación entre usuarios profesor y alumno. Para estos dos usuarios principales, se resolvió implementar todas las funciones ya existentes en el sistema, tales como para profesores: la gestión de usuarios, cursos, equipos, coevaluaciones y envío de notificaciones por correo; mientras que para alumnos responder coevaluaciones, editar respuestas y vista de resultados.

Además de las funciones ya mencionadas, se propone la adición de:

- Seguimiento de desempeño: Creación de una vista que permita seguir el progreso de un alumno (o también equipos en el caso de usuarios profesores) a través del uso de tablas y gráficos para preguntas y dimensiones.
- Autoevaluaciones: Posibilidad de crear coevaluaciones que también sean autoevaluaciones, permitiendo que los alumnos puedan evaluar su propio rendimiento en el curso.
- Dinamismo de estudiantes entre grupos: Entregar la opción a los profesores de poder cambiar a alumnos de equipo dentro del sistema, manteniendo un historial del los grupos anteriores en los que ha participado; relacionando además las notas a los grupos para mantener un registro, el cual pueda complementarse con la nueva función de seguimiento de desempeño.

Todo esto dentro de un sistema que sea robusto, mantenible, fácil de usar y que mantenga el diseño heredado desde el trabajo dirigido.

## 1.3. Metodología de Trabajo

Para desarrollar el trabajo de memoria se utilizó una metodología iterativa incremental [1], dividida en 4 partes. La primera iteración se encargó de comprender en profundidad el proyecto, analizando los sistemas relacionados ya existentes y realizando el diseño de las vistas a implementar, evaluándolas con usuarios del sistema.

En la segunda iteración se arreglaron bugs existentes en el sistema actual (los que se detallan en la sección 3.2.2) y se implementaron de las vistas bases del sistema, como perfil de estudiante y profesor, vista del detalle de cursos y estudiantes y el *home page* para el alumno. En esta etapa se realizaron entrevistas individuales a 8 alumnos para evaluar la funcionalidad de contestar una coevaluación y autoevaluación, aplicándoles la encuesta SUS para evaluar la usabilidad de esta función.

Durante la tercera iteración se desarrolló el sistema de administración de cursos para los profesores, la edición de nuevos alumnos y equipos, y las nuevas funcionalidades de seguimiento y dinamismo de integrantes de equipos. Al finalizar esta etapa se evaluó el sistema en dos partes, en primer lugar una demostración a dos profesores del área de Ingeniería de Software, y en segundo lugar, se ocupó el sistema para realizar la coevaluación de la segunda iteración del curso *CC4401 - Ingeniería de Software I*, en donde los alumnos debieron contestar las evaluaciones de sus compañeros de equipo para luego, voluntariamente, responder la encuesta SUS.

La cuarta y última iteración consistió en escribir el informe y arreglar detalles que surgieron durante la evaluación antes mencionada.

## 1.4. Estructura del documento

El resto de la memoria se estructura en 6 capítulos. En el Capítulo 2: Estado del Arte, se habla de los conocimientos necesarios para entender el documento (patrones y herramientas), proyectos relacionados y situación actual. El Capítulo 3: Concepción de la Solución, entrega una mirada a sistemas anteriores, los flujos esperados para el sistema y los requisitos recabados a partir de lo anterior. Luego, el Capítulo 4: Diseño de la Solución, detalla las arquitecturas, modelo de datos y diseño empleado, mientras el Capítulo 5: Implementación de la Solución, se encarga de llevar lo anterior a la realidad. El Capítulo 6: Validación, habla de las distintas validación que se hicieron al sistema, su funcionamiento y la percepción de este. Finalmente, el Capítulo 7: Conclusión, concluye el informe y enuncia posibles trabajos a futuro para el sistema.

# Capítulo 2

# Estado del Arte

En esta sección se abordarán conceptos necesarios de saber para entender este trabajo y su desarrollo, además de los trabajos que sirvieron como antecedente a esta memoria y que presentan la idea original del proyecto. Este capítulo se cierra con la explicación del problema a resolver.

## 2.1. Patrones MVC y MTV

Modelo-Vista-Controlador o MVC por sus iniciales, es un patrón de arquitectura que se basa en la separación de las aplicaciones en tres componentes lógicos principales: modelo, vista y controlador [5].

**Modelo** es la componente que se encarga almacenar los datos y su lógica relacionada, siendo la capa de abstracción de la base de datos, manejándola como un objeto. Responde a instrucciones del controlador para actualizarse y le envía a la vez la información del modelo de datos.

La componente **Vista** es a quien se le designa la lógica de presentación, o *front-end*. Se encarga de mostrar los datos del modelo al usuario y su interacción con ellos.

**Controlador** es la componente en donde más trabajo de *back-end* se hace, siendo el intermediario entre las solicitudes del usuario a través de la vista y del modelo con sus datos. Envía comandos para obtener y actualizar información del modelo, y para cambiar la presentación de las vistas.

Django ocupa una variación de este modelo, denominado Modelo-Vista-Template o MVT [4][12]. En él, el **modelo** sigue teniendo la misma definición, encargándose de la base de datos, pero esta vez se comunica con la vista, no el controlador. La **vista** se encarga de formatear los datos del modelo y entregarlo al template. Por su parte, **template** es la capa de presentación, manteniendo todo lo que el navegador reproduce, siendo similar a la capa vista del modelo MVC.

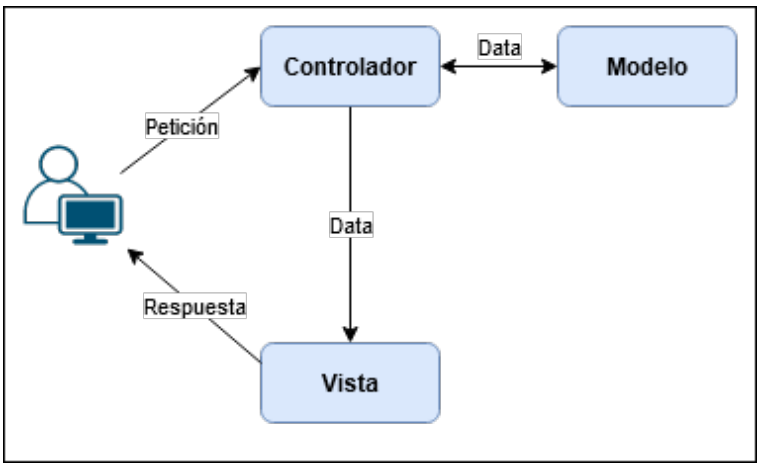



Figura 2.1: Patrón MVC

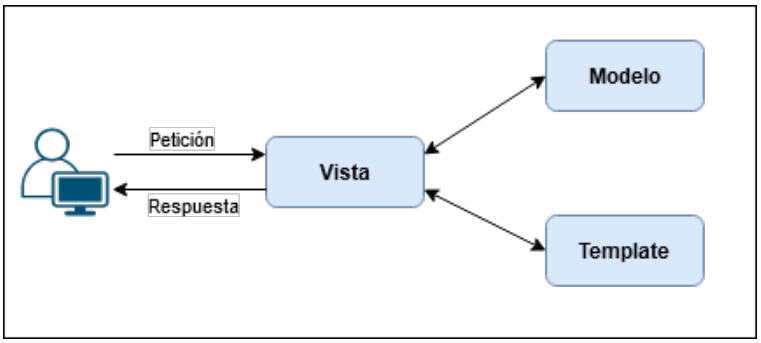



Figura 2.2: Patrón MVT

La principal diferencia entre MVC y MVT es que en el primero se debe escribir código específico para el controlador, mientras que en el segundo, el *framework*, en este caso Django, se encarga de él. Por ejemplo, cuando un usuario realiza una acción o hace una petición, en MVC se llama al controlador, el cual pide al modelo hacer cambios y actualiza la vista o retorna una vista basada en un modelo, pudiéndose decir que la vista es controlada por el modelo y el controlador.

En cambio, en MVT la vista correspondiente realiza la *query* en el modelo y recolecta el resultado de él, para luego completar el resultado en un template y mandarlo al usuario. A diferencia de MVC, en MVT la vista no está acoplada con el modelo, lo que hace que MVT tenga un menor acoplamiento y sea más fácil de modificar [10].

## 2.2. Django

Django es un *framework open-source* de desarrollo web escrito en Python que permite un desarrollo rápido de sitios web rápidos y mantenibles. Un *framework*, tal como lo es Django, agrupa a un conjunto de componentes usualmente necesarias en un sitio web, tales como autenticaciones de usuarios, panel de administración, formularios, formas de subir archivos, entre otros [11].

El flujo de una petición HTTP hecha por un usuario a un portal creado con Django se muestra en la figura 2.3. Usando el patrón MVT, se podría decir que Django "empareja" las componentes del patrón a archivos específicos, pero vemos además una nueva componente: **URLs**, la cual se encarga recibir las peticiones y redireccionarlas a su vista, basándose en la URL de la petición. Este mapeo de URLs permite también distinguir patrones de string o dígitos que aparezcan en una URL y pasarlos a la vista como parámetros [6].

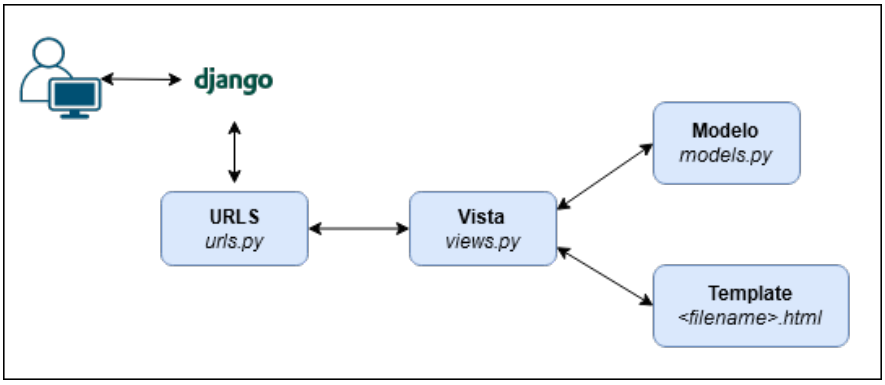



Figura 2.3: Patrón MVT en Django

## 2.3. Otras herramientas externas utilizadas

Algunas de las funciones implementadas requerían, además de la base en Django, Python y JavaScript, una librería adicional especializada. Cada una de estas librerías es mostrada a continuación, analizando además aquellas donde se barajaron distintas opciones.

Para las **visualizaciones**, se buscó tipos de gráficos simples, que enseñaran de forma clara la información y que, a la vez, no requirieran tener demasiada experiencia en gráficos para poder entenderlos. En un comienzo se decidió usar gráficos radar y lineal, utilizando para esto la librería **Chart.js**, pero en la práctica y al probar estos gráficos con datos reales, se observó que el gráfico lineal no entregaba nueva información al usuario, además de que resultaba difícil de leer debido a que muchos de los puntos de notas quedaban en el mismo lugar, produciendo que se sobrepusieran entre ellos, tal como se ve en la figura 2.4. Debido a la confusión que podía generar el gráfico lineal, es que se decidió dejar en el sistema solo el gráfico de tipo radar, el cual resultó fácil de leer y ayuda efectivamente a ver el avance de los alumnos a lo largo del curso.

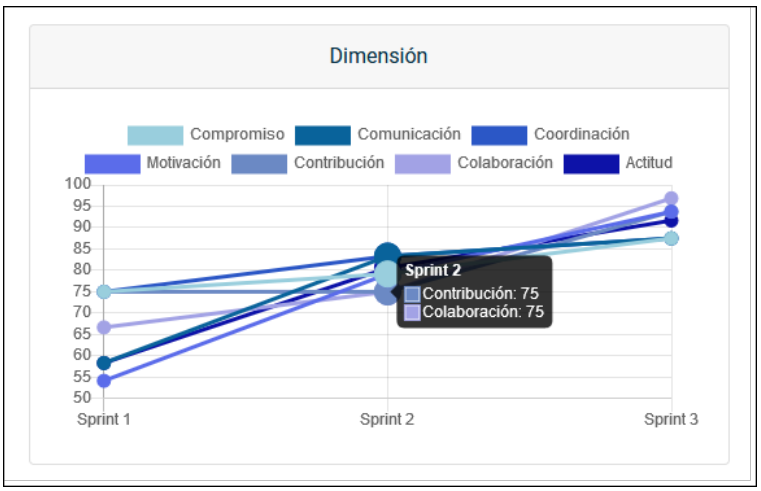



Figura 2.4: Problema gráfico lineal en concurrencia de puntos

Debido a que se tendrían gráficos con distinciones de colores, se hizo necesaria una librería que pudiera generar una gran cantidad de colores. **Random Color** [15] es la librería que se ocupó, ya que además de generar, permite elegir colores similares, lo que ayudaría a la estética de los gráficos.

Para el **manejo de planillas**, se investigaron las siguientes librería de Python: *xlwt*, *pyexcel_xls*, *pyexcel_xlsx*, *pyexcel-ods3* y *pyexcel*. xlwt es una librería para generar archivos xls y se consideró útil para realizar la exportación de resultados, función disponible para los usuarios de tipo profesor.

Debido a que existe la opción de incluir planillas provenientes desde notas en U-Cursos, es necesario tener funciones que soporten la lectura de archivos con extensión xls y ods. pyexcel_xls [23], pyexcel_xlsx [24] y pyexcel-ods [22] fueron las primeras librerías que se encontraron para la lectura de archivos xls, xlsx y ods, respectivamente. Todas ellas comentan en su documentación que son usadas junto a la librería **pyexcel** [21], por lo que se decide investigar sobre ella, descubriendo que por sí sola soporta variados formatos y entre ellos los necesarios, decidiendo ocupar esta última librería.

También, con la idea de aplicarlo al **cambio de alumnos entre grupos**, se investigaron librerías que permitieran realizar *drag and drop* entre elementos de varias listas, pudiendo dejar estáticos algunos de los elementos (los que corresponderían a los nombres de los equipos). En primer lugar, se encontró la librería Draggable de Shopify [27], la que cumplía con los requisitos funcionales deseados, más no con la estética, contando con un formato muy diferente al implementado en la página y una carga de animaciones que no iban con el diseño ocupado. A continuación se encontró Dragula [2], la cual funcionaba bien aunque siendo algo complicada de entender, por lo que se siguió buscando y se encontró la librería definitiva para este trabajo: **Sortable** [35]. Esta librería pertenece a JQuery UI, paquete de interacciones, efectos, widgets y temas construido sobre JQuery, y que además ya era utilizada por el sistema, lo que significaba no tener que instalar nada nuevo. Su uso era extremadamente sencillo y su documentación clara, pudiendo ocuparla y adecuarla a las necesidades del sistema en poco tiempo. Es por ello que se le consideró la más conveniente de usar.

Como ya se comentó, Sortable no es la única herramienta sacada desde JQuery UI. **Autocomplete** [34] es un widget de este mismo paquete que sugiere opciones dentro de una búsqueda de forma dinámica dentro de los elementos que se le entreguen, según se observa en la figura 2.5. Su utilidad se proyectó como parte de los inputs de **búsqueda de nombres de estudiantes**, haciendo más fácil para los profesores encontrarlos cuando fuera necesario.

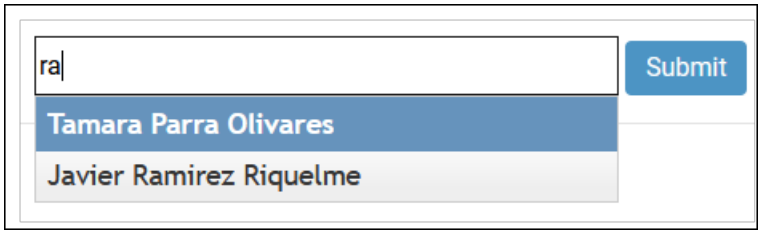



Figura 2.5: Widget Autocomplete de JQuery UI

Por último, vale la pena mencionar la que es la librería externa más importante de este sistema: **JQuery**. Esta librería de JavaScript permite la manipulación de DOM[1], el manejo de eventos y Ajax.

## 2.4. Encuesta SUS

Pensando en métodos para la validación del sistema se llegó al Sistema de Escalas de Usabilidad o SUS [36], herramienta que se ocupa para medir la usabilidad de un objeto, dispositivo o aplicación. Consiste en un cuestionario de 10 preguntas con 5 opciones de respuestas, las que van desde 1 "muy en desacuerdo" a 5 "muy de acuerdo". A continuación se muestra la lista de preguntas que contempla la encuesta, la que no se debe modificar ya que afecta en los resultados del test:

1. Creo que usaría este sistema frecuentemente.
2. Encuentro este sistema innecesariamente complejo.
3. Creo que el sistema fue fácil de usar.
4. Creo que necesitaría ayuda de una persona con conocimientos técnicos para usar este sistema.
5. Las funciones de este sistema están bien integradas.
6. Creo que el sistema es muy inconsistente.
7. Imagino que la mayoría de la gente aprendería a usar este sistema en forma muy rápida.
8. Encuentro que el sistema es muy difícil de usar.
9. Me siento confiado al usar este sistema.
10. Necesité aprender muchas cosas antes de ser capaz de usar este sistema.

Para calcular los resultados se consideran las siguientes reglas: a cada una de las respuestas en las preguntas impares se les resta 1, mientras que para las preguntas pares, a 5 se le resta el valor de la respuesta. Estos valores se suman y se multiplican por 2,5.

[1]DOM (Modelo de Objeto de Documento) es una representación orientada a objetos de la página web, que proporciona estructura al documento y define la forma en que puede ser modificado. [7]

De esto se obtiene un puntaje entre 0 y 100, el cual no es de ninguna forma interpretado como porcentaje. Basado en investigaciones, un puntaje SUS mayor a 68 se considera por sobre el promedio y cualquier cosa menor a eso es inferior al promedio. Sobre 80 es el puntaje recomendado [32].

## 2.5. Estado del sistema actual

### 2.5.1. Sistema de coevaluaciones en producción

Originalmente la integración y el procesamiento de las coevaluaciones era un proceso manual, lo cual llevaba consigo una alta posibilidad de errores y pérdida de horas docente. Para solucionar esto es que Riquelme desarrolla en el año 2014 la memoria "Sistema de Evaluación del Desempeño de los Miembros de un Equipo de Desarrollo de Software" [25], en donde se plantea la automatización del proceso, mediante el desarrollo e implementación de un "sistema de evaluación del desempeño de los miembros de un equipo de trabajo, particularmente para estudiantes que desarrollan software en equipos, dentro de un escenario universitario".

Este sistema cuenta con 3 tipos de usuarios: Administrador, Profesor y Alumno, en donde define para el profesor herramientas para evaluar las aptitudes de un alumno, crear equipos de trabajo con sus miembros, notificar por correo de las coevaluaciones a los estudiantes y obtener los resultado de forma instantánea al cerrar la coevaluación. Por su parte, los alumnos pueden evaluar a sus compañeros de equipo y obtener sus propios resultados. Por último los administradores pueden gestionar todos los usuarios, cursos, coevaluaciones y respuestas.

Para comprobar su correcto funcionamiento se realizó un experimento con el curso *CC5401 Ingeniería de Software II* durante el semestre Otoño 2014, en donde se formaron 3 grupos de 5 alumnos cada uno, realizando 2 coevaluaciones con distintas fechas y logrando resultados positivos en ambas, pudiendo los alumnos completar la tarea de responder una coevaluación y ver sus resultados inmediatamente una vez cerrada la coevaluación. La experiencia fue también positiva para el profesor y los ayudantes del curso, los que consideraron que este acercamiento disminuiría el tiempo y la cantidad de errores humanos producidos en el proceso manual de coevaluación.

Es importante notar que este sistema cuenta con la particularidad de que la coevaluación se crea mediante un formulario dinámico, siendo las preguntas creadas por el profesor cada vez que se abre una convocatoria, haciendo al sistema más flexible con respecto a los cursos que podían ocupar el sistema pero coartando la posibilidad de seguimiento de la evolución de notas y desempeño de los alumnos, debido a la independencia de las preguntas entre coevaluaciones.

### 2.5.2. Actualización del sistema de coevaluación en producción

Mediante un estudio de la memoria de Riquelme, Sánchez levanta su memoria "Extensión de un Sistema de Coevaluación de Miembros de Equipos de Desarrollo de Software" [31], en la que detecta limitaciones del sistema desarrollado, tales como que los datos procesados quedan enlazados a las coevaluaciones respectivas, impidiendo que se puedan realizar seguimientos de notas individuales o grupales; la falta de procesos de análisis de datos o mecanismos que

ayuden a los estudiantes a ver sus debilidades; problemas de usabilidad y la inexistencia de un rol ayudante, teniendo el profesor que compartir su cuenta.

A partir de esto, elabora una actualización al sistema que incluye una reingeniería al sistema de coevaluaciones mediante el cambio de interfaces y el modelo de datos, la adición de mecanismos de seguimiento, la modificación y categorización de los puntos evaluados mediante la incorporación de dimensiones, estandarizando el proceso través del uso de 10 preguntas específicas, según la investigación realizada por Silvestre, la que se abordará en la sección 2.5.3. Además, se agrega el envío de sugerencias a los estudiantes según su rendimiento, se repararon errores de usabilidad y se agregó el tipo de usuario Ayudante.[2]

Para evaluar el sistema se realizaron dos pruebas: una en la fase de desarrollo y otra sobre el producto final, con ayuda de los integrantes de los cursos *CC5401 Ingeniera de Software II* y *CC5402 Proyecto de Software* durante el semestre Primavera 2015. Estos cursos estaban compuestos de 7 y 29 alumnos respectivamente, formándose 1 equipo para el primer curso y 5 para el segundo. Las pruebas consistieron en realizar todo el proceso de coevaluación, verificando que al estar cerrada la coevaluación los alumnos pudieran acceder a los gráficos de seguimiento de rendimiento, y el correcto funcionamiento del sistema de sugerencias. El rol de Ayudante se comprobó creando una cuenta con este rol y encargándole tareas como crear coevaluaciones, publicarlas y ver resultados. Todas las pruebas se pasaron con éxito.

Existen dos aspectos importantes en esta actualización: la estandarización de las coevaluaciones y el sistema de seguimiento de alumnos. Ambas funcionalidades se mantendrán en el nuevo diseño, aunque con algunos cambios con respecto al seguimiento para así poder aumentar la claridad de las comparaciones.

### 2.5.3. Estandarización de la coevaluación

En la tesis "Diseño de Equipos de Desarrollo de Software en Escenarios Universitarios" de Silvestre [28], se propone una heurística para formar equipos de desarrollo de software basados en el perfil psico-social de los estudiantes de programas de ciencias de la computación. A partir de esto es que se estandarizan las coevaluaciones, realizando 10 preguntas específicas agrupadas en 7 dimensiones, según se observa en la tabla 2.1. De ellas, las 8 primeras tienen una escala del 1 ("Nunca") a 5 ("Siempre") y las últimas 2 como preguntas abiertas.

Por su parte y según un estudio de opinión hecho por Sánchez [31], se establecieron dos tipos de cursos: con y sin horas fijas. La diferencia entre ellos afecta la contribución de las dimensiones en la nota final. Por ejemplo, se considera que para cursos con horas fijas (como *CC5402*) el *Compromiso* es obligatorio, mientras que para cursos sin horas fijas (como *CC5401*), el *Compromiso* es un extra en el proyecto. Con esto en mente, las ponderaciones quedan según la tabla 2.2 y se calculan de acuerdo a la cantidad de puntos por pregunta, es decir, 1 punto equivale a un 0 % del puntaje de la pregunta, 2 puntos al 25 %, 3 puntos al 50 %, 4 puntos al 75 % y 5 puntos al 100 %. La nota final se calcula sumando los puntajes obtenidos de cada pregunta más el punto base.

[2]Una lista completa de las funcionalidades implementadas por el sistema de Riquelme y Sanchéz se encuentra en la tabla 3.1.

[3]La dimensión *Colaboración* además de incluir la pregunta 7, considera la variable *Contribución*

| # | Pregunta | Dimensión |
|---|---|---|
| **1** | Demuestra compromiso con el proyecto. | *Compromiso* |
| **2** | Cumple de manera adecuada con las tareas que le son asignadas. | *Compromiso* |
| **3** | Demuestra iniciativa para lograr el éxito del proyecto. | *Motivación, Actitud* |
| **4** | Mantiene buena comunicación con el resto del equipo. | *Comunicación* |
| **5** | Mantiene buena coordinación entre sus tareas y las de sus pares. | *Comunicación* |
| **6** | La calidad de su trabajo es la apropiada para lograr el éxito del proyecto. | *Contribución* |
| **7** | Ofrece apoyo en las tareas que van más allá del rol asignado. | *Motivación, Actitud, Colaboración*[3] |
| **8** | Es capaz de admitir sus equivocaciones y recibir críticas. | *Actitud* |
| **9** | Fortalezas. | |
| **10** | Debilidades. | |

Tabla 2.1: Preguntas y las dimensiones en las que están categorizadas.

| Pregunta | Puntaje **con** hora fija | Puntaje **sin** hora fija |
|---|---|---|
| 1 | 0,459 | 0,553 |
| 2 | 1,112 | 1,204 |
| 3 | 0,588 | 0,553 |
| 4 | 0,988 | 0,947 |
| 5 | 0,918 | 0,789 |
| 6 | 0,653 | 0,651 |
| 7 | 1,047 | 1,066 |
| 8 | 0,235 | 0,237 |
| **Total** | **6,0** | **6,0** |

Tabla 2.2: Puntaje de preguntas según tipo de curso.

### 2.5.4. Situación actual

Actualmente en los cursos *CC4401 - Ingeniería de Software I*, *CC5401 - Ingeniería de Software II* y *CC5402 - Proyecto de Software* se utiliza el Sistema de Coevaluación desarrollado por Riquelme y actualizado por Sánchez. Por otra parte, en el año 2019 como trabajo dirigido, Daniel Galvez y Rupali López comenzaron una actualización del sistema, con un nuevo modelo de datos desarrollado por Milenko Tomic y cambio de interfaces según el diseño de *cevicheCorp* en el ramo de *Ingeniería de Software I* [9]. Es este desarrollo el que se planea seguir, reimplementando funciones existentes en el Sistema de Coevaluaciones en producción y añadiendo otras nuevas.

El sistema actual, aunque cumple con el objetivo del trabajo dirigido de levantar un MVP (mínimo producto viable) del sistema de coevaluaciones, considerando la gestión de cursos, equipos de trabajos y coevaluaciones [29], no es posible de utilizar por sí solo y es necesario completarlo para lograr llegar al estado del sistema en producción, pudiendo agregar nuevas funcionalidades debido a que ya existe un trabajo previo. Vistas como el *home page* del profesor y la gestión de sus coevaluaciones se encuentran ya desarrolladas según muestran las figuras 2.6 y 2.7, respectivamente. El código de este sistema se encuentra en un respositorio de Github [9] e incluye el proyecto llamado "SistemaCoevaluaciones" y la app "coevaluaciones", con documentación de los trabajos dirigidos y un *script* para poblar inicialmente la base de datos, manteniendo un flujo con 4 ramas: *master*, *develop*, *dgalvez* y *rglopez*.


Figura 2.6: Vista *home page* profesor mientras se agrega una coevaluación.

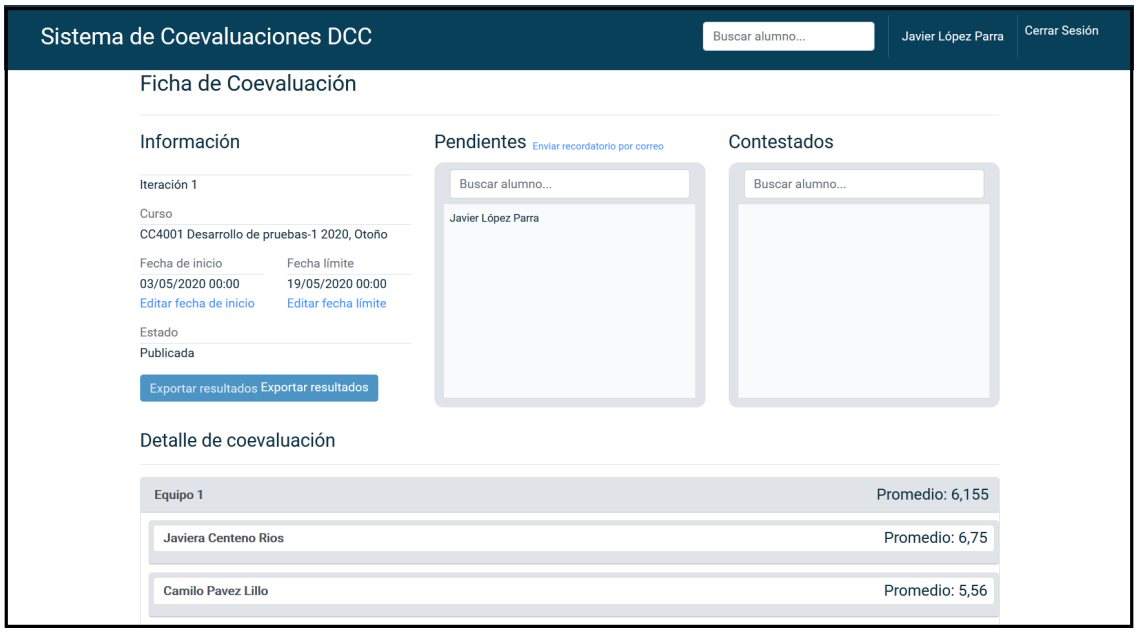



Figura 2.7: Vista gestión de coevaluación para profesor.

La figura 2.8 muestra un extracto de la ficha de coevaluación que responden los estudiantes en el sistema actual de acuerdo a lo visto en la sección 2.5.3, tabla 2.1.

La arquitectura física del sistema actual fue desarrollada en *Django* versión 1.5 sobre un servidor *Apache 2* y con un control de paquetes en *Python virtualenv*. En cuanto a la arquitectura lógica, tal como se verá con mayor detalle más adelante, el sistema actual maneja en el modelo de datos entidades como *Integrante*, *Curso*, *Grupo*, *Coevaluación* y *Respuesta* para representar el sistema, utilizando un sistema de gestión de base de datos *sqlite3*.

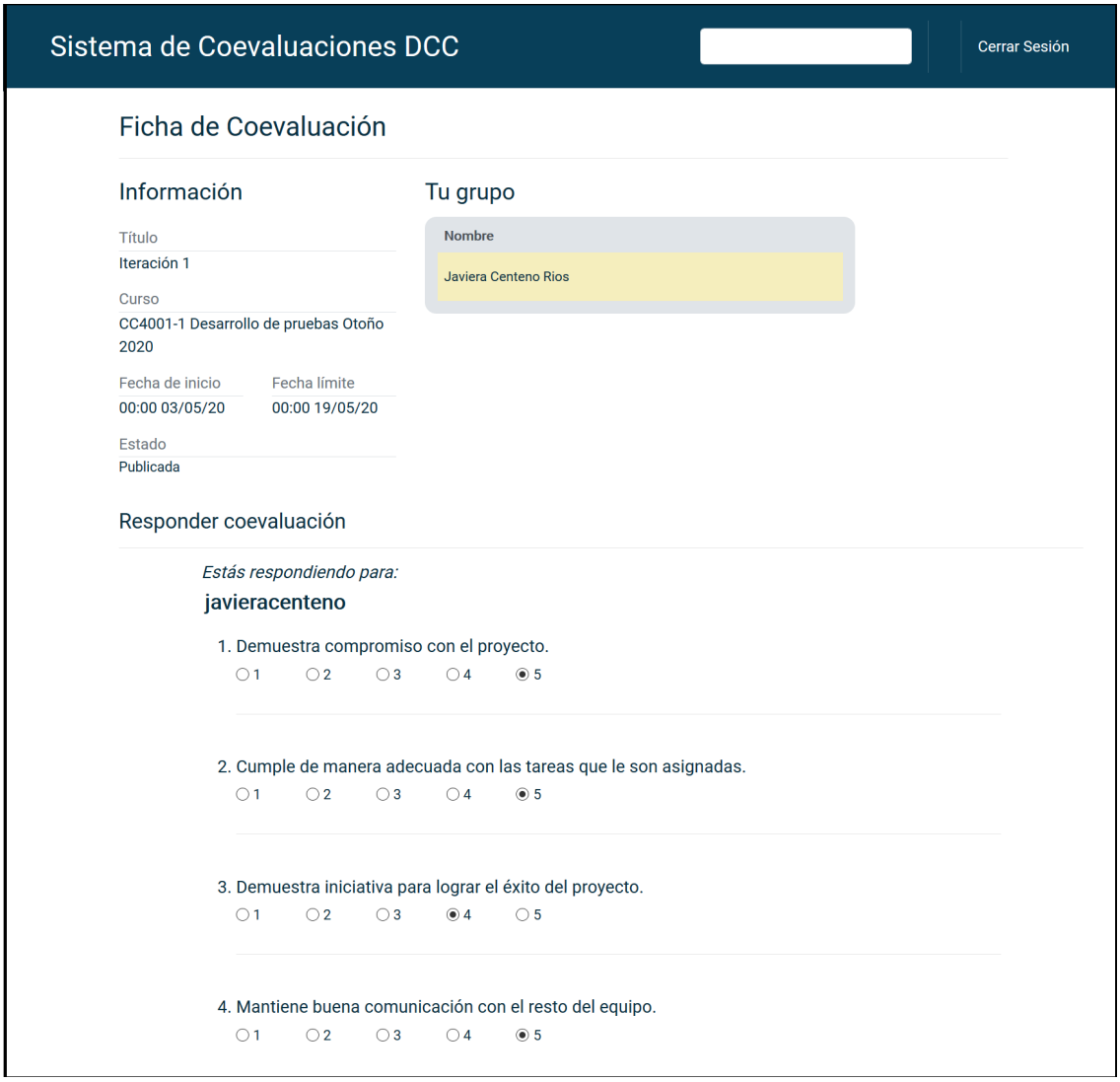



Figura 2.8: Extracto ficha de coevaluación del sistema actual.

## 2.6. Problema

El problema general abordado en esta memoria es la necesidad de actualización del sistema de coevaluaciones actualmente usado. Esto, debido a que su mantenimiento ya resulta ser demasiado complejo, además de la falta de algunas funciones que se descubrieron como necesarias durante el uso del sistema.

Mediante entrevistas y encuestas a los usuarios del sistema, según se detalla con mayor profundidad en el Capítulo 3.2.1, se encuentra que las principales falencias del sistema actual (y necesarias de mejorar) son la usabilidad, el diseño, y el manejo de errores, factores que son esenciales especialmente para los usuarios de tipo Profesor, debido a que su uso del sistema es más avanzado y requieren mayores funcionalidades para cumplir su propósito final.

Tal como se describió anteriormente, el sistema carece de algunas funciones que actualmente le aportarían gran valor, con algunas siendo completamente necesarias debido al crecimiento y evolución de los cursos. Específicamente estas funcionalidades son el cambio de

alumnos entre equipos, la posibilidad de hacer seguimiento a los alumnos durante el semestre y la opción de realizar autoevaluaciones en conjunto a las coevaluaciones.

A partir de lo anterior, se define como una solución aceptable un sistema autónomo, útil y amigable, que presente todas las funciones ya implementadas en el sistema anterior y las nuevas antes mencionadas, integrando cambios en su estética y usabilidad, manteniendo a la vez un código limpio y mantenible.

# Capítulo 3

# Concepción de la Solución

Para hablar sobre cómo se llegó a la solución a implementar, se comenzará explicando el flujo del sistema. Después, se mostrará el análisis que se hizo al sistema en producción, lo cual da paso al levantamiento de requisitos.

## 3.1. Flujo del sistema

La solución debe llevar a cabo exitosamente una serie de procesos relacionados a los elementos y al funcionamiento del sistema. En la tabla 3.1, donde la columna "SP", "TD" y "TM" se refieren a "Solución en Producción", "Trabajo Dirigido" y "Trabajo Memoria", respectivamente, se muestra un comparativo entre las distintas funcionalidades existentes en los sistemas desarrollados, en donde el color verde indica implementado, el amarillo no completamente implementado y el rojo es no implementado.

Existen algunas funciones más complejas que otras para entender como se mueven sus elementos y estados por el sistema, ya que no hay un desarrollo lineal de estos, manteniendo ciertas restricciones para su avance o retroceso en el proceso. Debido a que estas funciones son especialmente importantes, a continuación se muestran diagramas de flujos para explicarlas. Como el resto de las funciones maneja un flujo simple de CRUD (creación, lectura, actualización o borrado), se prefirió no indagar en ellas.

### 3.1.1. Ingreso al sistema

Para este proceso, graficado en la figura 3.1, el usuario escribe su nombre y contraseña, las que en caso de ser acertadas, el sistema se encarga de revisar el tipo de usuario que está ingresando, y de acuerdo a eso entrega el vista pertinente. Para usuarios de tipo Ayudante, el sistema muestra una vista intermedia, la cual pregunta si desea ingresar como alumno o profesor y dependiendo de esta decisión es cual de las vistas mostrará luego.

| Usuario | Función | SP | TD | TM |
|---|---|---|---|---|
| Administrador | Autenticación para entrar al sistema | | | |
| | Gestionar usuarios | | | |
| | Gestionar cursos | | | |
| | Gestionar coevaluaciones | | | |
| | Gestionar respuestas | | | |
| Profesor | Autenticación para entrar al sistema | | | |
| | Gestionar sus cursos | | | |
| | Gestionar coevaluaciones | | | |
| | Gestionar equipos | | | |
| | Agregar o eliminar alumnos de los cursos y equipos | | | |
| | Mantener registro de los equipos por los que ha pasado un alumno | | | |
| | Publicar evaluaciones con notificación por correo a los estudiantes | | | |
| | Ver resultados de coevaluaciones de sus cursos | | | |
| | Ver resultados mediante dashboard | | | |
| | Escribir sugerencias para los alumnos | | | |
| | Descargar resultados en formato Excel | | | |
| | Importar estudiantes y sus equipos desde planillas Excel | | | |
| Ayudante | Autenticación para entrar al sistema | | | |
| | Ver resultados de coevaluaciones de sus cursos | | | |
| | Escribir sugerencias para los alumnos | | | |
| Alumno | Autenticación para entrar al sistema | | | |
| | Responder coevaluaciones publicadas | | | |
| | Editar respuestas de coevaluaciones abiertas | | | |
| | Ver sus resultados en coevaluaciones cerradas | | | |
| | Ver sus resultados mediante dashboard | | | |

Tabla 3.1: Funciones según usuario.

### 3.1.2. Gestión coevaluaciones

Las coevaluaciones cuentan con un flujo enfocado en los estados que pasan durante su ciclo, el cual comienza con la creación de la coevaluación, su publicación para que sean contestadas, su paso a finalizada para que no se ingresen más respuestas pero se pueda extender el plazo de la evaluación y finalmente con respuestas publicadas, en donde ya no es posible dar más plazos y los resultados son enviados a los alumnos. Este flujo queda representado gráficamente en la figura 3.2

### 3.1.3. Agregar grupo a curso

En la figura 3.3 se puede ver que el agregar un nuevo grupo a un curso depende de si el nombre del curso ya existe dentro del curso. De no ser así, el grupo se crea y es asociado a todas las coevaluaciones que el curso posea.

## 3.2. Análisis sistemas existentes

### 3.2.1. Análisis sistema en producción

El sistema desarrollado por Riquelme y Sanchéz, o "sistema en producción" (disponible en el siguiente link), se encuentra en uso desde el año 2015 para los cursos de *Ingeniería de*

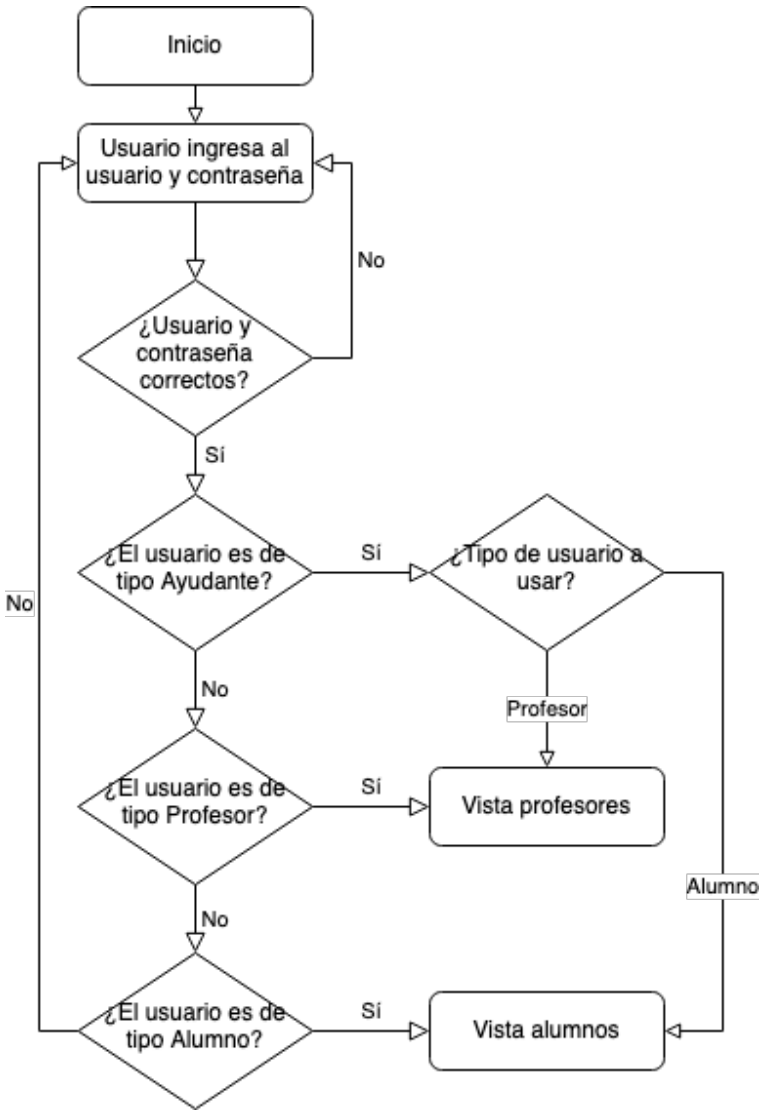



Figura 3.1: Diagrama de flujo para ingreso a sistema.

*Software II*, *Proyecto de Software* y posteriormente para *Ingeniería de Software I*. Su análisis se realizó de 3 formas: una entrevista con Milenko Tomic, usuario regular del sistema en los roles de Administrador y Profesor; un análisis propio del sistema desde el punto de vista del alumno y una encuesta realizada a 30 alumnos del departamento de computación.

#### 3.2.1.1. Entrevista sobre sistema de coevaluación para profesores y administradores

La entrevista con Tomic se realizó vía videollamada con una duración de aproximadamente 40 minutos, en donde el entrevistado recorrió el sistema indicando distintos aspecto sobre él. En cuanto a lo positivo del sistema de coevaluación en producción para profesores se encuentra el buen manejo de cambio de fechas de coevaluaciones, la existencia de un contador de coevaluaciones respondidas (permitiendo que se pueda saber cuando es necesario dar más tiempo para contestar), que la interfaz de resultados muestre cuando las respuestas vienen con o sin comentarios y la facilidad de la creación de cursos a través de importar plantillas.

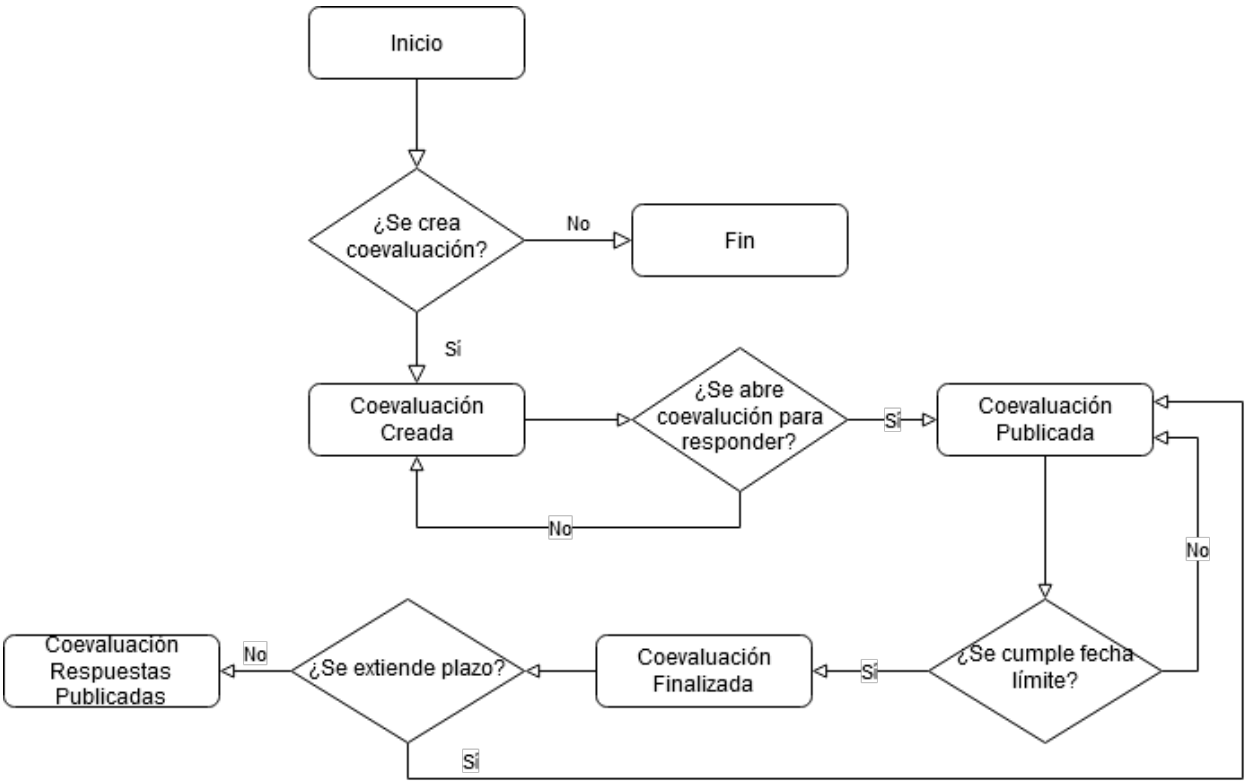



Figura 3.2: Diagrama de flujo para gestión de coevaluaciones.

Por otro lado, se pudo acotar que los principales problemas del sistema para los usuarios Profesor son la usabilidad, el diseño y el manejo de errores, específicamente:

- En la interfaz de resumen (similar a la de la figura 3.10) los gráficos demoran en cargar y la posición de estos en la pantalla es mala.
- En la interfaz de resultados por alumno (figura 3.4) los comentarios se encuentran escondidos, debiendo hacer click sobre un botón para poder verlos, incluso cuando el comentario es vacío. Además, la plantilla descargada con los resultados es compleja de entender, ya que posee campos innecesarios y todos los grupos se encuentran en la misma hoja, haciendo difícil su lectura.
- La interfaz que muestra los cursos (figura 3.5) no es relevante, ya que todo lo que aparece ahí se puede ver y hacer desde otras interfaces. Para lo único que se usa es para crear cursos y asignarles sus respectivos integrantes.
- El flujo de la creación de coevaluaciones no es intuitivo, debiendo primero crear la coevaluación y luego editarla para poder añadir las fechas y publicar. También, al crear una coevaluación se debe identificar si se trata de la última coevaluación, lo cual solo pareciera servir para formatear un texto.
- El envío de correos produce que el sistema, en vez de funcionar asíncronamente, espere hasta que se terminen de enviar todos los correos y, en caso de que este proceso falle en algún momento, el sistema no informa de este error.
- Las sugerencias no se usan frecuentemente.
- Cambiar integrantes de grupos resulta poco conveniente, ya que el sistema no está

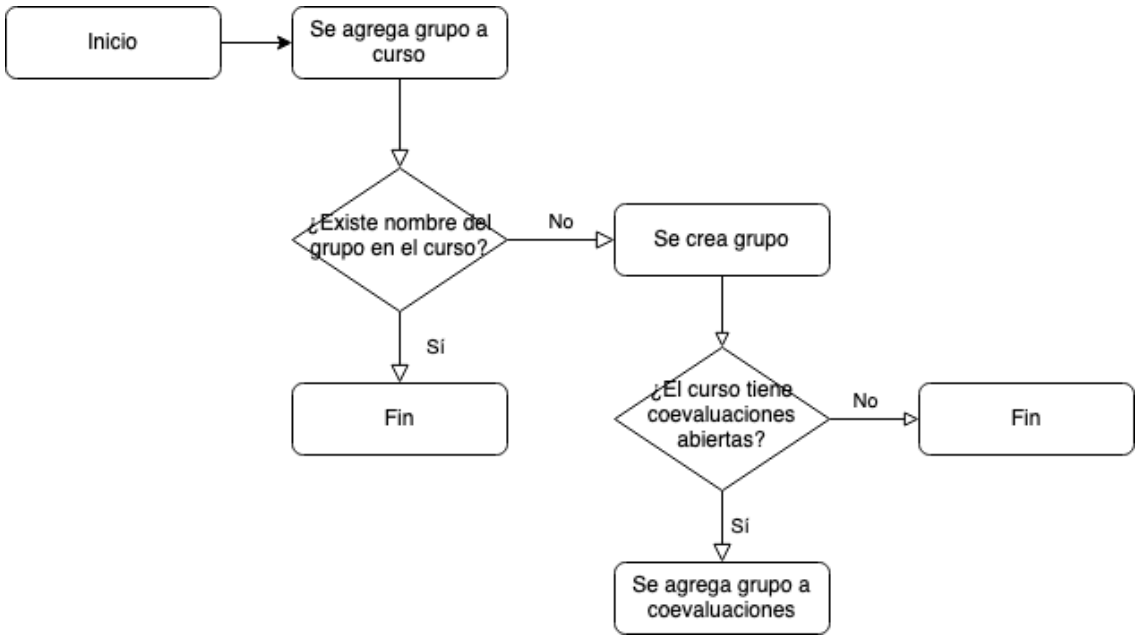



Figura 3.3: Diagrama de flujo para agregar grupos a curso.

diseñado para esto.

- No existe la opción de autoevaluación.
- Si no están todos los alumnos en un grupo no se puede crear una coevaluación, y el sistema no avisa sobre esto.
- El diseño no es adaptativo (responsive).
- Existen fallas del sistema en que no se logra identificar su razón.

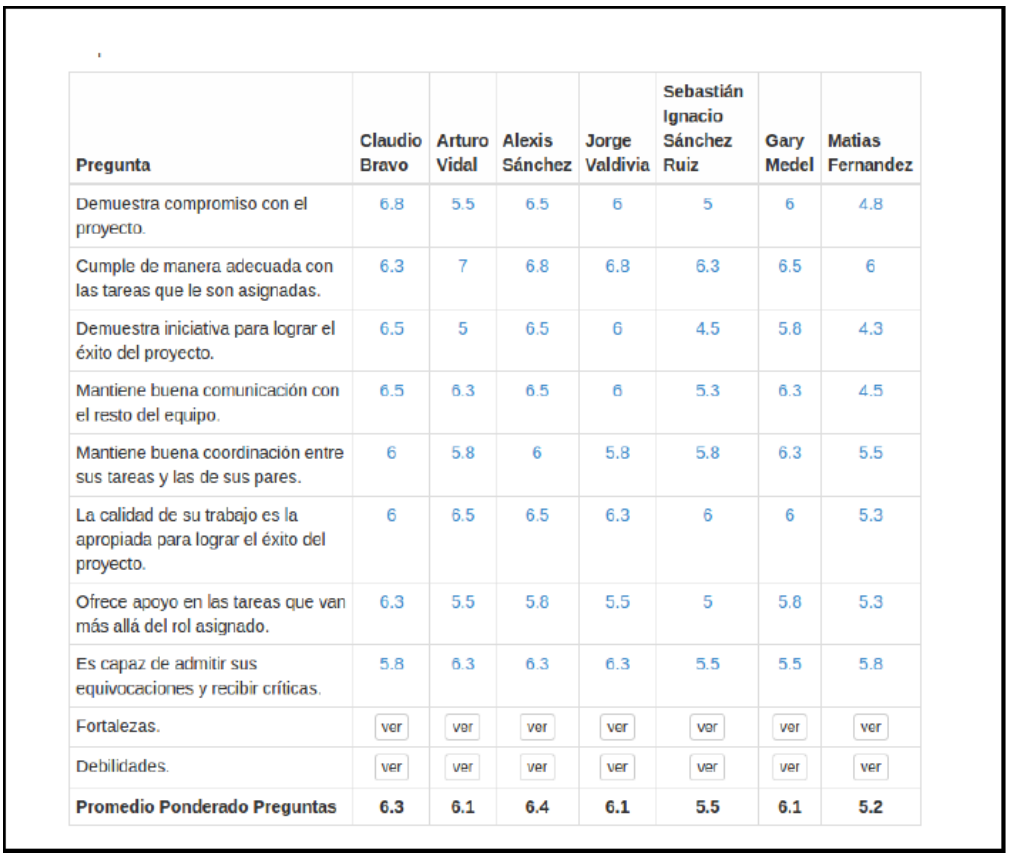
Para el usuario Administrador, Tomic indicó que el sistema no tenía mayores problemas, pero que se podría simplificar a solo una interfaz para crear los cargos de Profesores y Ayudantes.

#### 3.2.1.2. Evaluación sistema de coevaluación para estudiantes

Para realizar la evaluación del sistema de estudiantes, la autora de este documento investigó sobre evaluación heurística, específicamente los 10 criterios de Nielsen [13], los cuales fueron usados durante este proceso.

Con respecto a lo bueno del sistema, se encuentra la visibilidad de las fechas límites y estados de las coevaluaciones, la posibilidad de editar respuestas de una coevaluación mientras esta se encuentra abierta y la simpleza de las vistas. En tanto a lo negativo, se encontraron problemas consistentes a los del usuario Profesor, incluyéndose:

- Inconsistencia en los textos de las páginas, por ejemplo, el login del portal anuncia que se podrá *crear una coevaluación y comenzar a evaluar*.
- El diseño de la página hace que el espacio no sea bien aprovechado, produciendo que el usuario realice pasos extras para, por ejemplo, ver las coevaluaciones del curso en el que se encuentra actualmente.



| Pregunta | Claudio Bravo | Arturo Vidal | Alexis Sánchez | Jorge Valdivia | Sebastián Ignacio Sánchez Ruiz | Gary Medel | Matias Fernandez |
|---|---|---|---|---|---|---|---|
| Demuestra compromiso con el proyecto. | 6.8 | 5.5 | 6.5 | 6 | 5 | 6 | 4.8 |
| Cumple de manera adecuada con las tareas que le son asignadas. | 6.3 | 7 | 6.8 | 6.8 | 6.3 | 6.5 | 6 |
| Demuestra iniciativa para lograr el éxito del proyecto. | 6.5 | 5 | 6.5 | 6 | 4.5 | 5.8 | 4.3 |
| Mantiene buena comunicación con el resto del equipo. | 6.5 | 6.3 | 6.5 | 6 | 5.3 | 6.3 | 4.5 |
| Mantiene buena coordinación entre sus tareas y las de sus pares. | 6 | 5.8 | 6 | 5.8 | 5.8 | 6.3 | 5.5 |
| La calidad de su trabajo es la apropiada para lograr el éxito del proyecto. | 6 | 6.5 | 6.5 | 6.3 | 6 | 6 | 5.3 |
| Ofrece apoyo en las tareas que van más allá del rol asignado. | 6.3 | 5.5 | 5.8 | 5.5 | 5 | 5.8 | 5.3 |
| Es capaz de admitir sus equivocaciones y recibir críticas. | 5.8 | 6.3 | 6.3 | 6.3 | 5.5 | 5.5 | 5.8 |
| Fortalezas. | ver | ver | ver | ver | ver | ver | ver |
| Debilidades. | ver | ver | ver | ver | ver | ver | ver |
| **Promedio Ponderado Preguntas** | **6.3** | **6.1** | **6.4** | **6.1** | **5.5** | **6.1** | **5.2** |

Figura 3.4: Resultados de una coevaluación para profesor

- Al ver sus resultados, el estudiante no puede visualizar de inmediato los comentarios que se le hicieron, debiendo ampliar las preguntas de fortalezas y debilidades. También ocurre que el sistema muestra incluso los comentarios vacíos.
- En la interfaz de Resumen el título cambia a "Seguimiento de resultados". Además, los gráficos mostrados en esta página no resultan de mayor utilidad y pueden llegar a ser difíciles de leer.
- La interfaz de Cursos solo muestra un acordeón que contiene los cursos que se dieron en cada año más una breve descripción de estos. La información presente aquí no resulta ser de mayor utilidad para el alumno, pudiendo encontrarla en la interfaz de Coevaluaciones o a través de U-Cursos.
- El *feedback* del sistema referente a en qué interfaz se encuentra no es del todo clara, en especial al ir a las secciones "Acerca de" y "Cambiar contraseña".
- Al cambiar la contraseña, el sistema no verifica la seguridad de la nueva contraseña mas allá de que tenga al menos 6 caracteres. Por ejemplo, acepta contraseñas como *"aaaaaa"*.
- Luego de cerrar la sesión y devolverse a la página anterior el sistema permite seguir viendo el contenido de esta. Recién cuando se interactúa con el sistema se vuelve a la página de login avisando que se necesita loguearse para ver el contenido.

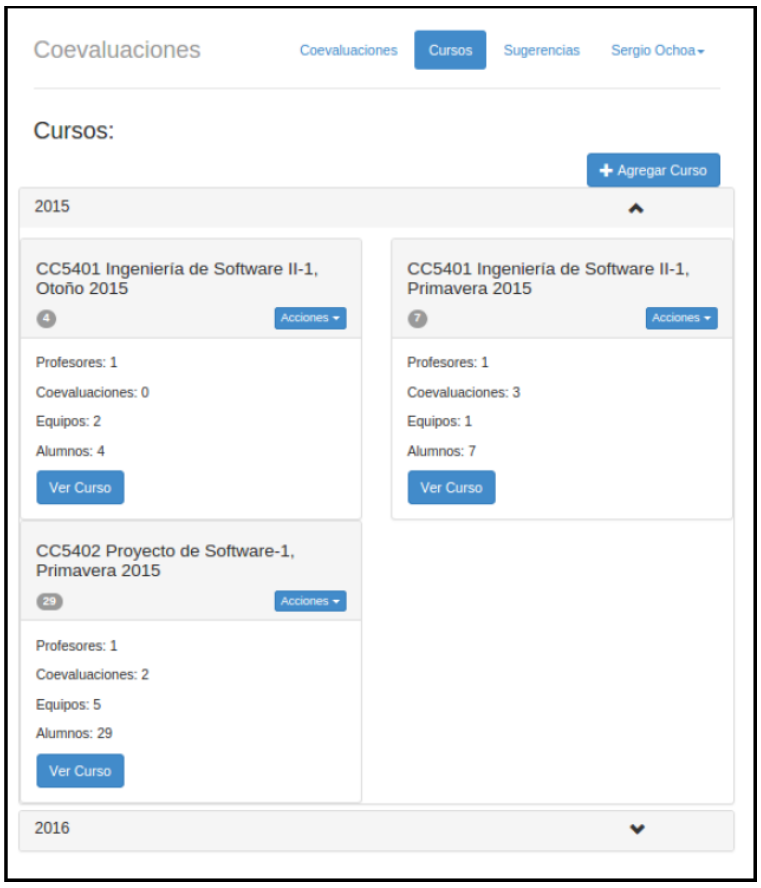



Figura 3.5: Interfaz de Cursos para profesor

#### 3.2.1.3. Encuesta a estudiantes sobre sistema de coevaluación

Finalmente, se conoció la opinión de 31 estudiantes de Ingeniería Civil en Computación del Departamento de Computación con respecto al sistema de coevaluación a través de una encuesta de Google Forms, la cual fue publicada en el foro "Alumnos DCC" de U-Cursos y enviada por correo a estudiantes de los cursos de Ingeniería de Software I e Ingeniería de Software II. De estas personas, la gran mayoría usó por última vez el sistema en el semestre Otoño 2020, según muestra la tabla 3.2 y los cursos en los que han respondido coevaluaciones se pueden observar en la tabla 3.3. La lista completa de pregunta se encuentra en el Apéndice A, pero a continuación se detallarán algunas de las preguntas.

| | Otoño 2017 | Primavera 2017 | Otoño 2019 | Primavera 2019 | Otoño 2020 |
|---|---|---|---|---|---|
| # Alumnos | 1 | 1 | 3 | 4 | 22 |

Tabla 3.2: Resultados pregunta: "¿Cuándo usó el Sistema de Coevaluación por última vez?"

| | Ingeniería de Software I | Ingeniería de Software II | Proyecto de Software |
|---|---|---|---|
| # Alumnos | 30 | 21 | 13 |

Tabla 3.3: Resultados pregunta: "¿En qué cursos ocupó el Sistema de Coevaluación?"

A través de la pregunta "Los resultados de una coevaluación son agrupados en dimensiones

por el sistema, como por ejemplo actitud, colaboración y comunicación. ¿Ha notado esta división en el sistema?" con respuestas "Sí" y "No", se estableció que será necesario aumentar la visibilización del concepto de dimensión en una coevaluación, ya que solo el 30 % de los encuestados dice conocerlo. Con respecto al sistema en general, los aspectos primordiales a mejorar serían el diseño y el seguimiento de avances, manteniendo la facilidad de uso del sistema.

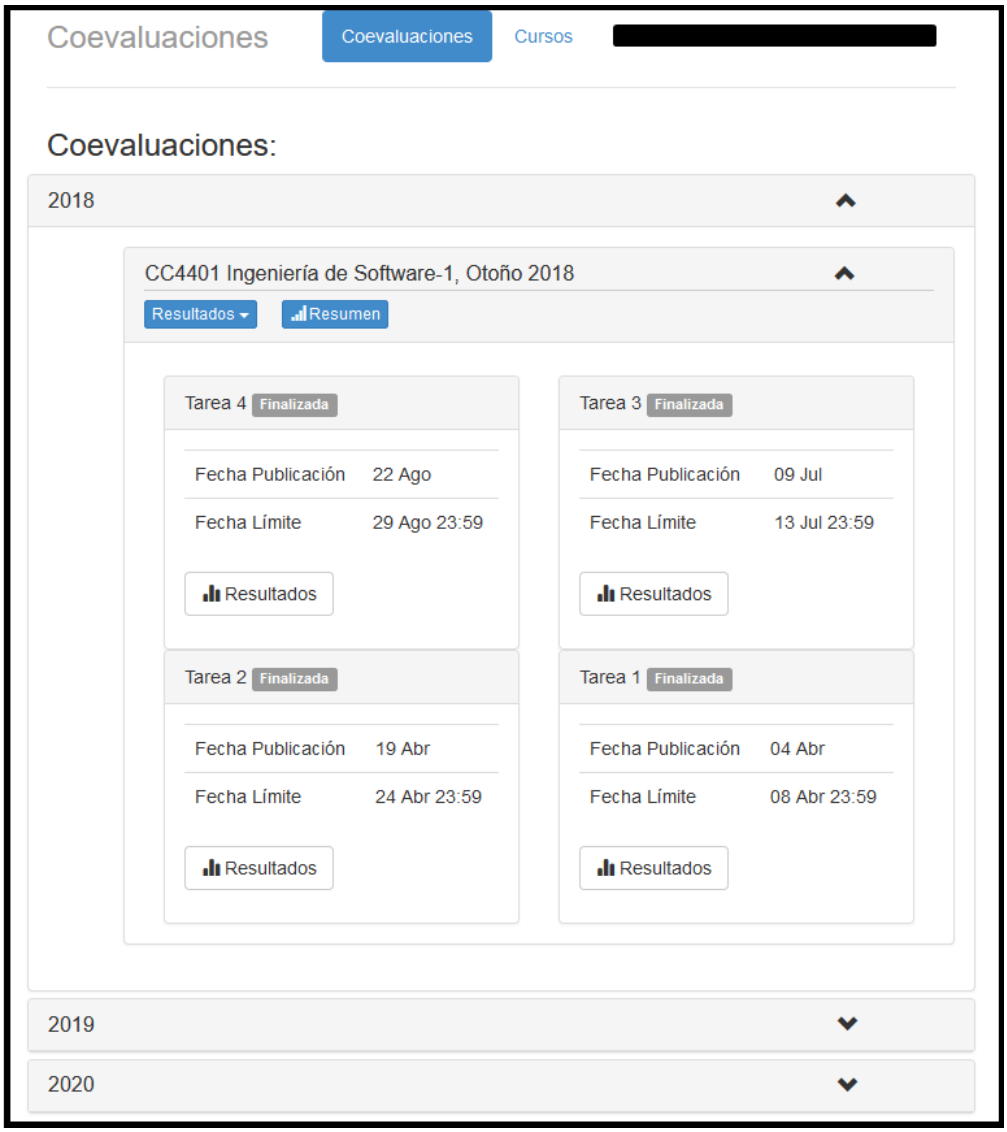



Figura 3.6: Home page sistema en producción

Luego se pidió una evaluación a la página de inicio del sistema, considerando la figura 3.6 como referencia, con respecto a qué tan comprensible, amigable, atractiva y útil era. Aquí, los encuestados debían responder en una escala de 1 a 5, considerando un 1 como "Completamente insatisfecho" y un 5 como "Completamente satisfecho". Los resultados se muestran en el gráfico de barra de la figura 3.7, donde el eje y corresponde a la cantidad de respuestas, el eje x es el tópico evaluado y el puntaje asignado se representa con colores (1: azul, 2: rojo, 3: naranjo, 4: verde, 5: morado). Se puede observar que la página de inicio se destaca en los aspectos de comprensibilidad y utilidad, pero con una clara deficiencia en atractivo, lo cual queda

también plasmado en los comentarios positivos y negativos entregados por los estudiantes en las preguntas abiertas "¿Qué te gustó de la interfaz anterior? " y "¿Qué NO te gustó de la interfaz anterior?".

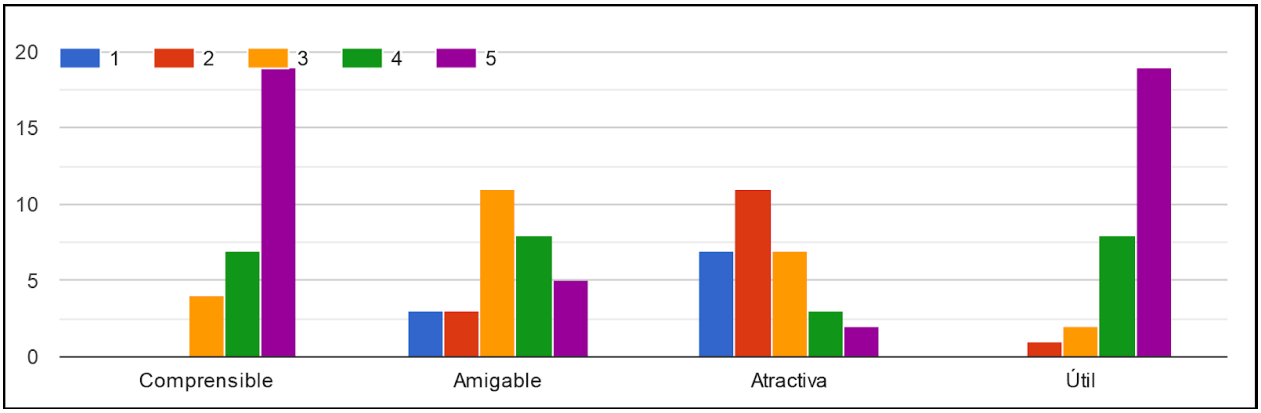



Figura 3.7: Resultados encuesta home page, sistema en producción

Al consultar la apreciación por la vista para resultados de la figura 3.8 con respecto a los mismos tópicos antes nombrados (comprensible, amigable, atractiva y útil) se obtienen resultados parecidos a los anteriores como muestra la figura 3.9, en los que se destaca en comprensibilidad y utilidad. Además de esto, al consultar "¿Le gustaría poder ver todas sus notas de un ramo en particular en una sola interfaz?" un 93.3 % responde afirmativamente, mientras que esta cifra baja a 63 % al preguntar "¿Le gustaría poder ver las notas de todos sus ramos (Ingeniería de Software I, Ingeniería de Software II y Proyecto de Software) en una sola interfaz?".

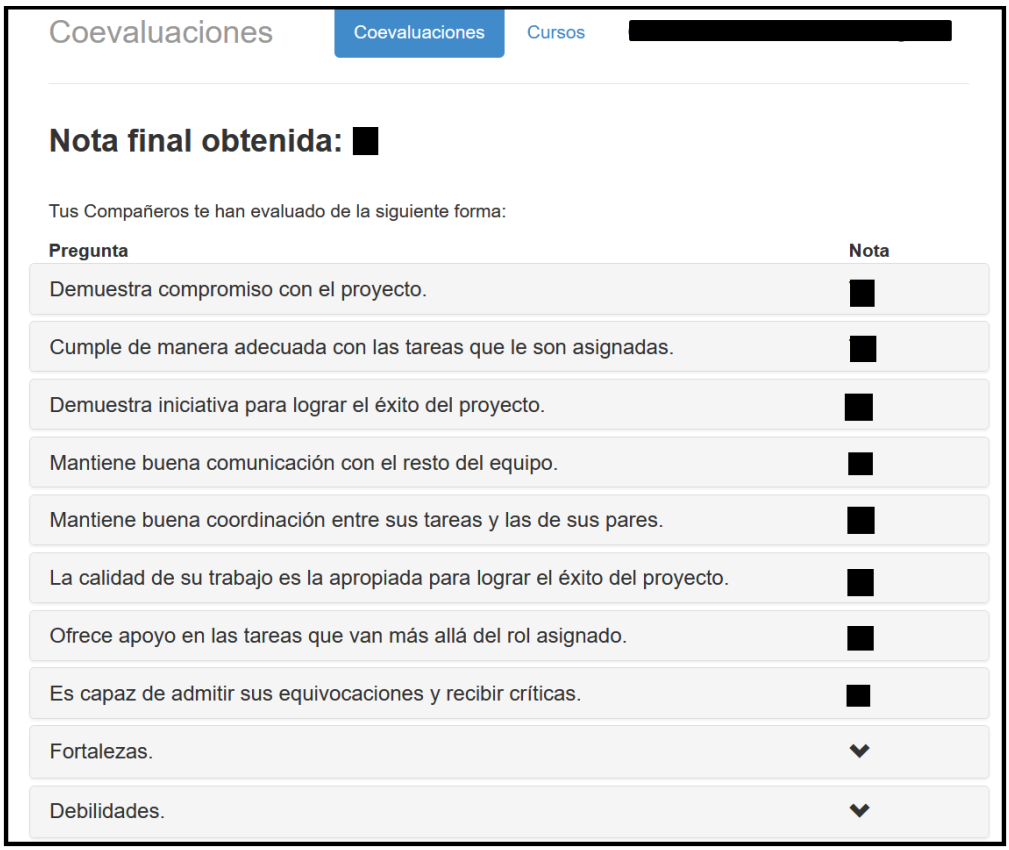



Figura 3.8: Resultados estudiante

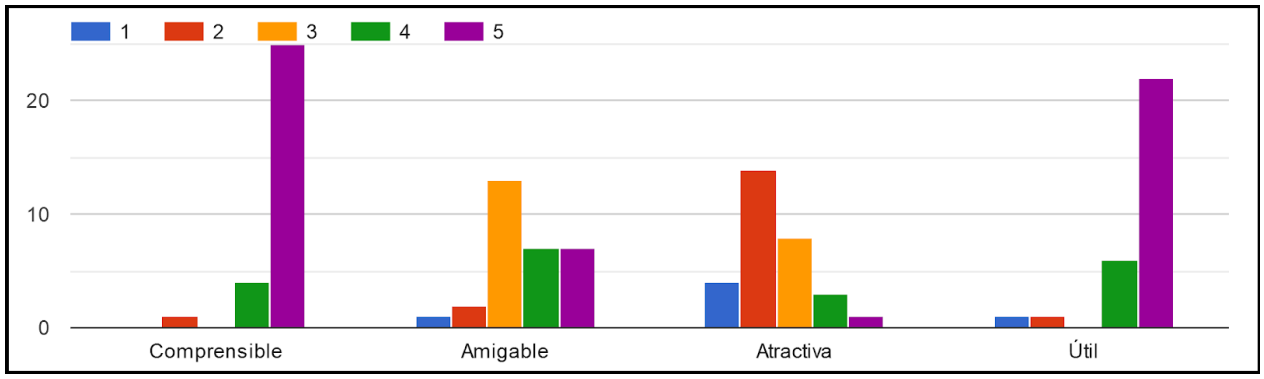



Figura 3.9: Resultados encuesta vista de resultados estudiante, sistema en producción

Con respecto a la vista del resumen de notas ocurre algo interesante, ya que al consultar "¿Alguna vez ha visto el resumen de sus notas como el presentado en la figura 3.10?" la mayoría de los encuestados (63.3 %) dice no haber visto la interfaz. Luego, al preguntar "¿Por qué no revisó el resumen de notas que entrega el sistema de coevaluaciones?" con las opciones "No estaba interesado", "No conocía la funcionalidad" y/u "Otro" (pudiendo elegir más de una opción a la vez), todos contestaron que fue debido al desconocimiento de la funcionalidad, lo cual sumado a las respuestas positivas del párrafo anterior a poder ver todas las notas en una sola interfaz hace pensar en la necesidad de aumentar la visibilidad de esta función.

Por otro lado, para la pregunta de apreciación de la vista con respecto a comprensibilidad, amigabilidad, atractivo y utilidad, la percepción general de la vista resulta ser menos clara (figura 3.11), pero ante la pregunta "Con respecto a los gráficos del resumen de notas, ¿entiende lo que representan?" un 53.3 % de los encuestados reconoce no entender los gráficos y con la pregunta "Con respecto a los gráficos del resumen de notas, ¿los considera adecuados?" un 73.3 % apela por un cambio en ellos, específicamente expresando en la pregunta abierta "¿Algún comentario adicional sobre la interfaz de resumen de notas?" problemas en los tipos de gráficos, el tamaño de estos y la necesidad de leyendas más claras.

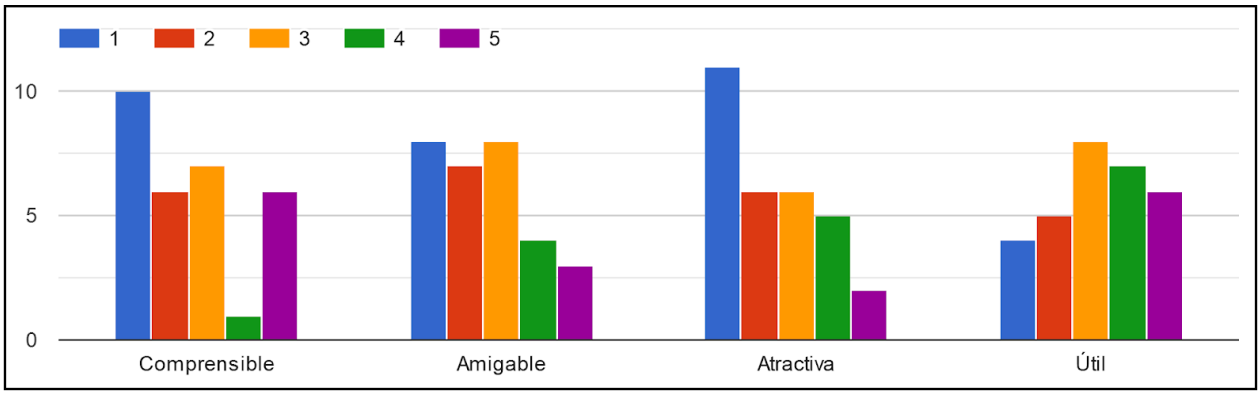



Figura 3.11: Resultados encuesta vista de resumen de notas, sistema en producción

### 3.2.2. Análisis del sistema del trabajo dirigido

El sistema desarrollado por Galvez, López y Tomic, de ahora en adelante "sistema del trabajo dirigido" o "trabajo dirigido", es un proyecto en *Django* llamado "SistemaCoevaluaciones" con un tamaño de 8.11 MB y una aplicación "coevaluaciones". Cuenta con un script en *Python* para poblar inicialmente la base de datos y documentación del trabajo realizado.

Para el análisis, se comenzó con el modelo de datos, asegurándose que pudiera soportar las funcionalidades de las que dispondrá el sistema. El sistema del trabajo dirigido maneja el modelo de datos de la figura 3.12, en donde se han excluido los atributos de las entidades para lograr mejor visibilidad[1]. En él, se presentan entidades como `Coevaluación`, que define la coevaluación que se realizará a un `curso`, las preguntas, fechas de inicio y término y un indicador de si es autoevaluación; `Encuesta`, que es la evaluación respondida por un `integrante` a un miembro de su `Grupo`, conteniendo la coevaluación a la que pertenece, el grupo y la nota calculada; `Integrante`, que representa a cualquier integrante de un `Curso` y que se distingue según su `Rol` de profesor o estudiante; y `Pregunta` que representa una pregunta de la coevaluación junto a su dimensión `DimensionPregunta`, con su tipo de pregunta almacenada en `TipoPregunta`, pudiendo ser tipo puntaje o texto.

[1]La versión extendida del modelo se presentará en la sección 4

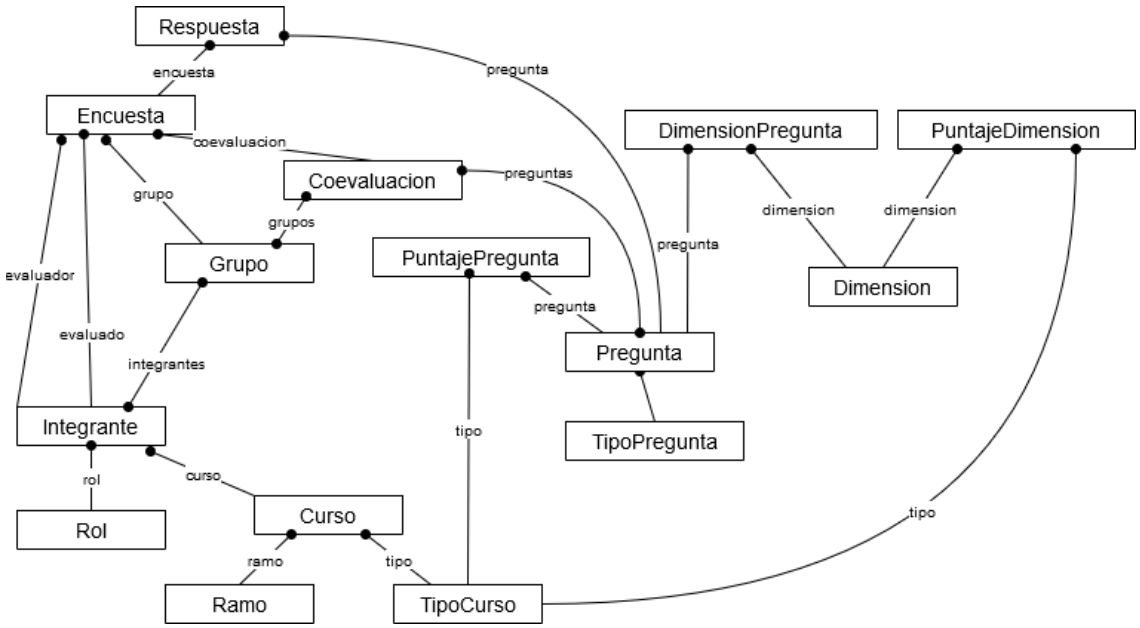



Figura 3.12: Modelo de datos simplificado.

Luego de comprobar que este modelo cumplía las funcionalidades presentes en el sistema en producción (columna SP de la tabla 3.1), se analizó si también podía soportar las nuevas funcionalidades, específicamente ver resultados mediante dashboard, la posibilidad de cambiar a alumnos de equipos durante un curso y mantener registro de los equipos por los que ha pasado un alumno. El resultado del análisis fue positivo, pero estas dos últimas funcionalidades requirieron mayor exploración del modelo.

El modelo de datos actual cuenta con una relación *Many-To-Many* entre las entidades `Integrante` y `Equipo`, según muestra la figura 3.13 y el código 3.1, con la idea de que cada vez que un estudiante se agregue a un grupo, no se borre su registro anterior, de tal forma que el grupo actual sea el último grupo de la lista de grupos a los que pertenece, mientras que todo el resto son grupos pasados.

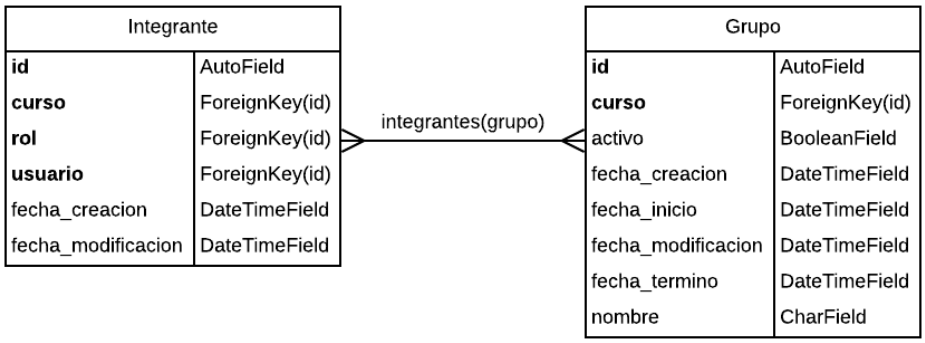



Figura 3.13: Relación *Many-To-Many* Integrante y Grupo.

Esto falla en el caso en que un alumno vuelva a un grupo en el que estuvo antes, lo cual se explica con un ejemplo:

Si existiera el caso en que un integrante X del Grupo 1 es cambiado al Grupo 2, X es agregado al Grupo 2 sin necesidad de salirse del anterior, obteniendo su grupo actual mediante el llamado al último grupo al que pertenezca. Ahora bien, si este integrante llegase a volver al Grupo 1, y se llama al último grupo al que pertenece, el modelo entregaría Grupo 2, debido a que el Grupo 1 ya existe dentro de la lista de grupos del integrante y no puede volver a agregarse, por lo que no hay un registro temporal de cuando ocurrió esto. Se podría pensar que una solución a esto sería eliminar al integrante X del Grupo 1 y volver a unirlo, pero esto haría que se perdiera el registro de su primera vez en el Grupo 1.

```
class Integrante(models.Model):
    curso = models.ForeignKey(Curso, on_delete=models.CASCADE)
    rol = models.ForeignKey(Rol, on_delete=models.CASCADE)
    usuario = models.ForeignKey(settings.AUTH_USER_MODEL, on_delete=models
    .CASCADE)
    fecha_creacion = models.DateTimeField(auto_now_add=True)
    fecha_modificacion = models.DateTimeField(auto_now=True)

    def __str__(self):
        return '{} - {} - {}'.format(self.usuario.first_name, self.rol,
    self.curso)

    class Meta:
        unique_together = ('curso', 'usuario')

class Grupo(models.Model):
    curso = models.ForeignKey(Curso, on_delete=models.CASCADE)
    integrantes = models.ManyToManyField(Integrante)
    fecha_inicio = models.DateTimeField(auto_now_add=True)
    fecha_termino = models.DateTimeField(blank=True, null=True, default=
    None)
    activo = models.BooleanField(default=True)
    nombre = models.CharField(max_length=100)
    fecha_creacion = models.DateTimeField(auto_now_add=True)
    fecha_modificacion = models.DateTimeField(auto_now=True)

    def __str__(self):
        return '{}'.format(self.nombre)

    class Meta:
        unique_together = ('curso', 'activo', 'nombre')
```

Código 3.1: Modelado Integrante y Grupo

Se analizaron varias posibles soluciones, tales como mantener un historial del campo *integrantes* de Grupo, usando las librerías django-field-history [14] y django-simple-history [37], pero ninguna de las dos soportaba campos de tipo *Many-To-Many*. Luego, se pensó en cambiar el modelo de datos de dos formas: la primera opción representada por la figura 3.14 sugería cambiar a una relación *One-To-Many*, agregando un campo para determinar si la instancia de Integrante estaba asociada al grupo actual; la segunda opción, como se muestra en la figura 3.15 era mantener la relación *Many-To-Many* para guardar los grupos anteriores

del integrante y agregar una relación *One-To-Many* para guardar el grupo actual. Después, se investigó usar las *signals* de Django[17], ocupando *m2m_changed* para alterar el comportamiento en la acción *pre_add*[2], levantando un error en caso de que se quisiera agregar un integrante a un grupo distinto pero que fuera del mismo semestre a alguno de los que ya pertenecía, para así obligar al usuario a antes de cambiar al integrante de grupo, borrar su grupo anterior, como se muestra en el código 3.2, pero esto producía el problema de que se perdía el registro de los grupos anteriores de un mismo semestre.

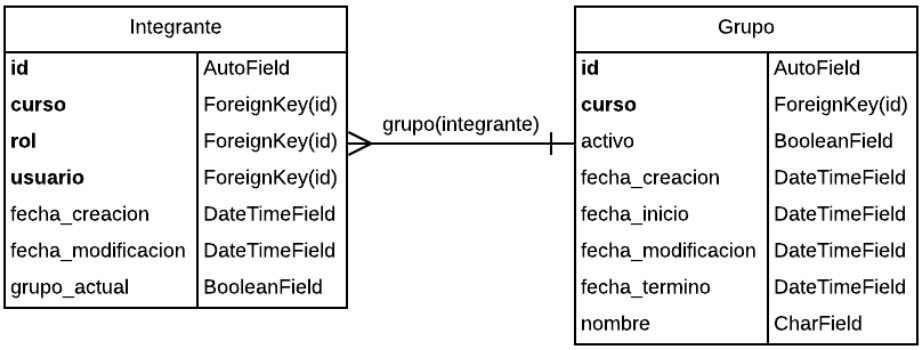



Figura 3.14: Relación *One-To-Many* Integrante y Grupo.

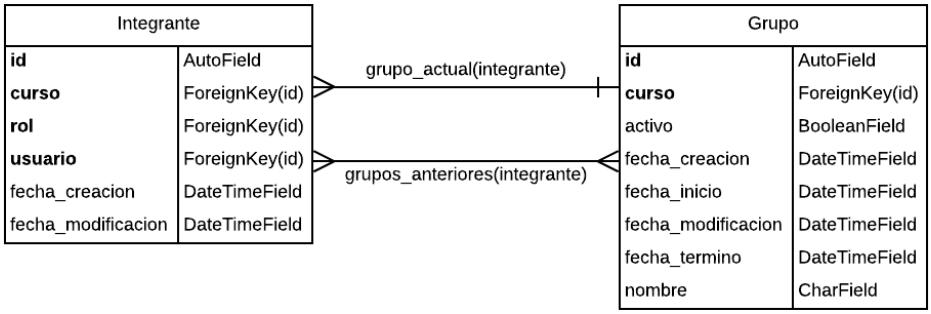



Figura 3.15: Relaciones *One-To-Many* y *Many-To-Many* Integrante y Grupo.

```
@receiver(m2m_changed, sender=Grupo.integrantes.through)
def verify_uniqueness(sender, **kwargs):
    grupo = kwargs.get('instance', None)
    action = kwargs.get('action', None)
    integrantes = kwargs.get('pk_set', None)

    if action == 'pre_add':
        for integrante in integrantes:
            if Grupo.objects.filter(curso=grupo.curso).filter(integrantes=
   integrante):
```

[2] *m2m_changed* es la señal que se envía cuando un campo del tipo *Many-To-Many*, específicamente cuando se ocupa la acción *pre_add*, es decir, se realiza antes de agregar un elemento al campo integrantes de Grupo.

```
10                raise IntegrityError('Grupo with curso %s already exists
    for integrante %s' % (grupo.curso, integrante))
```

Código 3.2: Modificación signal *m2m_changed*, acción *pre_add*

Finalmente, se consideró que la mejor opción era volver tener una relación *One-To-Many* entre el Integrante y el Grupo. Con esto, el integrante solo podría pertenecer a su equipo actual, y para encontrar su historial de pertenencia a grupos pasados, se decidió ver todas las instancias de la clase `Encuesta`, considerando como grupos anteriores aquellos en que el integrante aparece como evaluador o evaluado y en que el parámetro `grupo` no coincide al que pertenece actualmente.

Una vez terminado este análisis, se pasó a investigar las vistas actuales del sistema, definiendo puntos en que se podría mejorar y encontrando bugs que necesitan ser arreglados durante la realización de esta memoria. En el Apéndice B se encuentra la lista completa de estos temas, mas los principales detalles encontrados, los que fueron:

- Error al iniciar sesión con estudiante sin coevaluaciones.
- Una vez cerrada una sesión, si se vuelve a la página anterior se puede ver la sesión como abierta e incluso realizar acciones.
- La función agregar curso desde profesor no funciona.
- Existe un error al calcular la nota de los estudiantes, ya que se intentan sumar valores de tipo decimal y float.
- Hay que restringir los alumnos y las coevaluaciones que pueden asignarse a un grupo, debiendo ser pertenecer al mismo curso al que pertenece el grupo.

## 3.3. Requisitos de la solución

Habiendo ya entendido el problema y teniendo una noción de cómo y para qué se utilizará, es que se levantan los requisitos que el sistema debe cumplir.

**R01 -** Un estudiante debe poder ver, responder y revisar sus resultados de una coevaluación

**R02 -** Un profesor debe poder ver y gestionar los integrantes, grupos y coevaluaciones de sus cursos.

**R03 -** Un profesor debe poder crear cursos donde él sea docente.

**R04 -** Todos los usuarios deben poder administrar sus cuentas, teniendo la opción de modificar su email y contraseña.

**R05 -** Los administradores pueden gestionar usuarios.

**R06 -** Los profesores deben tener la opción de descargar los resultados de las coevaluaciones.

**R07 -** Los profesores deben poder ver el estado de las coevaluaciones y la cantidad de alumnos pendientes por responderlas, pudiendo enviar una alerta.

**R08 -** Los profesores deben poder ver los resultados de todos los integrantes del sus cursos con el detalle de la asignación de notas.

**R09 -** Profesores y alumnos deben poder hacer seguimiento de desempeño en sus cursos.

**R10 -** Deben poder crearse autoevaluaciones.

**R11 -** Los auxiliares deben poder crear coevaluaciones y ver los resultados de estas.

**R12 -** Los auxiliares deben poder ver los cursos en los que son alumnos.

Estos requisitos fueron recabados durante las reuniones iniciales con docentes de los cursos del área de ingeniería de software. La prioridad de estos requisitos es alta, y todos ellos, menos los requisitos R11 y R12, son necesarios para considerar que la solución ha sido exitosa. Se excluyen los requisitos relacionados a auxiliares ya que esta versión del sistema está enfocada en alumnos y profesores.

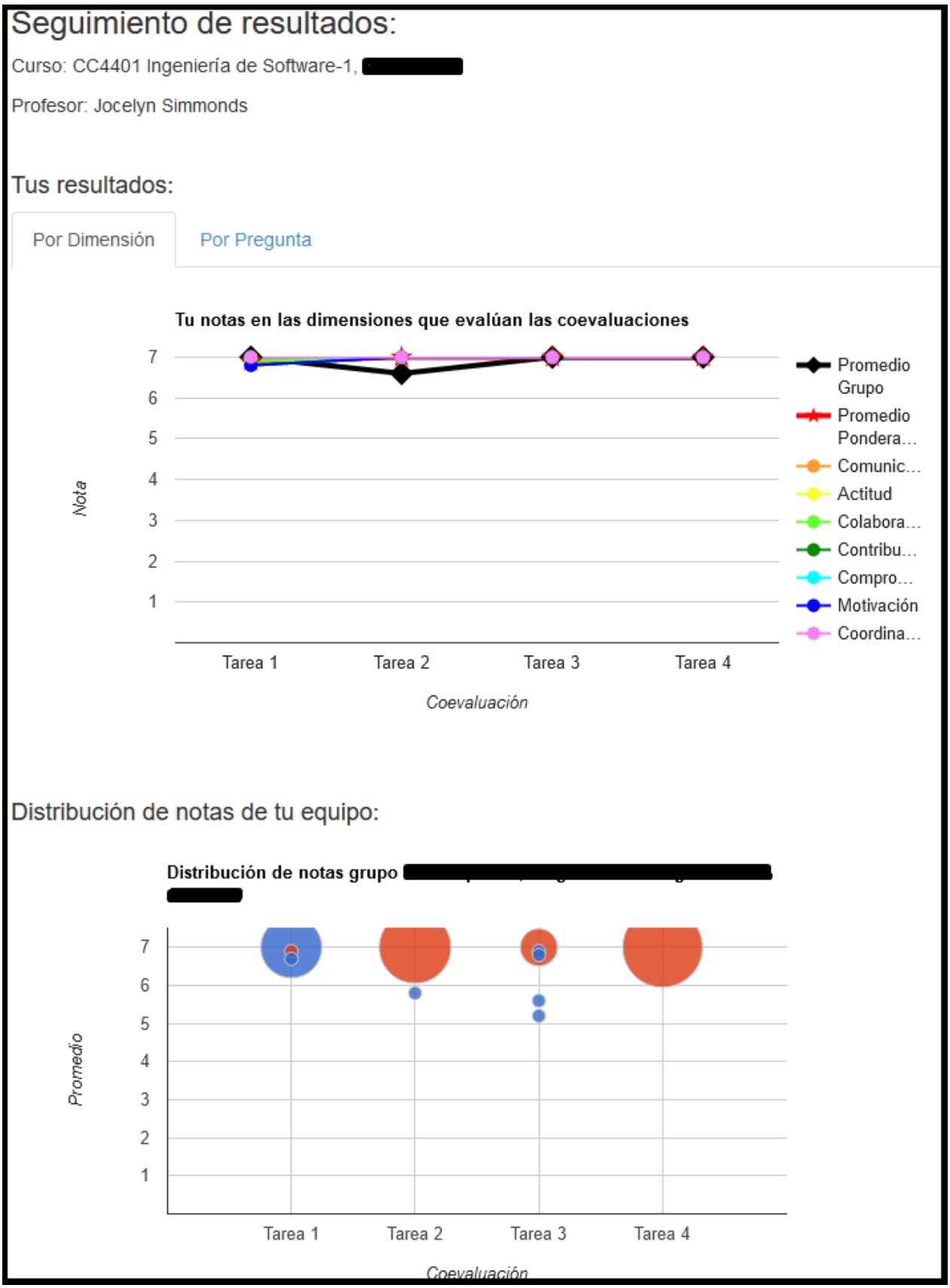



Figura 3.10: Resumen de notas, sistema en producción

# Capítulo 4

# Diseño de la Solución

En este capítulo, se hablará sobre las arquitecturas física y lógica del sistema, el modelo de datos usado considerando las modificaciones antes mencionadas en el capítulo de Concepción de la Solución. Finalmente, se mostrará el diseño del sistema en término de componentes generales y mockups.

## 4.1. Arquitectura y modelo de datos

La arquitectura física (figura 4.1) utiliza un motor de base de datos PostgreSQL, montando el servidor sobre Django 3.1 y un ambiente virtual en Python 3.8.5. Esta arquitectura fue heredada desde el trabajo dirigido, y se decidió mantenerla ya que PostgreSQL es un sistema de bases de datos relacional, *open-source* y de alta disponibilidad (capaz de garantizar continuidad de servicios), la cual es soportada por Django, *framework* escrito en Python y enseñado durante la carrera de Ingeniería Civil en Computación en la FCFM, haciendo que el código sea de fácil adaptación para futuras generaciones que pudieran tener que mantener o actualizar este sistema, contando con funciones prediseñadas tales como el manejo de sesiones, contraseñas y la administración de la información, presente en el sistema a través de un portal.

Además de lo anterior, debido a que Python es un lenguaje de computación creado hace 30 años, posicionado como el cuarto lenguaje de programación más usado según la encuesta anual de Stack Overflow para el año 2020[16], que sigue siendo mantenido y actualizado, y del cual se han publicado diferentes librerías para áreas tan contingentes como Ciencia de Datos, Inteligencia Artificial y Big Data, le entrega fiabilidad y flexibilidad para el futuro.

A esto se le puede agrergar el hecho de que esta arquitectura era igual a la ya utilizada en el sistema en producción (salvo la actualización de versiones), la cual ha funcionado con éxito durante los 6 años que lleva en uso.

La arquitectura lógica se presenta en la figura 4.2. Siguiendo el patrón de arquitectura descrito en la sección 2.1 usado por Django, es que podemos ver los distintos módulos dentro de los controladores. Los módulos de gestión de alumnos, equipos, profesores, coevaluaciones y cursos cuentan en ellos con funciones para la vista, creación, edición y eliminación de


Figura 4.1: Arquitectura física del sistema

entidades.

Por su parte, el módulo de dashboard está integrado por la creación de gráficos radar para notas de preguntas y puntajes de dimensiones. Este módulo interactúa con el módulo de seguimiento de desempeño, el cual muestra la progresión de las calificaciones de alumnos y grupos en el tiempo mediante el uso de gráficos y tablas comparativas.

Notificaciones es el módulo que se encarga del envío de correos a los estudiantes, esto al ser ingresados al sistema por primera vez (con el envío de sus usuarios y contraseñas) y luego el envío de notificaciones para toma de acción a aquellos integrantes que estén pendientes para contestar una coevaluación.

El modelo de base de datos se desarrolló según muestra la figura 4.3. Tal como se mencionó en el capítulo anterior, este modelo se hereda desde el trabajo dirigido, contando con 15 clases en total más el modelo User creado por Django. En él, un `Curso` va asociado a un `Ramo` (entidad general para asignar código y nombre de la asignatura) y a un `TipoCurso` (con o sin horas fijas). Se podría pensar en el `Curso` como un detalle de un `Ramo`, en donde se da mayor granularidad a este, describiendo el año, sección y semestre en que se dicta.

Cada `Curso` está constituido por integrantes (clase `Integrante`), grupos (`Grupo`) y coevaluaciones (`Coevaluacion`). Cada `Integrante` tiene un `Rol` dentro del curso, el cual puede ser *"Alumno", "Auxiliar''* o *"Profesor"*. Un `Grupo` está conformado por integrantes, y puede estar activo o no. Cada integrante solo puede pertenecer a un grupo dentro del mismo curso.

Por su parte, una `Coevaluacion` tiene encuestas (`Encuesta`) y preguntas (`Pregunta`), especificando además propiedades como su nombre, fechas de inicio y término, si corresponde a la vez a una autoevaluación y el estado en el que se encuentra. Existen 4 posibles estados para una coevaluación: *Creada (C), Publicada (P), Finalizada (F)* y *Respuestas publicadas (RP)*, los cuales vienen desde el diagrama de flujo de la sección 3.2. Las encuestas relacionadas a una coevaluación se explican como la instancia de respuesta, en palabras más simples, en una evaluación física sería la hoja de papel con los datos del evaluador, evaluado, grupo al que pertenecen ambos y la nota promedio que se asignó, mientras que las respuestas (`Respuesta`) asociadas a una encuesta serían el detalle de la evaluación, con los puntajes o comentarios asociados a las preguntas. Las respuestas también poseen una relación directa hacia las preguntas.

Una coevaluación de este sistema está asociada a las 10 preguntas mencionadas en el capítulo 2.5.3 de "Estandarización de la coevaluación". Estas preguntas tienen un `TipoPregunta`

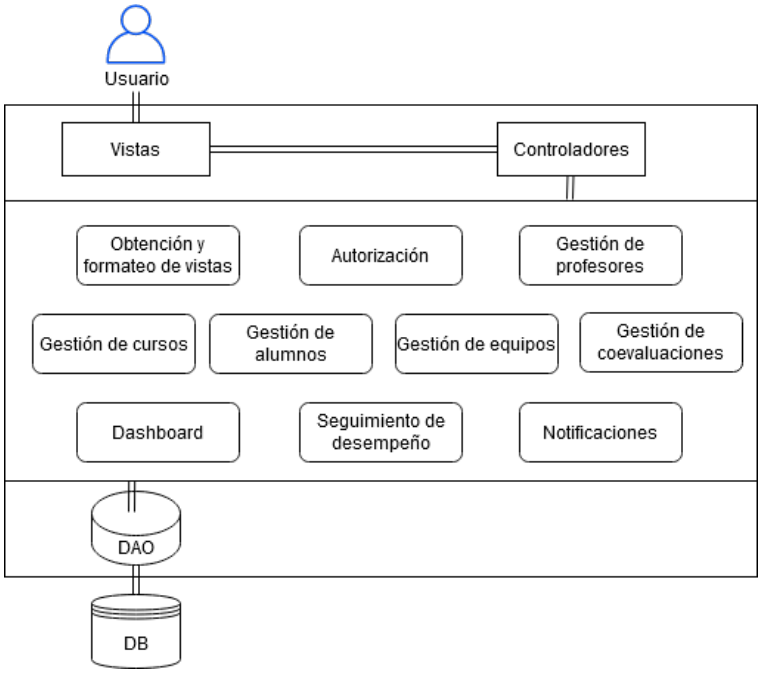
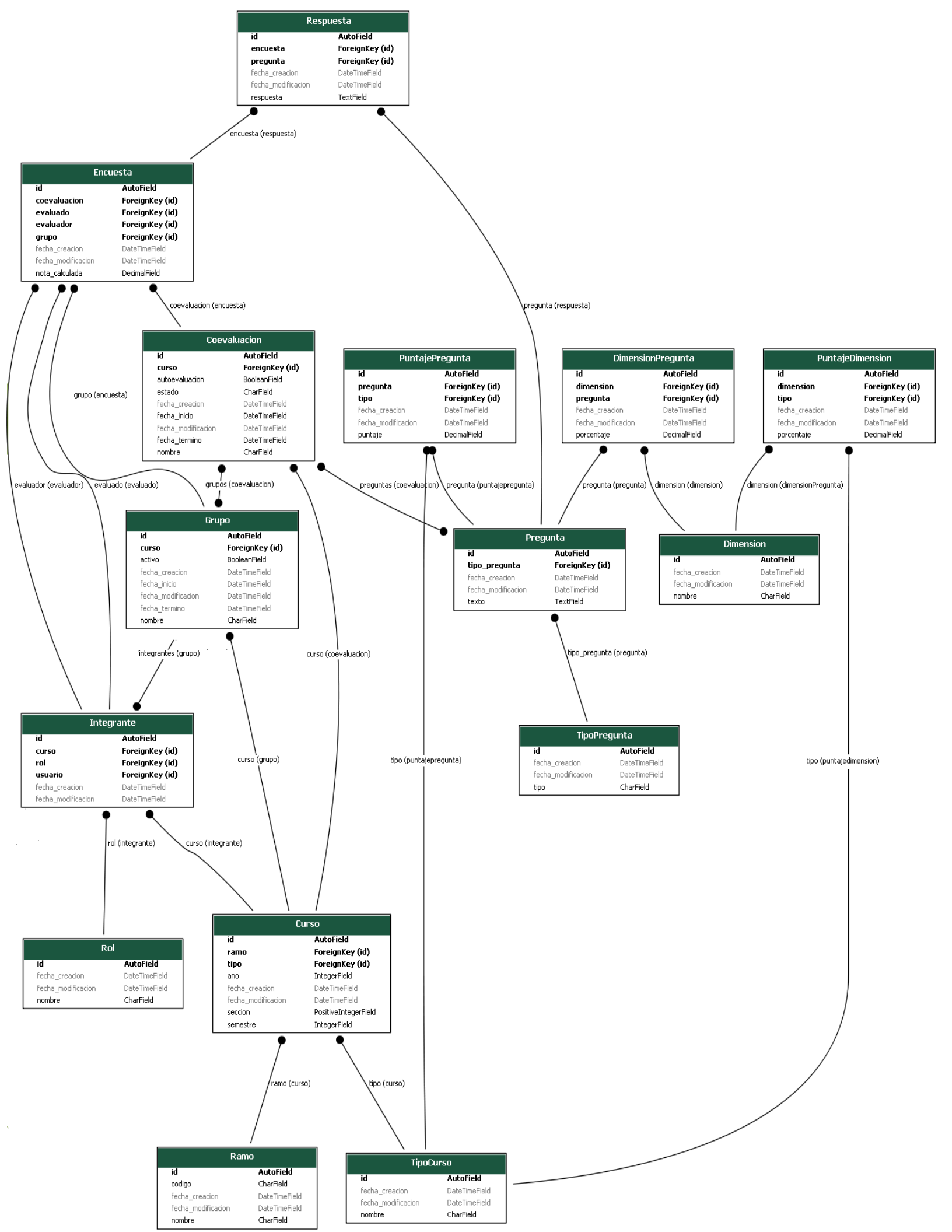



Figura 4.2: Arquitectura lógica del sistema

para describir si son de tipo *puntaje* o *texto*. Cada pregunta tiene sus puntajes asociados (`PuntajePregunta`), debiendo tener 2 de ellos, cada uno correspondiente a cursos con y sin horas fijas. De la misma forma, el curso puede tener 0 más relaciones hacia una dimensión (`DimensionPregunta`), siendo 0 solo en los casos de preguntas de tipo *texto*. Las dimensiones aquí presentes se obtienen desde la clase `Dimension` y son las mismas 7 dimensiones mencionadas en el estudio de Silvestre (sección 2.5.3). A ellas van también asociados puntajes (`PuntajeDimension`), los que al igual que `PuntajePregunta`, deben ser 2 asociaciones por dimensión, una para cada tipo de curso.

Figura 4.3: Modelo de base de datos del sistema


Respuesta
id AutoField
encuesta ForeignKey (id)
pregunta ForeignKey (id)
fecha_creacion DateTimeField
fecha_modificacion DateTimeField
respuesta TextField
encuesta (respuesta)
pregunta (respuesta)
Encuesta
id AutoField
coevaluacion ForeignKey (id)
evaluado ForeignKey (id)
evaluador ForeignKey (id)
grupo ForeignKey (id)
fecha_creacion DateTimeField
fecha_modificacion DateTimeField
nota_calculada DecimalField
coevaluacion (encuesta)
Coevaluacion
id AutoField
curso ForeignKey (id)
autoevaluacion BooleanField
estado CharField
fecha_creacion DateTimeField
fecha_inicio DateTimeField
fecha_modificacion DateTimeField
fecha_termino DateTimeField
nombre CharField
grupo (encuesta)
PuntajePregunta
id AutoField
pregunta ForeignKey (id)
tipo ForeignKey (id)
fecha_creacion DateTimeField
fecha_modificacion DateTimeField
puntaje DecimalField
DimensionPregunta
id AutoField
dimension ForeignKey (id)
pregunta ForeignKey (id)
fecha_creacion DateTimeField
fecha_modificacion DateTimeField
porcentaje DecimalField
PuntajeDimension
id AutoField
dimension ForeignKey (id)
tipo ForeignKey (id)
fecha_creacion DateTimeField
fecha_modificacion DateTimeField
porcentaje DecimalField
evaluador (evaluador)
evaluado (evaluado)
grupos (coevaluacion)
preguntas (coevaluacion)
pregunta (puntajepregunta)
pregunta (pregunta)
dimension (dimension)
dimension (dimensionPregunta)
Grupo
id AutoField
curso ForeignKey (id)
activo BooleanField
fecha_creacion DateTimeField
fecha_inicio DateTimeField
fecha_modificacion DateTimeField
fecha_termino DateTimeField
nombre CharField
Pregunta
id AutoField
tipo_pregunta ForeignKey (id)
fecha_creacion DateTimeField
fecha_modificacion DateTimeField
texto TextField
Dimension
id AutoField
fecha_creacion DateTimeField
fecha_modificacion DateTimeField
nombre CharField
integrantes (grupo)
curso (coevaluacion)
tipo_pregunta (pregunta)
Integrante
id AutoField
curso ForeignKey (id)
rol ForeignKey (id)
usuario ForeignKey (id)
fecha_creacion DateTimeField
fecha_modificacion DateTimeField
curso (grupo)
tipo (puntajepregunta)
TipoPregunta
id AutoField
fecha_creacion DateTimeField
fecha_modificacion DateTimeField
tipo CharField
tipo (puntajedimension)
rol (integrante)
curso (integrante)
Rol
id AutoField
fecha_creacion DateTimeField
fecha_modificacion DateTimeField
nombre CharField
Curso
id AutoField
ramo ForeignKey (id)
tipo ForeignKey (id)
ano IntegerField
fecha_creacion DateTimeField
fecha_modificacion DateTimeField
seccion PositiveIntegerField
semestre IntegerField
ramo (curso)
tipo (curso)
Ramo
id AutoField
codigo CharField
fecha_creacion DateTimeField
fecha_modificacion DateTimeField
nombre CharField
TipoCurso
id AutoField
fecha_creacion DateTimeField
fecha_modificacion DateTimeField
nombre CharField

En la práctica, existen reglas muy importantes que se deben seguir al emplear esta base de datos. Estas son:

1. Un integrante se considera parte de un grupo cuando ya ha contestado o le han contestado a él una evaluación. Es decir, cuando aparece una entrada en la tabla `Encuesta` con él como evaluador o evaluado.
2. Si el integrante de un curso[1] ya pertenecía a un grupo y este se reemplaza, para obtener un historial de los grupos anteriores se debe revisar las encuestas asociadas a el y buscar ahí el grupo en donde ocurrió, considerando como grupo anterior todos aquellos que sean distintos al por el que se está reemplazando.
3. Un usuario puede ser integrante de dos o más cursos distintos dentro del mismo año y semestre, pero la combinación *"Alumno"* y *"Profesor"* no está permitida. La combinación *"Auxiliar"* y *"Profesor"* sí se permite, mas no es recomendada ya que este usuario podría crear nuevos cursos a su nombre.
4. Las entradas de las clases `Rol`, `TipoCurso`, `Pregunta`, `PuntajePregunta`, `Dimension`, `DimensionPregunta` y `PuntajeDimension` son pre-cargadas a la base de datos y se sugiere no modificarlas para asegurar el correcto funcionamiento del sistema. Sin embargo, nada prohíbe que se creen nuevas instancias de estas clases.

## 4.2. Diseño

Con respecto al diseño, se decidió continuar con el proveniente del sistema del trabajo dirigido. Esto se consideró porque este diseño ya había sido aprobado por las docentes Simmonds y Bastarrica durante este proceso, las cuales serán además usuarias del sistema tema de esta memoria. A pesar de esto, el diseño de nuevas vistas sí fue sometido a validaciones mediante entrevistas con parte del cuerpo docente, a través de lo que fueron ajustadas y mejoradas para llegar a la versión final que se muestra en esta sección.

En primer lugar se tiene la paleta de colores general del sistema de acuerdo a la figura 4.4, la que maneja tonos sobrios, fríos y azulados. Luego, en la figura 4.5 se muestra la paleta particular para las coevaluaciones, parte esencial del desarrollo ya que ayuda a dar un *feedback* visual de los estados del sistema. Como se observa, para los estados se dejó el color más brillante para las coevaluaciones publicadas, instando así a la toma de acción, por otro lado, los colores para coevaluaciones finalizadas y con respuestas publicadas son parecidos a propósito, ya que se quiere expresar que ambas coinciden en que la coevaluación está cerrada, pero se usa el color más brillante en las coevaluaciones finalizadas por el mismo motivo que en las publicadas, para así destacarla del resto.


Figura 4.4: Paleta de colores general del sistema.

[1] Al decir "integrante de un curso" en realidad se está repitiendo una información dos veces, porque la entidad integrante conlleva ya la asociación a un curso, mas se usa esta expresión para hacer énfasis en que esta regla va relacionada con el curso


Figura 4.5: Paleta de colores para estados de coevaluaciones: Creada, Publicada, Finalizada y Respuestas Publicadas.

En general, la paleta de colores del sistema es opaca, tendiendo a colores azulados y grises, con ciertos matices de verdes y amarillos para indicar funciones realizadas con éxito o que requieren alguna acción extra.

En segundo lugar se tienen los componentes del sistema, los que manejan un estilo constante a lo largo del desarrollo y se repiten constantemente, simplemente cambiando la información que traen. En la figura 4.6 se puede observar el estilo de las tablas, usando un fondo gris con letras en negrita para headers y fondo blanco para las filas de datos. Al poner el mouse sobre filas con opción de clickear, el fondo pasa a ser un tono azulado y el cursor cambia para dar cuenta al usuario de esta función.

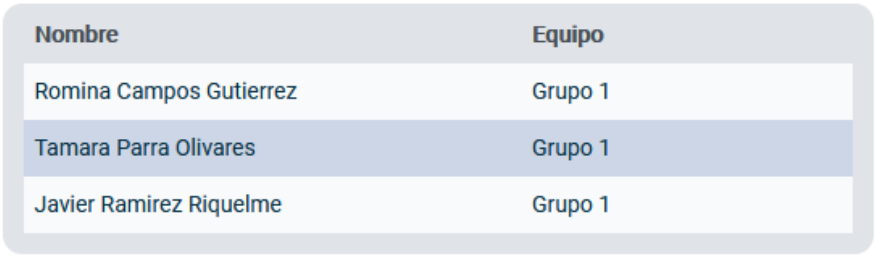



Figura 4.6: Ejemplo formato tablas.

Para tarjetas de información el formato es el observado en el ejemplo de la figura 4.7, con la etiqueta de color grisáceo y la información mucho más destacada. El ejemplo mostrado es de una vista de coevaluación para profesores, por lo que también trae dos íconos de edición y eliminación.

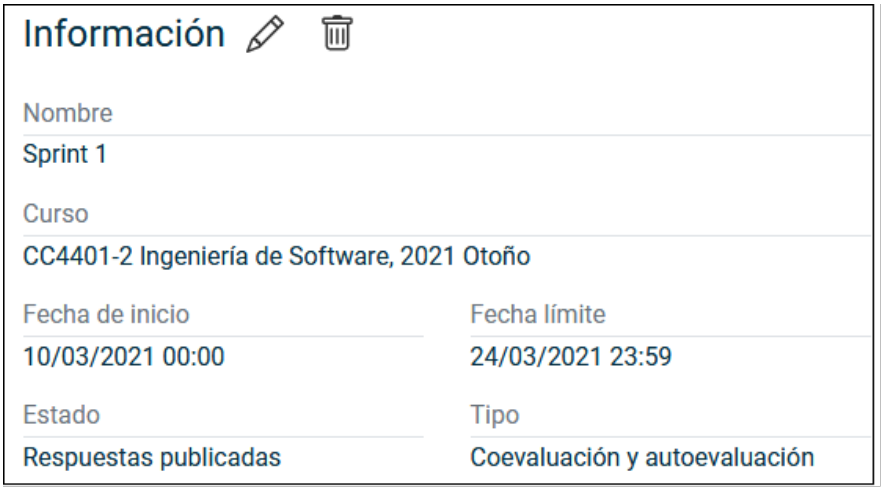



Figura 4.7: Ejemplo tarjeta de información.

Las notas presentan el formato de la figura 4.8, en donde el peso de la fuente es mayor y, en caso de ser inferior a 4.0, se cambia el color a rojo. Por su lado, las listas de equipo siguen el estilo de la figura 4.9, con un color grisáceo para resaltar el nombre del equipo pero a la vez mantener la paleta sobria del sistema.

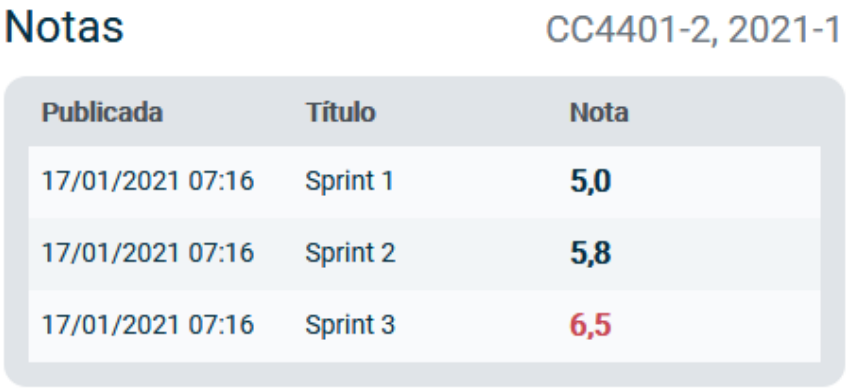



Figura 4.8: Ejemplo tabla con notas menores y mayores a 4.0.

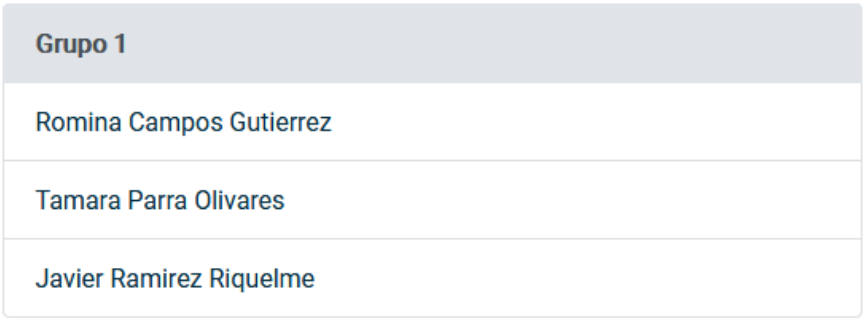



Figura 4.9: Ejemplo lista de grupo.

### 4.2.1. Mockups de vistas

Para el diseño de vistas se realizaron mockups usando la herramienta Balsamiq Wireframes. Estos diseños fueron creados antes de comenzar el trabajo, y varias de ellas sufrieron cambios durante el desarrollo de este. Los cambios realizados serán explicados en la siguiente sección cuando sea necesario. En esta sección solo se mostrarán mockups de las vistas no diseñadas en el trabajo dirigido, mas del resto solo se nombrarán los cambios realizados. En este capítulo existen referencias a figuras presentes en el Apéndice C.

#### 4.2.1.1. Vistas para rol Profesor

Lo primero que se hizo fue modificar el diseño del **home page**, agregando a la vista de la figura 2.7 el botón para ir al seguimiento de desempeño, opciones para filtrar las tablas y una columna de avance a la tabla de coevaluaciones para indicar cuantas personas faltan

por responder una coevaluación. Además, para agregar una nueva coevaluación se agregó un checkbox para indicar si sería a la vez un autoevaluación.

Luego, en la vista del **perfil del profesor** se podrá ver su información y hacer cambios en el correo y contraseña asociados a la cuenta según muestra la figura C.1 presente en el Apéndice C.

Para el **detalle de un curso específico** la vista contiene un tab para elegir entre ver los estudiantes por orden alfabético o separados por equipos, según las figuras 4.14 y C.3, además de tarjetas para representar cada una de las coevaluaciones. Cuando un alumno haya sido cambiado de equipo, su equipo anterior se mostrará con con una fuente de color grisáceo en la lista por orden alfabético. Existen también botones para editar los integrantes, equipos y coevaluaciones, y si se clickea sobre un estudiante o una coevaluación, se redireccionará al detalle de este.

El **detalle de un estudiante** se tiene en la figura C.4 del Apéndice C, donde existe un historial de coevaluaciones con la nota general obtenida y la opción de ver el detalle de ellas clickeando la columna "Detalle", lo que muestra el puntaje obtenido según pregunta y dimensión, y el *feedback* recibido por sus compañeros, según se ve en la figura 4.10.

En vista de **edición de integrantes** de la figura 4.11 se pueden agregar, editar y eliminar estudiantes, a través de la importación de una planilla o manualmente al apretar el botón "Agregar nuevo". El equipo será opcional al crear un estudiante.

En la figura 4.12 se muestra la vista de **edición de equipos**. Aquí, los equipos se crean usando el botón "Agregar nuevo", lo que abre un pop up como el de la figura 4.13, mientras que su edición y eliminación se hace con los iconos del lado derecho de sus respectivas tarjetas. Los estudiantes son agregados o cambiados de grupo a través de *drag and drop*, manteniendo a la vez una lista de aquellos alumnos que no poseen equipo aún.

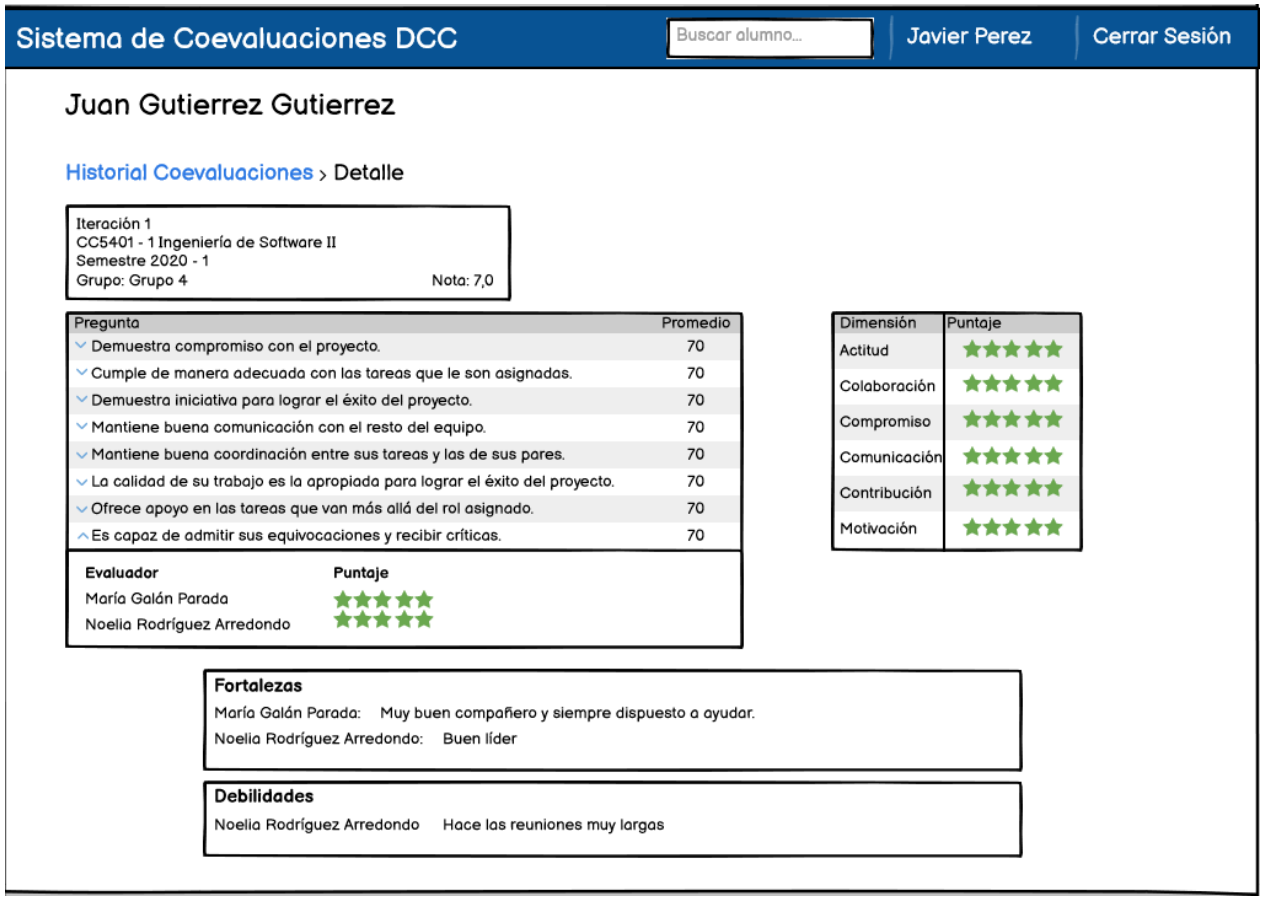



Figura 4.10: Detalle de una coevaluación específica de un alumno

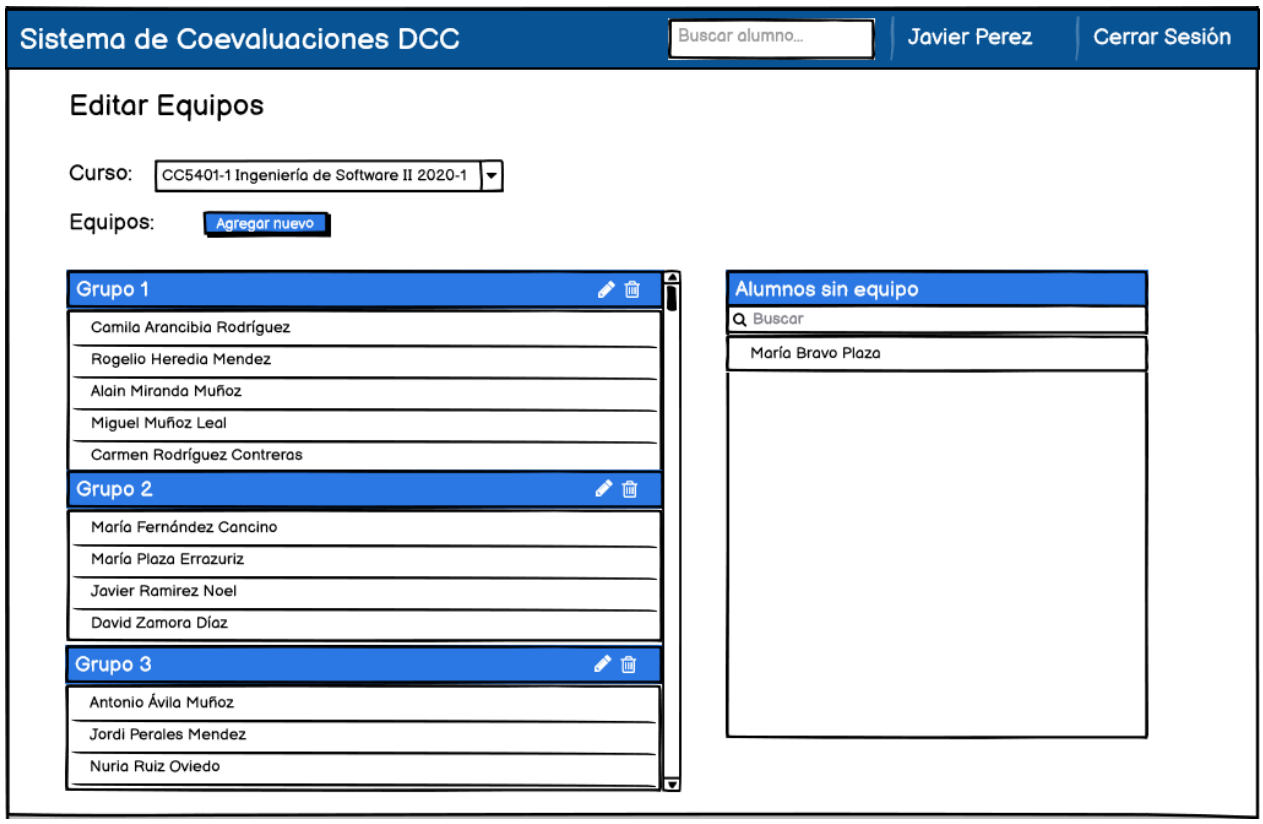



Figura 4.12: Editar equipo

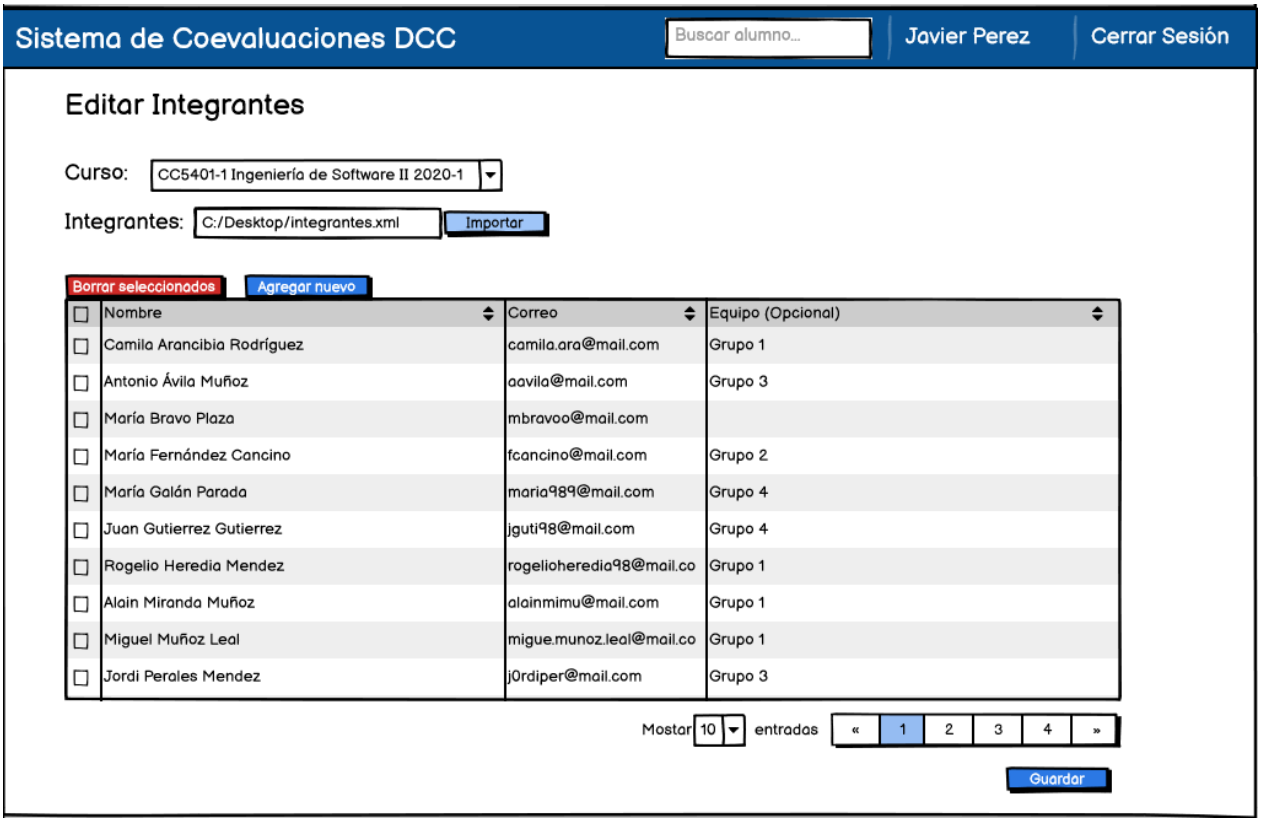



Figura 4.11: Edición de integrantes cargando plantilla

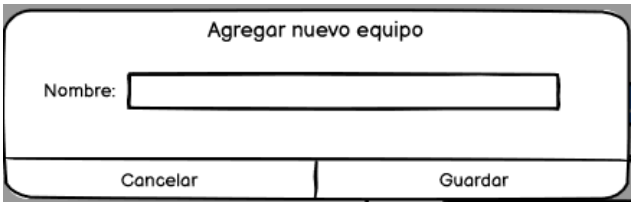



Figura 4.13: Editar equipo - Crear nuevo

El **seguimiento de desempeño** se puede hacer de dos formas: filtrando por alumno (figura 4.15) o por equipo (figura C.2 del Apéndice C). El filtro por alumno cuenta con una vista de todas las notas que ha obtenido el alumno durante el curso, un dashboard para mostrar gráficamente los resultados por pregunta y dimensión, y el detalle de estos resultados, mientras que el filtro por equipo muestra el resumen por cada uno de los estudiantes que pertenece a él, un cuadro que muestra los estudiantes que fueron agregados al equipo durante el semestre y gráficos separados por dimensión/pregunta y coevaluación, de tal forma de lograr comparar el desempeño de todos los integrantes del equipo entre ellos y entre distintas coevaluaciones.

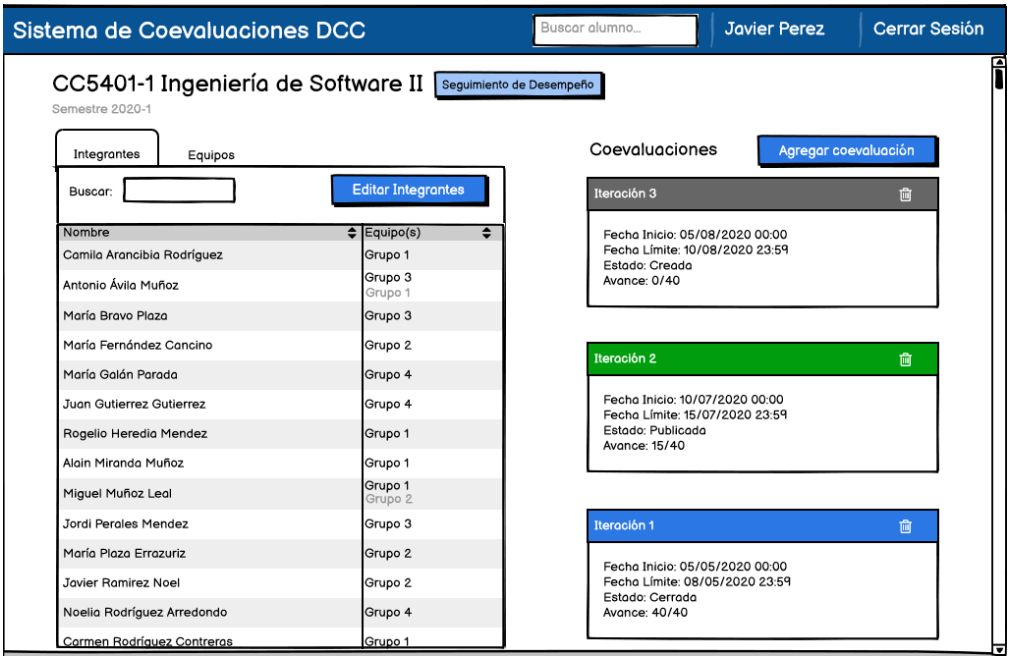



Figura 4.14: Detalle curso - Lista integrantes

#### 4.2.1.2. Vistas para rol Alumno

En primer lugar está el diseño para el **home page**, donde se omite la tabla de los cursos por los que ha pasado el estudiante y se queda solamente con la de sus coevaluaciones asignadas. Además tendrá a su disposición un botón que lo llevará a la página de seguimiento de desempeño.

El **perfil del estudiante** es igual al perfil del profesor, con las funcionalidades de cambiar correo y contraseña. De la misma forma, la vista de **resultados** es similar a la de detalle de una coevaluación específica de alumno para el profesor, con la diferencia de que el alumno no puede ver quien puso sus notas y de que sus comentarios aparecen de forma desordenada, de tal forma que no pueda rastrear qué compañero fue el que comentó según el orden de aparición.

Tal cual lo anterior, el **seguimiento** de su desempeño resulta ser igual al seguimiento de desempeño para alumnos del usuario profesor.

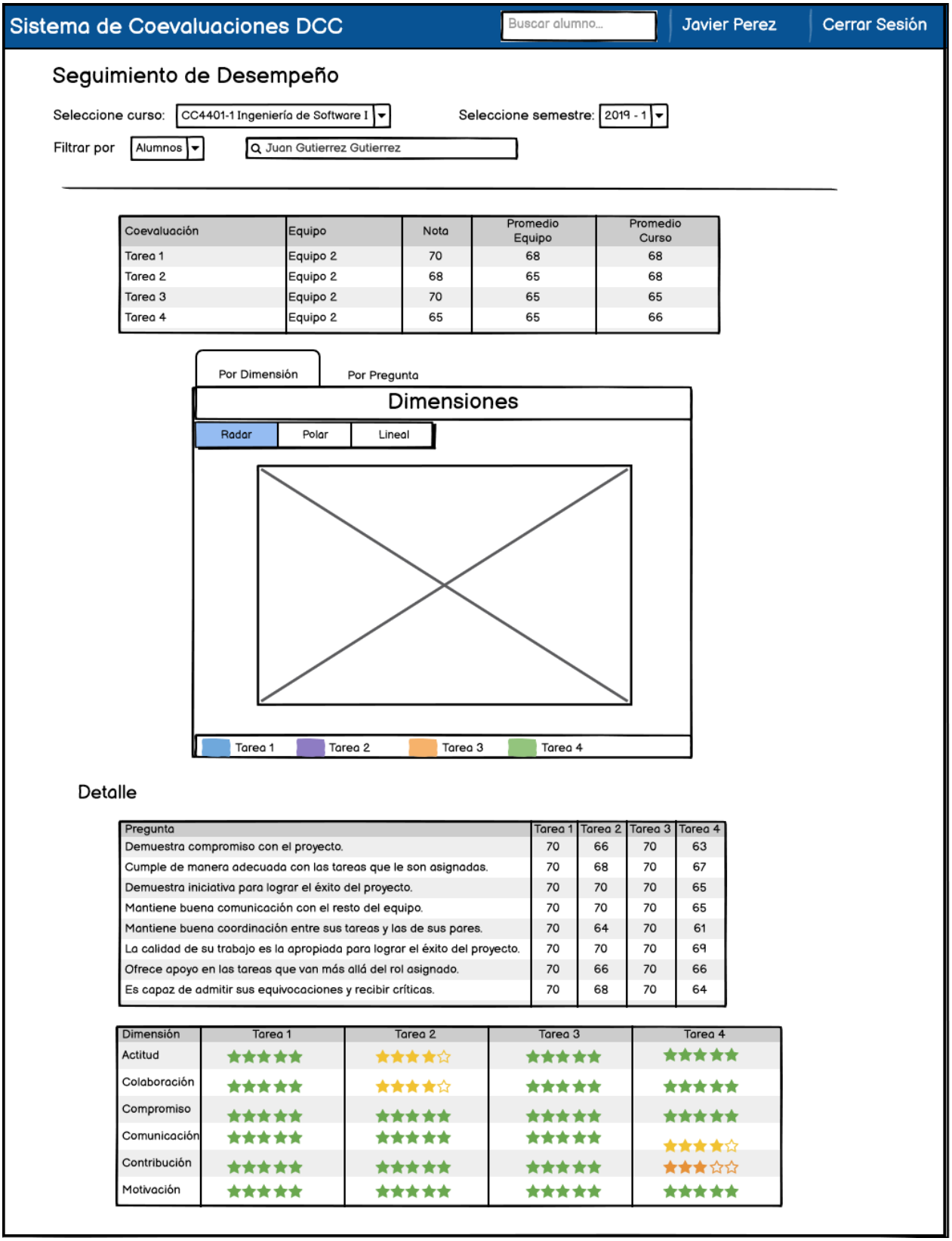



Figura 4.15: Seguimiento de alumno

# Capítulo 5

# Implementación de la Solución

Para abordar la implementación con mayor claridad, se explicarán en primer lugar tres de las vistas más representativas del sistema, detallando decisiones de diseño, relaciones hacia el *back-end*, módulos involucrados y su realización, además de cuando sea pertinente, indicar elementos que son ocupados en otras vistas no mencionadas. En segundo lugar, se hablará sobre funciones originales del sistema desarrollado, las cuales no se encontraban en las versiones anteriores o que contaban con defectos de implementación o diseño que las hacían difíciles de usar.

## 5.1. Vistas críticas del sistema

A continuación se presentan tres vistas: home page profesores, ficha de respuesta de coevaluación para alumnos y resultados coevaluación profesor. En ellas se podrán ver componentes comunes del sistema, las cuales serán mencionadas cuando sea pertinente.

### 5.1.1. Home Page profesores

Esta interfaz es la que recibe a los usuarios de tipo profesor, en donde pueden encontrar un resumen de sus cursos y evaluaciones, abrir nuevas evaluaciones y cursos y redirigirse al seguimiento de desempeño. Para esto, interactúa con los modelos de integrantes, equipos, usuarios, coevaluaciones, cursos y ramos, a través de la muestra de datos y la creación de nuevas entradas.

Lo primero que el sistema hace es verificar el tipo de usuario que está logueado, a través de una búsqueda en la clase `Integrante` del modelo de datos. Si encuentra a un usuario con rol *"Ayudante"*, muestra una vista en donde se debe elegir si se verá el sistema como alumno (para ver sus propios cursos) o como profesor (para ver los cursos donde es ayudante. Esta solución se reconoce como temporal, ya que entrega demasiadas facilidades al usuario ayudante. Si resulta no ser este tipo de usuario, se pregunta por el Rol *"Profesor"*, el que recibe la vista objeto de esta sección. Finalmente, se pregunta si es alumno, el que observa una vista simplificada de la versión del profesor.

La vista de los profesores posee dos tablas: la principal, que muestra las coevaluaciones ordenadas por estado y fecha de término, con celdas clickeables que redirigen al detalle de la evaluación; y la secundaria, que muestra sus cursos ordenados según el semestre más reciente, redirigiendo hacia el modulo de detalle de curso si se hace click. Para los usuarios de tipo *Alumno* solo se muestra la tabla principal.


Figura 5.1: Home page profesor.

Para traer los datos de la tabla principal, se consulta la clase `Coevaluación`, trayendo todas aquellas pertenecientes a los cursos que dicta el profesor, ordenadas según estado (*Publicada, Finalizada, Creada* y *Respuestas Publicadas*), en donde tanto las fechas y el estado son modificados para que se muestren en un formato más claro para el usuario. De forma similar, para la tabla secundaria se traen todos los cursos en que el usuario aparezca como profesor, siendo ordenados de forma descendiente por año y semestre (es decir, del más reciente al más antiguo).

En caso de que las tablas no tengan información, se muestra una fila vacía con un mensaje alusivo a esto para que el usuario sepa que efectivamente no hay información en la base de datos y no crea que es un problema de la carga de datos. Para detectar que no existen datos se utiliza el tag `empty` perteneciente a Django[18]. Este comportamiento fue implementado en todas las tablas del sistema y para todos los usuarios, en los cuales cambia solamente el mensaje dependiendo de cual es el elemento que se muestra.


Figura 5.2: Tabla sin datos.

El ingreso de nuevas coevaluaciones se hace en un formulario que aparece al hacer click sobre el botón “Agregar coevaluación”, lo cual es controlado por JavaScript, cambiando el estado del display desde *none* a *block*. El formulario se muestra en la figura 5.3, donde además de los campos que se muestran en la tabla principal, se debe elegir si la evaluación es también una autoevaluación en un checkbox. Para esta versión se permite solo crear autoevaluaciones asociadas a coevaluaciones, no por sí solas.

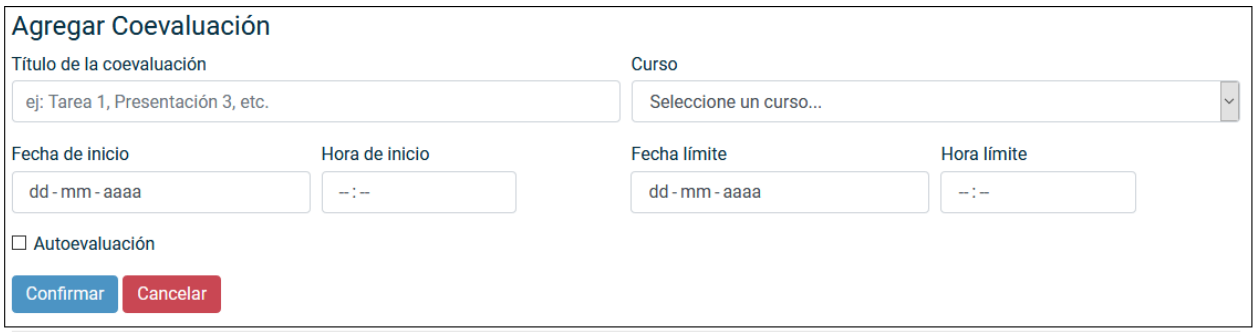



Figura 5.3: Formulario para agregar una coevaluación.

Si el usuario hace click sobre el botón "Guardar", se envía una solicitud de tipo post a la url *agregarCoevaluacion*, la cual toma cada campo del formulario para crear una nueva coevaluación con estado *Creada*, la que es asociada al curso y a todos los equipos que pertenecen a él. En el caso en que el usuario elija la opción "Cancelar", se colapsa el formulario, desapareciendo de la pantalla y ninguno de los campos es almacenado.

El botón "Agregar curso" tiene un funcionamiento similar al de "Agregar coevaluación", en donde la diferencia es el formulario mismo, el cual va asociado a las tablas Ramo y Curso. Los campos a rellenar se pueden observar en la figura 5.4, en donde se observa el checkbox para especificar si el curso tiene o no horas fijas de trabajo. Como se mencionó en el capítulo 2, actualmente, entre los cursos para los que se desarrolló el sistema, el único que cuenta con horas fijas es CC5402 - Proyecto de Software, mas el sistema no establece esto como precondición, siendo obligación del creador del curso especificar este apartado. El botón "Guardar" en este caso también hace una solicitud post, pero hacia la url *agregarCurso*, en donde tanto el curso como el ramo solo son creados en caso de que no existan anteriormente usando el método `get_or_create`[20], según se muestra en el código 5.1, donde se verifica si el código y el nombre ingresados ya existen en Ramo. En caso de existir, el campo `ramo_created` se setea a *true* y en ramo se guarda la entrada coincidente. En el caso contrario, `ramo_created` es *false* y se crea una instancia de `Ramo` con los parámetros buscados (`codigo` y `nombre_ramo`) y los presentes en defaults (`fecha_creacion` y `fecha_modificacion`).[1]

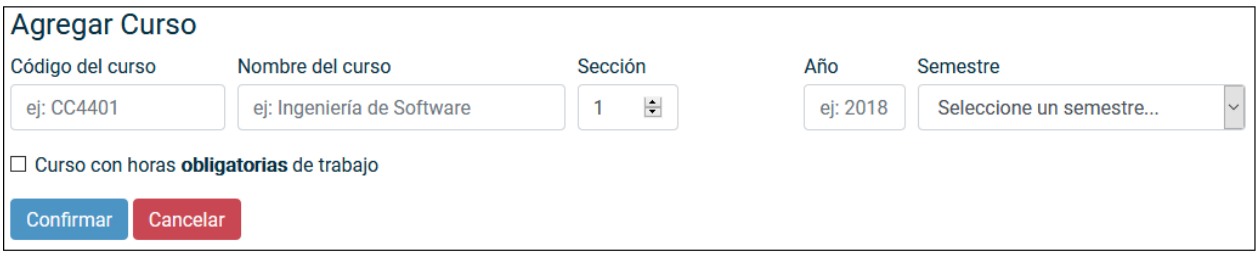



Figura 5.4: Formulario para agregar un curso.

Las filas de ambas tablas redirigen a información más específica, siendo el detalle de coevaluación para la tabla principal y el detalle de cursos en la tabla secundaria.

[1]En este caso el parámetro `nombre` se vuelve a escribir en defaults debido a que la búsqueda es case insensitive, lo que se hace usando el tag *iexact*. Si ese tag no estuviera, se omitiría ese parametro en defaults.[30]

```
        ramo, ramo_created = Ramo.objects.get_or_create(
            codigo=codigo,
            nombre__iexact=nombre_ramo,
            defaults={'nombre': nombre_ramo, 'fecha_creacion':
    fecha_creacion,
                      'fecha_modificacion': fecha_modificacion},
        )
```

Código 5.1: Ejemplo función *get_ or_ create* para un ramo

Finalmente, tanto la creación de una coevaluación y un curso entregan mensajes simples para avisar del éxito o fracaso de la adición de nuevas entidades al modelo.

Similar a esta vista es la perteneciente a alumnos, la cual solo posee la tabla principal como resumen de las coevaluaciones. Se decidió no entregar una tabla con los cursos a los alumnos ya que su cantidad de cursos asociados suele ser poca en comparación a la de los profesores y, debido a que ellos no realizan ningún cambio en el modelo de datos de los cursos, de ser incluida, esta tabla sería solo de visualización, lo cual puede ser reemplazado por la tabla principal, en donde aparecen también los cursos a los que pertenece cada coevaluación, y por tanto, cada alumno.

A diferencia del usuario profesor, la tabla de coevaluaciones para alumnos tiene una fuerte dependencia con el estado en que se encuentran las coevaluaciones. Las coevaluaciones en estado *Creada* no se muestran en la tabla, para las en estado *Publicada* el usuario será redirigido al modulo de respuesta de coevaluación. Luego, cuando la coevaluación pase a *Finalizada*, el click se desactiva, para finalmente, cuando llegue al estado final de *Respuestas publicadas*, el usuario es llevado a sus resultados particulares.

Tablas del mismo estilo que las indicadas en esta vista se pueden encontrar en el detalle de curso para mostrar los nombres de los estudiantes y sus equipos, y las tablas de notas según pregunta, integrante o coevaluación.

### 5.1.2. Ficha de respuesta de coevaluación para alumnos

En la ficha de coevaluación, el alumno podrá responder las coevaluaciones de sus compañeros de grupo y, cuando corresponda, su autoevaluación. Para llegar a esta vista, el usuario de tipo alumno puede hacer click en la tabla principal en una coevaluación con estado *"Publicada"*.

En primer lugar, se observa en el lado derecho superior toda la información relacionada con la coevaluación, tales como nombre, curso, fechas importantes y su estado, proporcionada por la clase `Coevaluacion`. A su lado, se muestran las evaluaciones a responder, aunque en la imagen aparecen dos tablas, una para la autoevaluación y otra para la coevaluación, en caso de que la evaluación no sea autoevaluación, la primera tabla desaparece. Se tomó esta decisión debido a como está construido el sistema, en donde se asume que la evaluación es siempre coevaluación con la opción de ser además autoevaluación. Es por esto, que la tabla de autoevaluación es opcional, mientras que la otra es obligatoria. Para la segunda tabla se traen desde la clase `Grupo` todos los integrantes que sean del mismo grupo que el usuario.


Figura 5.5: Ficha de respuesta de coevaluación.

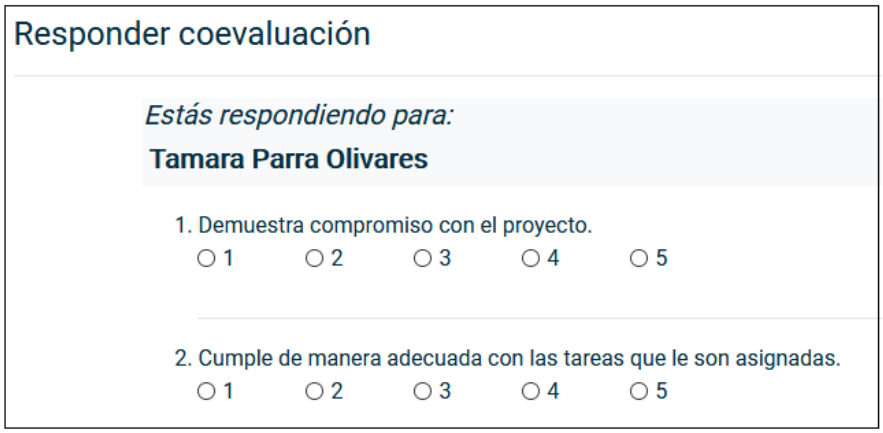



Figura 5.6: Parte de formulario de respuesta coevaluación.

Para responder o editar una coevaluación, el usuario debe hacer click en la fila con el nombre de la persona a evaluar. Esto envía una petición POST a la url *responderCoevaluacion*, la que simplemente redirige hacia la misma página pero proporciona los parámetros de id de coevaluación y el id del estudiante a evaluar.

Para mostrar la evaluación, lo primero que se hace es comprobar si existe o no alguna respuesta ya enviada. Si la hay, significa que se está haciendo una edición, por lo que se marcan las respuestas anteriores, en caso contrario, el formulario aparece en blanco. Solo las 8 primeras preguntas son obligatorias para enviar el formulario, ya que no todos los cursos exigen que se dejen comentarios. Además, las dos preguntas finales de "Fortalezas" y "Debilidades" no aparecen en el formulario de una autoevaluación.

Al hacer click sobre el botón "Enviar coevaluación", el sistema envía una petición de tipo post a `enviarCoevaluacion`, para guardar las respuestas con su encuesta asociada. Para realizar esto, en primer lugar se busca si existe una encuesta asociada con la combinación `evaluador, evaluado, grupo` y `coevaluacion`, y en caso de no existir, se crea una encuesta,

para luego hacer el mismo proceso con las respuestas, es decir, se busca si la respuesta asociada a los parámetros `encuesta` y `pregunta` existe y de no ser así la crea. Es importante notar que cuando la entidad buscada ya existe, se cambian aquellos valores que se editaron al enviar la coevaluación, siendo `nota_calculada` en `Encuesta` y `respuesta` en `Respuesta`. El cambio de respuesta es un reemplazo simple por el nuevo valor, mientras que el cambio en `nota_calculada` incluye calcular la nota en base a las respuestas proporcionadas en el formulario.

Para calcular la nota, se comienza con una variable `nota_calculada` igual a 1, a la cual se le suma el resultado de la multiplicación de cada respuesta con su puntaje, el que como se mencionó antes, depende del tipo de coevaluación que sea. Este valor de tipo *float* se redondea a un solo decimal, pero debido a que la función *round* en Python funciona de forma diferente a la necesitada, se creó un método llamado *round_ normal*, mostrado en el código 5.2, el que se encarga de redondear hacia arriba si el valor de la décima es mayor o igual a 5 y hacia abajo si no lo es.

```
def round_normal(n):
    number = n * 10
    unidad = int(str(int(number))[-1])
    if (unidad >= 5):
        return ceil(number) / 10
    else:
        return floor(number) / 10
```

Código 5.2: Método *round_ normal*

Existen 2 formas de identificar si se ha respondido o no la evaluación de una persona: color e icono. Para respuestas ya enviadas, la fila posee un color verde y un icono de edición, mientras que para respuestas sin enviar, el color de la fila es amarillo y posee un icono para contestar.

Esta vista es única para estudiantes, pero el profesor cuenta con una versión similar para el detalle de una coevaluación, en donde en la parte superior muestra la información general de coevaluación, a su lado la tabla con alumnos pendientes y abajo una lista colapsable de integrantes agrupados por grupos, donde se muestra su promedio general y por pregunta.

### 5.1.3. Resultados coevaluación para profesor

La vista de resultados para el profesor muestra la información del estudiantes, su nota obtenida en la evaluación, el promedio del curso, el detalle de las notas en cada pregunta y un gráfico con el que se muestran las dimensiones. La url en donde se muestra esta vista tiene la estructura `/getAlumno/<int:alumnoId>/detalleCoev/<int:coevId>`, desde donde se obtiene el id del alumno y de la coevaluación.

Para calcular la nota del alumno, se toman todas las encuestas que tienen como evaluado al alumno, sumando el parámetro `nota_calculada`, dividiendo el total por el número de encuestas y redondeándolo. Aquí, se usa el número de encuestas como divisor y no el número de integrantes en el grupo para no considerar a aquellos alumnos que no contestaron las evaluaciones. El proceso para calcular la nota del promedio del curso es análogo al de la nota

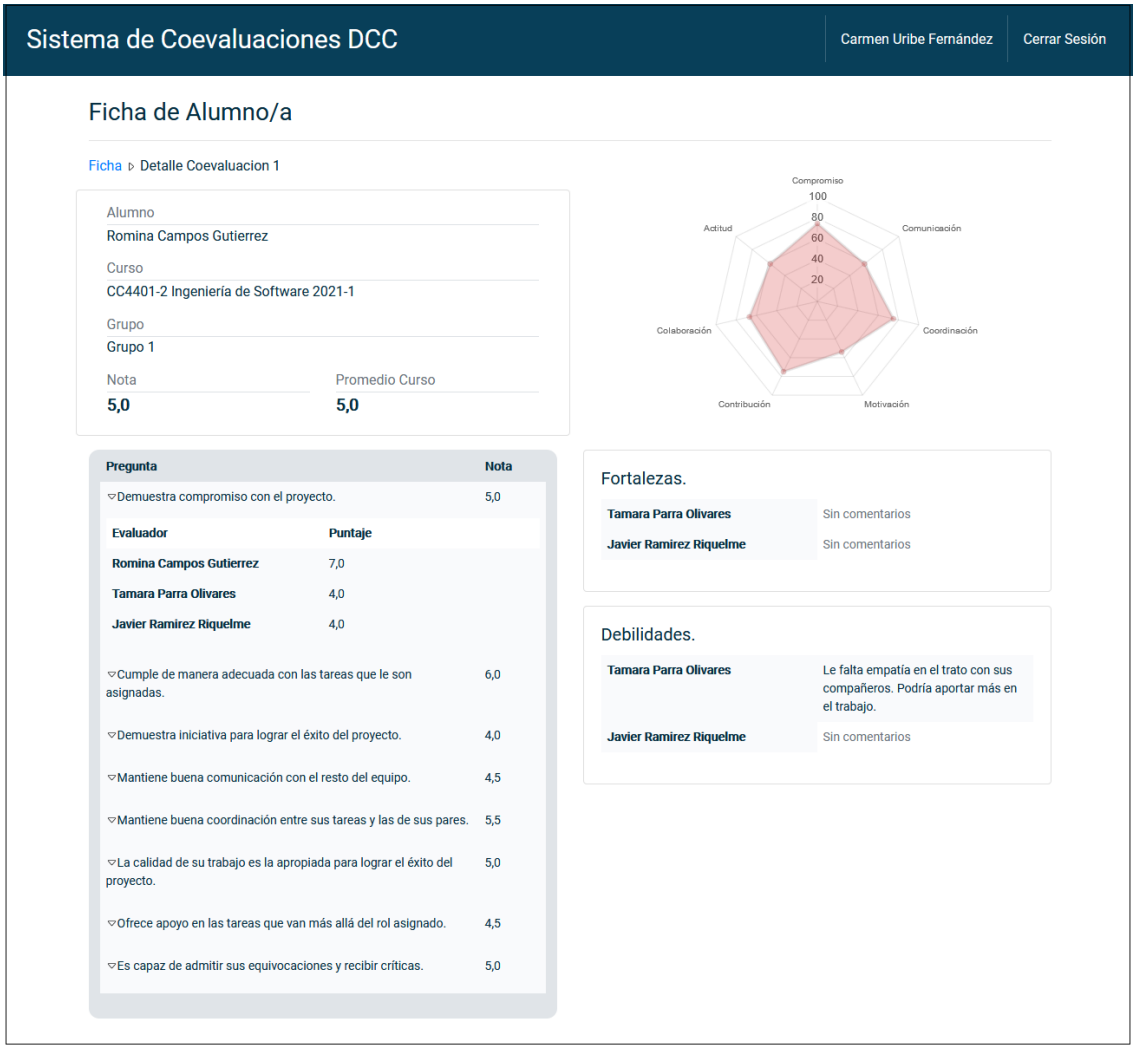



Figura 5.7: Resultados de coevaluación desde perspectiva de profesor.

del alumno, pero en este caso se saca el promedio sobre todas las encuestas pertenecientes al curso, sin hacer la distinción de quién es el evaluado.

Para obtener las notas de cada pregunta de tipo puntaje, se transforman los puntos a notas con la fórmula `puntos * 6.0 + 1.0`, guardando este valor en el parámetro *respuesta*, para luego sacar el promedio de esto con la fórmula `nota_pregunta = float(respuestas.aggregate(avg=Avg('respuesta'))['avg']) * 6.0 + 1.0`. Así es que se guarda en un arreglo de diccionarios la pregunta, todas las respuestas asociadas y el promedio de la pregunta, desde donde se logra obtener además el detalle de las notas individualmente colocadas por cada evaluador al evaluado. En las preguntas de tipo texto se obtiene el mismo arreglo, pero se ignora la primera parte que calcula notas.

El gráfico con el detalle de dimensiones se crea usando un ploteo de tipo radar disponible en la librería Chart.js. Para lograrlo, se creó una función denominada "drawChart" en Javascript que recibe dos parámetros: labels *l* y datos *d*, los que e grafican una vez que la vista sea cargada sobre el canvas de id "myChart", según muestra el código 5.3.

```
1         <div class="col-md-6">
```

```
            <canvas id="myChart" ></canvas>
        </div>

        <script>
            $(document).ready(function () {
                drawChart({{ labels| safe}}, {{ data| safe}} );
            });
        </script>
```

Código 5.3: Gráfico de tipo radar en vista de resultados

Los datos de este gráfico se obtienen de igual manera que la de las notas, calculando el promedio de las respuestas, pero para obtener el porcentaje final se multiplica este promedio por el porcentaje que aporta a la dimensión cada una de las preguntas.

Para los usuarios de tipo alumno la vista es análoga, con la diferencia de que para ellos las respuestas de sus evaluadores son anónimas, no pudiendo ver el detalle de notas en cada pregunta (solo ven el promedio) ni viendo de quién es cada uno de los comentarios. También, para agregar anonimato al proceso, los comentarios son mostrados en orden aleatorio, para que no puedan rastear quien escribe cada uno de ellos a partir del orden en que aparecen.

Hay partes en este diseño que se repiten a lo largo del sistema, tales como la tabla de notas, disponible también en el seguimiento de desempeño y el detalle de una coevaluación para los usuarios profesores, y el gráfico radar de dimensiones, también disponible en el dashboard de seguimiento de desempeño, pero en un formato más avanzado.

## 5.2. Funciones originales

Además de las vistas antes mencionadas, existen algunas funcionalidades de autoría propia de la autora de este documento, las cuales se explicarán a continuación.

### 5.2.1. Autoevaluaciones

La posibilidad de crear autoevaluaciones resulta ser algo nuevo en este sistema, comparado con el sistema en producción. Estas autoevaluaciones van asociadas a las coevaluaciones, siendo representadas por un campo booleano en la clase `Coevaluacion` llamado *"autoevaluacion"*. En la práctica, este tipo de evaluaciones se maneja de igual forma que una coevaluación, pero una respuesta perteneciente a una autoevaluación se puede identificar como aquella en que tanto el evaluador como el evaluado son el mismo en la tabla Encuesta. Dependiendo de cual es el objetivo, son estas dos formas las que se usan para identificar este tipo de evaluaciones.

### 5.2.2. Dashboard de seguimiento de desempeño

El dashboard de seguimiento sirve para mostrar de forma gráfica la evolución de los resultados de los estudiantes con respecto a sus compañeros de grupo o al tiempo. Estos dashboards analizan puntajes por pregunta y dimensión, teniendo algunas diferencias si el seguimiento es a un integrante de un curso o a un equipo. Para realizarlo, se usó JavaScript, los gráficos radar de Chart.js y la librería Random Color para definir los colores.

En el seguimiento de alumnos, el dashboard luce como el de la figura 5.8. A la izquierda, se puede ver el detalle por dimensión, similar al que ya se mostraba en la sección 5.1.3 mientras que a la derecha se observa por preguntas.

Para cargar la información relacionada a las notas de preguntas, en `views.py` se crea la función `getPromedioPreguntasCoevaluaciones`, función que recibe la lista de coevaluaciones pertenecientes al curso buscado y el usuario de interés `alumno` y entrega un arreglo de diccionarios, los que contienen el texto de la pregunta y los promedios calculados. Para calcular los promedios, la función obtiene desde el modelo `Pregunta` todas las preguntas de la coevaluación que sean del tipo `puntaje`, las recorre calculando para cada una de las coevaluaciones buscadas las encuestas relacionadas al alumno y la coevaluación y sus respuestas según correspondan a la pregunta. Luego, la función `getPromedioPreguntasCoevaluaciones` calcula el promedio de todas respuestas usando el método de agregación `Avg` disponible en Django [19] y, como las notas en las respuestas viene en formato de porcentaje (con opciones de 0, 0.25, 0.5, 0.75 o 1.0), el valor obtenido se multiplica por 6.0 y se le suma 1.0, las cuales finalmente se agregan a un arreglo que contiene estos promedios para cada coevaluación. Una vez que todas las coevaluaciones son recorridas, se crea un diccionario que asocia la pregunta y su arreglo de promedios según coevaluación, el que se va agregando al arreglo final. Cuando todas las preguntas son recorridas, se retorna algo similar a lo mostrado en el código 5.4

```
grafico_integrante = [{pregunta:"P1", notas:[60, 40, 70]},
            {pregunta:"P2", notas:[70, 67, 70]},
            ...,
            {pregunta:"P8", notas:[70, 70, 70]}]
```

Código 5.4: Ejemplo estructura de datos para seguimiento de integrante con 3 evaluaciones y 8 preguntas.

Para el caso del cálculo de puntajes en Dimensiones, `getPuntajeDimensionesIntegrantes` funciona de la misma forma que `getPromedioPreguntasCoevaluaciones`, pero en este caso la primera iteración se hace sobre las dimensiones del sistema presentes en el modelo `Dimension` de la base de datos, y no sobre las preguntas, por lo que si se observa el código 5.4, se reemplaza el valor de `pregunta` por la dimensión.

Estos resultados para notas y preguntas son entregados a la vista de seguimiento de desempeño a través del contexto, en donde al ser cargada, la vista llama a la función `graficosRadar` creada en JavaScript, la cual se encarga de graficar la información de la misma forma que se explicó anteriormente para la vista de resultados para profesores. (sección 5.1.3).

Para el caso de seguimiento de equipos existe un gráfico por evaluación y los datos corresponden a las notas obtenidas por cada uno de los integrantes del equipo. Esto es similar al caso anterior de seguimiento por integrante, siendo el único cambio la estructura de los datos con los que se grafica. Si antes el arreglo de datos quedaba como en el código 5.4, en este punto los datos quedan como en el código 5.5, en donde en vez de llamar a las funciones `getPromedioPreguntasCoevaluaciones` y `getPuntajeDimensionesIntegrantes` con un arreglo con todas las coevaluaciones a graficar, aquí se llaman con una sola coevaluación, y su resultado se agrega a un diccionario que tiene el nombre de la coevaluación con la que se llamó al sistema y el promedio calculado, el cual se agrega a un arreglo general que contiene esta estructura para todas las coevaluaciones del curso.


Figura 5.8: Dashboard seguiimiento de alumnos.

```
grafico_equipo = [{nombre_coev:"Coev 1", promedios:[{pregunta:"P1", notas
    :[60]}, {pregunta:"P2", notas:[70]},...,{pregunta:"P8", notas:[70]}]},
    ...,
    {nombre_coev:"Coev 3", promedios:[{pregunta:"P1", notas:[50]}, {
    pregunta:"P2", notas:[68]},...,{pregunta:"P8", notas:[68]}]}
    ]
```

Código 5.5: Ejemplo estructura de datos para seguimiento de equipo con 3 evaluaciones y 8 preguntas.

Estos datos son enviados según el caso (integrante o equipo) a través de la vista al template especial de seguimiento, el cual lo recibe y pasa a una función programada en JavaScript, la cual se encarga de parsear los datos y graficarlos de manera similar a lo que se hace en la vista de resultados, sección 5.1.3.

### 5.2.3. Drag and drop para edición de equipos

Para mover estudiantes entre equipos se ocupa una interacción de tipo drag and drop entre los elementos de listas no ordenadas, dejando fijas las filas "headers" de las listas que indican el nombre del grupo. Para implementar esto, se utilizó Sortable de jQuery UI.

El código para implementar esto, tal como se muestra en el código 5.6 resulta ser bastante sencillo, en donde se conectan las listas de alumnos con y sin equipos por la clase `connectedSortable`, presente en todas, para poder mover elementos de tipo `li:not(.ui-state-disabled)` entre ellas. Esta última sentencia indica que se moverán todos los elementos de la lista que no tengan la clase `.ui-state-disabled`, la cual solo está presente en las filas que tienen los nombres de los equipos.

```
$(function () {
  $("#equipos, #sin-grupo ").sortable({
    connectWith: ".connectedSortable",
    items: "li:not(.ui-state-disabled)",
  });
});
```

Código 5.6: Drag and drop en equipos

### 5.2.4. Importación archivos para carga de estudiantes

El sistema maneja dos formas de agregar integrantes a los cursos: individualmente o importando una planilla con extensión xls. Para importar archivos, se adjunta un archivo y se hace click en el botón "Cargar lista", lo que envía una petición post al sistema. El archivo de carga tiene la misma estructura que el que se obtiene al exportar las notas de una evaluación desde U-Cursos, a la que se le agrega la columna "Correo" para ingresar los correos de los estudiantes.

Para explicar cómo se convierten las planillas, se usará el archivo de ejemplo de mostrado en la tabla 5.1. Desde la vista, se toma el archivo y, usando la librería pyexcel y el método `get_records`, se retorna una lista de diccionarios de los registros presentes en el archivo importado según el código 5.7, en el que cada diccionario representa una fila de la plantilla.

| Nº | Alumno | RUT | Estado | Observaciones | Nota | Correo |
|---|---|---|---|---|---|---|
| | Proyectos - Campana | | | | | |
| 1 | Cáceres Muñoz, Josefa Ignacia | 5555555-5 | Activo | | | caceresmunoz@mail.com |
| 2 | Gutiérrez Parra, Antonia Zuñiga | 20591300-9 | Activo | | | gantonia@mail.com |
| 3 | Vallejos Perez, Luis Miguel | 20470848-7 | Activo | | | vallejosluis@mail.com |
| | Proyectos - Portal postulación | | | | | |
| 4 | Araya Fuentes, José Miguel | 20471185-2 | Activo | | | josemiguelaraya@mail.com |
| 5 | Ferrera Ferrera, Sebastián Pablo | 20416785-0 | Activo | | | sebastianferrera2000@mail.com |

Tabla 5.1: Ejemplo de archivo de integrantes.

```
[OrderedDict([('N', ''), ('Alumno', 'Proyectos - Campana'), ('RUT', ''), (
    'Estado', ''), ('Observaciones', ''), ('Nota', ''), ('Correo', '')]),
OrderedDict([('N', ''), ('Alumno', 'Caceres Munoz, Josefa Ignacia'), ('RUT
    ', '5555555-5'), ('Estado', 'Activo'), ('Observaciones', ''), ('Nota',
    )'', ('Correo', 'caceresmunoz@mail.com')]),
...,
OrderedDict([('N', 5), ('Alumno', 'Ferrera Ferrera, Sebastian Pablo'), ('
    RUT', '20416785-0'), ('Estado', 'Activo'), ('Observaciones', ''), ('
    Nota', ''), ('Correo', 'sebastianferrera2000@mail.com')])]
```

Código 5.7: Resultado de get_records sobre plantilla

Este resultado se parsea usando la función `parse_plantila_notas(records)`. En esta función se asume que toda línea que no posea RUT pertenece al nombre del grupo (el cual está escrito en la columna Alumno) y los alumnos que siguen a esta fila son sus integrantes, hasta que se vuelva a encontrar otra línea sin RUT, la que identificará el termino del grupo actual y se pasará a un nuevo grupo. Los alumnos sin grupo deberán estar bajo la fila de grupo con nombre *Sin Grupo*.

```
[{'grupo': 'Proyectos - Campana', 'rut': '55555555', 'nombres': 'Josefa
    Ignacia', 'apellidos': 'Caceres Munoz', 'correo': 'caceresmunoz@mail.
    com'},
{'grupo': 'Proyectos - Campana', 'rut': '205913009', 'nombres': 'Antonia
    Zuniga', 'apellidos': 'Gutierrez Parra', 'correo': 'gantonia@mail.com'
    },
...,
{'grupo': 'Proyectos - Portal postulacion', 'rut': '204167850', 'nombres':
     'Sebastian Pablo', 'apellidos': 'Ferrera Ferrera', 'correo': '
```

```
    sebastianferrera2000@mail.com'}]
```

Código 5.8: Resultado de parseo de records

Considerando esto, se va iterando record por record, aplicando la regla de los nuevos grupos cuando sea pertinente, y para cada nuevo record de alumno en primer lugar se revisa si este ya está presente en la base de datos o es un registro totalmente nuevo. En caso de ya estar en la base, se toma su correo desde ella, de lo contrario, se sigue con el especificado en la plantilla. Así es como se crea por cada registro un diccionario que contiene grupo al que pertenece, rut, nombres, apellidos y correo; con los nombres y apellidos siendo obtenidos al llamar a la función `split(', ')` sobre la columna Alumno. Todos los diccionarios son agregados a la variable de tipo arreglo `curso`, la cual se retorna cuando termina la iteración por los registros.

Así, estos resultados son mostrados en la vista de edición de integrantes haciendo una iteración sobre cada entrada del arreglo, llamando al valor según la columna correspondiente.

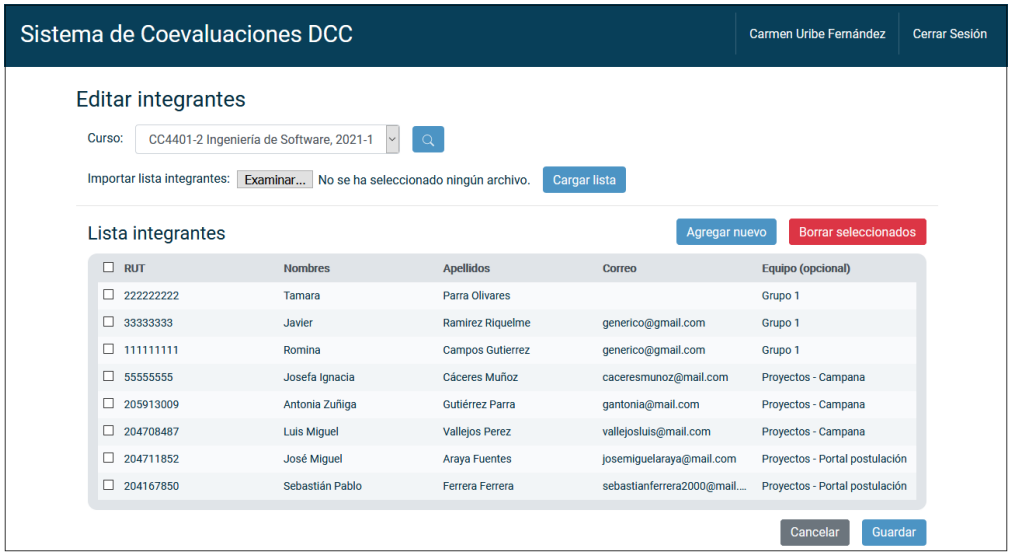



Figura 5.9: Importación de archivo de estudiantes.

# Capítulo 6

# Validación

A medida que se implementaban funciones, se fueron haciendo distintas validaciones para comprobar el funcionamiento y usabilidad de estas, tales como cálculos con programas en Python, entrevistas a distintos tipos de actores del sistema y encuestas. A continuación se verán cada una de las validaciones hechas al sistema desarrollado, la metodología ocupada, sus resultados y los análisis a estos.

## 6.1. Calculadora de notas

### 6.1.1. Metodología

El primer paso para una validación, fue comprobar que las notas estuvieran siendo calculadas de forma correcta. Como cada pregunta aporta en un porcentaje distinto a la nota final, y este porcentaje depende además del tipo de curso que se está evaluando, el cálculo no era fácil comprobar manualmente, tomándose la decisión de hacer un programa simple en Python que fuera capaz de recibir el tipo de curso y el puntaje asignado en cada pregunta, y entregar la nota calculada.

Este programa pide por consola el tipo de curso "con" o "sin" horas fijas y la nota de cada pregunta con un puntaje de 0 a 70 (sin decimal). Luego, llama a la función `calculadora`, la cual comienza con un arreglo en 0 de 8 elementos llamado `notasDimension`, para luego rellenarlo recorriendo un arreglo de diccionarios que contiene la dimensión, las preguntas a las que se asocia y su porcentaje, según muestra el código 6.1, multiplicando el porcentaje relacionado a la pregunta de la dimensión considerada.

```
dimension_pregunta = [{"dimension": 7, "pregunta": 3, "porcentaje": 33.3},
                      {"dimension": 7, "pregunta": 7, "porcentaje": 33.3},
                      {"dimension": 7, "pregunta": 8, "porcentaje": 33.3},
                      {"dimension": 6, "pregunta": 2, "porcentaje": 25.0},
                      {"dimension": 6, "pregunta": 6, "porcentaje": 25.0},
                      {"dimension": 6, "pregunta": 7, "porcentaje": 50.0},
                      {"dimension": 5, "pregunta": 2, "porcentaje": 50.0},
                      {"dimension": 5, "pregunta": 6, "porcentaje": 50.0},
                      {"dimension": 4, "pregunta": 3, "porcentaje": 50.0},
                      {"dimension": 4, "pregunta": 7, "porcentaje": 50.0},
```

```
11                          {"dimension": 3, "pregunta": 5, "porcentaje":
    100.0},
12                          {"dimension": 2, "pregunta": 4, "porcentaje":
    100.0},
13                          {"dimension": 1, "pregunta": 1, "porcentaje": 50.0},
14                          {"dimension": 1, "pregunta": 2, "porcentaje": 50.0}]
```

Código 6.1: Variable dimension_pregunta

Usando el nuevo valor del arreglo `notasDimension` y según dependa de si el curso es con o sin horas fijas, se hace una sumatoria de cada elemento del arreglo por el porcentaje que la dimensión le asigna a cada pregunta según los datos de la tabla 2.2, el cual se retorna e imprime en pantalla.

Para llevar a cabo la validación, se probaron distintos conjuntos de notas, pero para este informe solo se presentaran 3 conjuntos representativos (2 casos bordes y uno general), mostrados en arreglos de la forma `[P1, P2, P3, P4, P5, P6, P7, P8]`:

1. `[10, 10, 10, 10, 10, 10, 10, 10]`
2. `[70, 70, 70, 70, 70, 70, 70, 70]`
3. `[66, 59, 48, 62, 55, 48, 51, 62]`

### 6.1.2. Resultado y Análisis

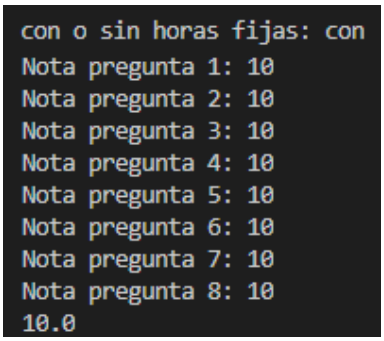
En un principio, al comparar resultados entre el sistema y la calculadora, se vieron casos en que habían diferencias decimales entre los resultados, por ejemplo, en el sistema se calculaba la nota como 6.0 y en la calculadora se obtenía 59 (o 5.9). Esto hizo posible descubrir el problema que tenía el sistema: el redondeo de la segunda cifra fallaba por el funcionamiento de la función `round`, el que producía que cifras como 6.55 fuera redondeado a 6.5 y no 6.6. Luego de arreglar este error según se explicó en la sección 5.1.2, todas las notas igualaban según lo esperado.

Ejemplo de esto son los tres casos enunciados anteriormente mencionados. En el primero, la figura 6.1 muestra los resultados obtenidos en consola por la calculadora y el resultado obtenido en un curso sin horas fijas, siendo esto igual al esperado de 10. En el segundo y tercero ocurre lo mismo, salvo diferencias decimales debido a que la calculadora no usa ninguna función para redondear debido al problema antes mencionado, pero se puede observar que al redondearlo dan el mismo valor que el sistema.

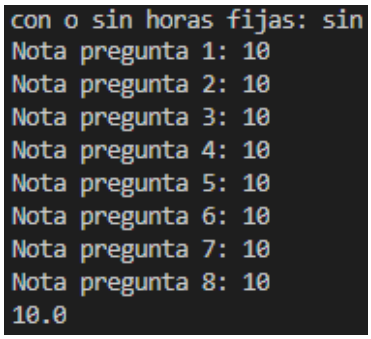
## 6.2. Primera validación de respuestas a coevaluaciones

### 6.2.1. Metodología

El día 10 de Noviembre de 2020 se abrió una convocatoria mediante U-Cursos a través de los foros de la comunidad “Alumnos DCC” y el curso “CC4401 - Ingeniería de Software 1” en su versión Primavera 2020 para invitar a los alumnos a participar en una entrevista privada donde se evaluaría la usabilidad del sistema en relación a contestar coevaluaciones y autoevaluaciones. El objetivo era conseguir al menos dos estudiantes cursando actualmente



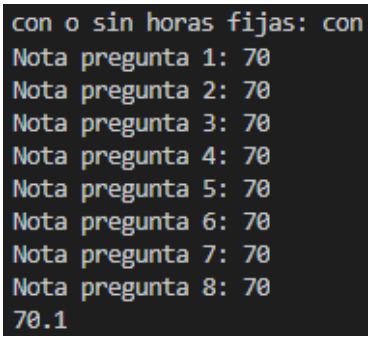

(a) Caso con horas fijas.



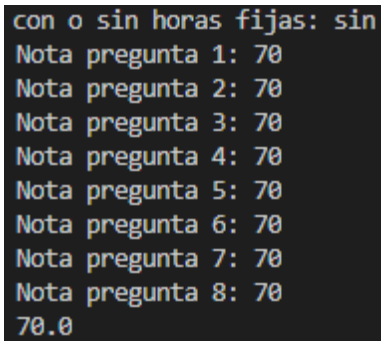

(b) Caso sin horas fijas.


(c) Resultado en sistema.

Figura 6.1: Cálculo caso 1: Solo 10



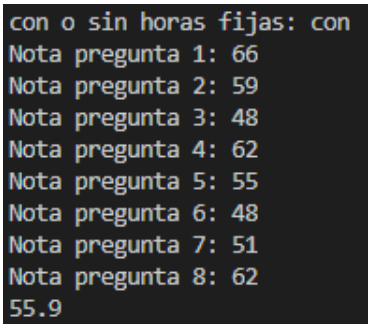

(a) Caso con horas fijas.



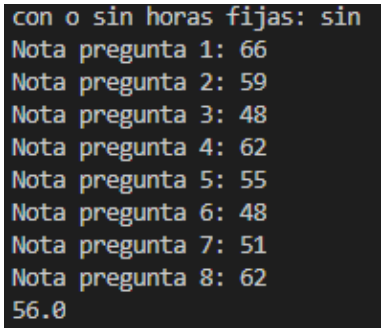

(b) Caso sin horas fijas.


(c) Resultado en sistema.

Figura 6.2: Cálculo caso 2: Solo 70




(a) Caso con horas fijas.




(b) Caso sin horas fijas.


(c) Resultado en sistema.

Figura 6.3: Cálculo caso 3: Distintas notas

CC4401, para así representar a usuarios con poca experiencia en el sistema en producción, y 4 estudiantes que hubieran usado el sistema entre Otoño 2019 y Primavera 2020.

Así fue que desde el 11 al 19 de Noviembre se realizaron entrevistas privadas con 8 estudiantes del Departamento de Computación, 2 de ellos usuarios relativamente nuevos al sistema (solo llevaban una coevaluación usando el sistema), y 6 usuarios experimentados, habiendo usado el sistema en al menos 2 cursos finalizados y uno en curso. Al preguntarles cuando fue la última vez que utilizaron el sistema, un 50 % indicó primavera 2020, un 25 % durante otoño 2020 y 12.5 % en otoño 2018 y primavera 2019.

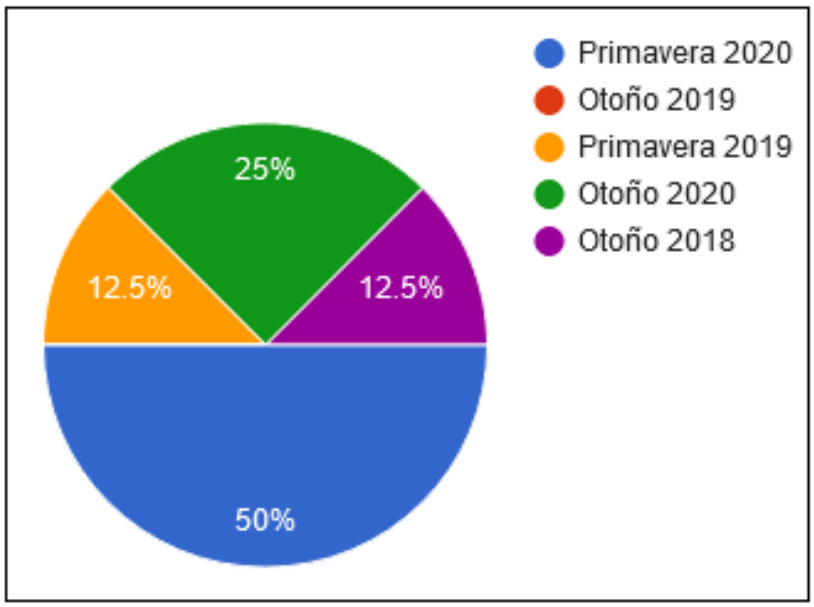



Figura 6.4: Primera validación: ¿Cuando usó el sistema por última vez?.

Todas las entrevistas fueron usando la plataforma Zoom, y para llevar a cabo la dinámica se usaron las herramientas de compartir pantalla y control remoto. La experiencia consistía en presentarle 4 tareas a los usuarios en forma secuencial, pasando a la siguiente cuando el usuario expresaba como terminada la tarea en curso (sin revelarle si la había cumplido con éxito o no), y al finalizarlas todas pedirle que contestara un cuestionario SUS realizado con Google Forms. Además, para conseguir información extra sobre el diseño o la interacción con la interfaz, se le pedía también al entrevistado ir expresando sus pensamientos a medida que realizaba las tareas y al final se le preguntaba cualquier comportamiento inesperado que se pudiera haber observado durante la interacción con el sistema.

En la dinámica, el usuario era Romina Campos Gutiérrez, estudiante de CC4401 y perteneciente al grupo llamado Grupo Test, donde comparte con otros 3 compañeros. Para comenzar, el entrevistado empezaba fuera del sistema, en la página de ingreso a él, en donde debía acceder con el usuario “111111111” y clave “contraseña” proporcionados por el entrevistador. Una vez adentro, debía contestar la evaluación llamada “Coev y Autoevaluacion test”, la cual era una auto y coevaluación. Después, su tarea era editar la coevaluación de su compañero de equipo “Ignacio Parra Olivares”. Finalmente, y luego de que la evaluación cambiara de estado a “Respuesta publicada”, el entrevistado debía revisar sus resultados. Los enunciados

de todas estas tareas fueron mostrados en diapositivas de Google Slides[3].

El cuestionario de Google Forms se dividía en dos partes: La primera para caracterizar al usuario preguntándole los cursos que había cursado y cuando usó el sistema por última vez; y la segunda, la cual era la encuesta SUS en sí, ocupando las preguntas nombradas en el capítulo 2.4, y una pregunta opcional para comentarios.

### 6.2.2. Resultados y Análisis

La entrevista duró en promedio 10 minutos y 30 segundos, viéndose un esperable aumento en los tiempos de los usuarios con menor experiencia en el sistema, principalmente porque se detenían a leer las preguntas completas del formulario de la coevaluación, mientras que a los usuarios avanzados les bastaba con leer el comienzo. En la tabla 6.1 y la figura 6.5 se pueden observar los tiempos de los participantes, mostrándose en la tabla ordenado por mejores tiempos.

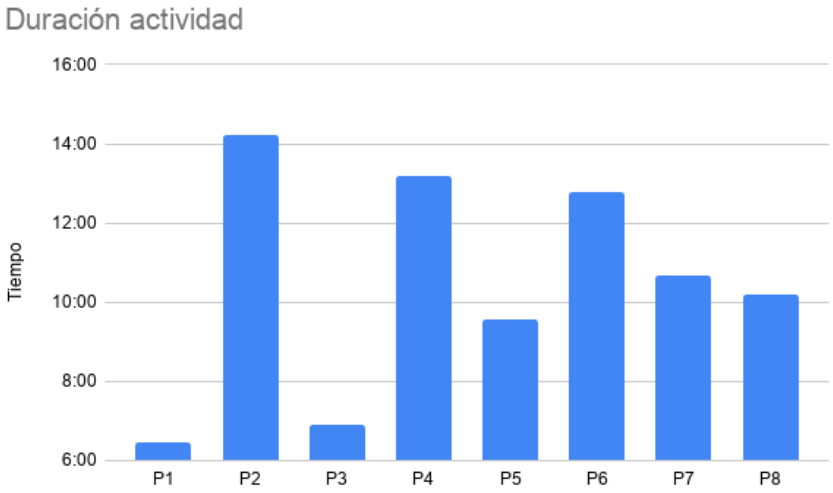



Figura 6.5: Primera validación: Duración actividad.

Todos los participantes lograron finalizar las tareas con éxito, entregando en general buenas críticas, con ciertos reparos como que para contestar la coevaluación de un compañero se debía hacer click específicamente en el icono para contestar, la pérdida del nombre del evaluado al hacer *scroll* en el formulario de la evaluación y el detalle de que no había ningún distintivo de que se estaba contestando la autoevaluación cuando así era. De forma cuantitativa, la encuesta SUS obtuvo un promedio de 92.18 puntos y una mediana de 93.75, encontrándose todos los puntajes sobre los 80 puntos, lo cual se considera una A[26], concluyéndose que el sistema, en términos de usabilidad con respecto al flujo simple de una evaluación (ingresar - contestar - editar - revisar) presenta buenos resultados.

Los comentarios de la encuesta también reflejan un buen recibimiento del sistema, centrándose especialmente en el nuevo diseño: "(sic) esta mas bonita la interfaz y el gráfico de los resultados", "Se ve bien bien amigable a la vista y los pasos para completar las tareas me parecen bastante intuitivos. Sí lo usaría de volver a pasar por algún curso que necesite de

| | Tipo | Tiempo |
|---|---|---|
| P1 | Avanzado | 6:27 |
| P3 | Avanzado | 6:54 |
| P5 | Avanzado | 9:35 |
| P8 | Avanzado | 10:11 |
| P7 | Nuevo | 10:41 |
| P6 | Avanzado | 12:46 |
| P4 | Avanzado | 13:12 |
| P2 | Nuevo | 14:14 |

Tabla 6.1: Primera validación: Tabla con resultados de tiempos.

| Participante | A1 | A2 | A3 | A4 | A5 | A6 | A7 | A8 | A9 | A10 | Resultado SUS |
|---|---|---|---|---|---|---|---|---|---|---|---|
| P1 | 3 | 1 | 5 | 1 | 5 | 1 | 5 | 1 | 5 | 1 | 95 |
| P2 | 5 | 1 | 4 | 1 | 4 | 2 | 5 | 2 | 5 | 1 | 90 |
| P3 | 5 | 1 | 4 | 1 | 5 | 1 | 5 | 1 | 4 | 1 | 95 |
| P4 | 5 | 1 | 4 | 2 | 4 | 1 | 4 | 4 | 4 | 1 | 80 |
| P5 | 4 | 1 | 5 | 1 | 5 | 1 | 4 | 1 | 5 | 1 | 95 |
| P6 | 4 | 1 | 5 | 2 | 5 | 2 | 5 | 1 | 5 | 1 | 92.5 |
| P7 | 5 | 1 | 4 | 1 | 5 | 1 | 5 | 1 | 5 | 1 | 97.5 |
| P8 | 4 | 1 | 5 | 1 | 4 | 1 | 5 | 1 | 4 | 1 | 92.5 |

Tabla 6.2: Primera validación: Puntaje SUS.

coevaluación." y "La nueva versión se ve mucho más mejorada y mejor forma de ver la infomación", son prueba de que la interfaz para contestar coevaluaciones cumple con el principio de ser fácil de usar buscado desde la propuesta de la solución.

El resto de los resultados de la encuesta se encuentra en el Apéndice D.

## 6.3. Validación final del sistema con profesores

### 6.3.1. Metodología

En una videollamada realizada por Zoom con Milenko Tomic y Jocelyn Simmonds el día 17 de Diciembre de 2020, se realizó una demostración completa del sistema, mostrando el funcionamiento de la plataforma para los usuarios alumno y profesor. Esta reunión tuvo una duración de 1 hora y 30 minutos, donde a medida que se enseñaban las partes del sistema se resolvían dudas y recibían comentarios, para terminar con comentarios finales sobre el sistema.

### 6.3.2. Resultados y Análisis

Los comentarios recibidos permitieron recabar una lista de elementos por implementar al sistema, entre los cuales se encuentran los siguientes:

1. En todas las interfaces del sistema se debe separar las autoevaluaciones de las coevaluaciones, siendo necesarias considerarlas (al menos en el frontend) como una entidad

aparte a las coevaluaciones. Esto considera notas, resultados y seguimiento de desempeño.

2. En elementos como tablas y gráficos, cambiar los enunciados de las preguntas por identificadores, para así ocupar de mejor manera los espacios.
3. Integrar la opción de recuperar contraseña.
4. Uniformizar estilos, considerando tamaños de fuentes de texto, colores de botones y colores en vistas.
5. Orden en que la información se muestra en tablas. Establecer un orden de acuerdo a la tabla, por ejemplo, para coevaluaciones que sea por estado (las que se encuentren publicadas que aparezcan primero) y fecha límite.
6. Revisión de interfaces en casos extremos, como muchos integrantes en grupos o comentarios muy largos en las preguntas de fortalezas y debilidades.
7. Formateo automático del RUT al ingresarlo en el login.
8. Recuperación de contraseña automático y sin tener que contactar a un administrador.

A excepción de los dos últimos, todos estos puntos fueron considerados e implementados en la versión final de la memoria, produciendo un resultado final con mejor diseño y más uniforme, y mejorando la lectura de tablas y gráficos.

Como comentarios finales, tanto Tomic como Simmonds expresaron satisfacción con el nuevo sistema, reconociéndolo como una mejor opción que el sistema en producción y considerando que en esta versión las funcionalidades se encuentran más a la vista, sin ser necesaria una búsqueda tan amplia para llegar a lo que se desea.

## 6.4. Validación final con alumnos del curso CC4401

### 6.4.1. Metodología

Por último, el sistema se evaluó en un escenario real, realizando la segunda coevaluación del curso *CC4401 - Ingeniería de Software I* en el sistema. Para lograr esto, se subió la aplicación a la plataforma Python Everywhere, la cual ofrece acceso a máquinas con ambiente de Python instalado.[8] Como se utilizó la versión gratuita del sistema, la dirección web quedó como `http://carolinacontreras.pythonanywhere.com`.

Usando el administrador de Django, se creo el primer curso y se agregó como integrante de tipo Profesor a un usuario de prueba, con el cual desde el portal en sí se creó el curso, con sus 42 estudiantes y sus 9 grupos de 4 o 5 integrantes cada uno, y la coevaluación con nombre "Sprint 2" con fecha de inicio 23 de Diciembre de 2020 y fecha de término el 30 de Diciembre de 2020. Al finalizar esto, se reemplazó al usuario de prueba por la profesora del curso.

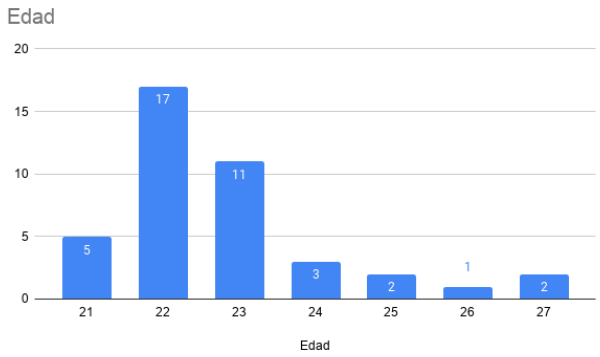
Una vez abierta la coevaluación, se avisó mediante el foro de U-Cursos del uso de esta nueva plataforma para evaluar a sus compañeros, informándoles además de la existencia de una encuesta opcional para que respondieran después de haber respondido la coevaluación. Todos los alumnos recibieron su usuario y contraseña a través de un email.

### 6.4.2. Resultados y Análisis

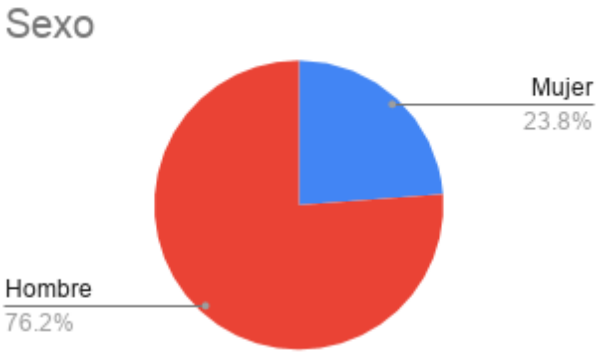
Los participantes de este estudio tomaban todos por primera vez el curso, con edades entre los 21 y 27 años, en donde un 23.8 % eran mujeres. El número de cursos inscritos fluctúa entre 2 a 9, con 12 a 48 créditos inscritos según se aprecian en las siguientes figuras.

Hubo un total de 40 respuestas en el sistema de coevaluación, de las cuales solo una persona presentó problemas menores para ingresar sus credenciales, los cuales no tenían aparente relación con el sistema. Por otra parte, la encuesta fue contestada por un 38 % de estudiantes, versus el óptimo de 24 % considerado al momento de crear la encuesta.

La encuesta aplicada se dividía en dos partes: una encuesta SUS con sus 10 afirmaciones, debiendo contestar en una escala de 1 a 5 (1 siendo muy en desacuerdo y 5 muy de acuerdo), y una parte para comentarios opcionales comparando con el sistema en producción y generales. En total, la encuesta logró buenos resultados, con un puntaje promedio de 85 y mediana 91,25. A continuación, se comentaran un par de preguntas que se consideran que entregan mayor información, mas el resto son las preguntas que aparecen en el Apéndice E.

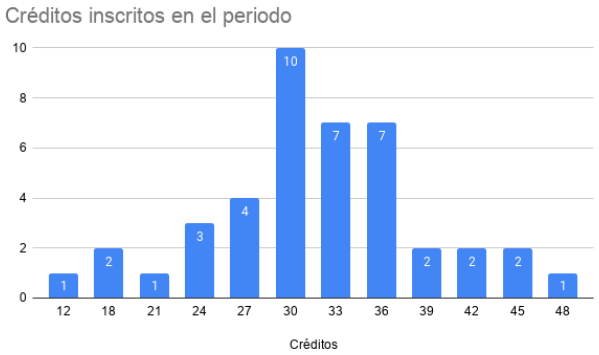



(a) Edades participantes.




(b) Sexo participantes.


(c) Cursos inscritos por participantes en periodo actual.




(d) Créditos inscritos por participantes en periodo actual.

| Participante | A1 | A2 | A3 | A4 | A5 | A6 | A7 | A8 | A9 | A10 | Resultado SUS |
|---|---|---|---|---|---|---|---|---|---|---|---|
| P1 | 3 | 3 | 4 | 2 | 5 | 1 | 5 | 1 | 4 | 1 | 82.5 |
| P2 | 5 | 1 | 5 | 1 | 5 | 1 | 5 | 1 | 5 | 1 | 100 |
| P3 | 5 | 1 | 5 | 1 | 5 | 1 | 5 | 1 | 5 | 1 | 100 |
| P4 | 2 | 4 | 3 | 2 | 3 | 3 | 2 | 3 | 3 | 1 | 50 |
| P5 | 2 | 3 | 3 | 1 | 5 | 2 | 3 | 2 | 3 | 1 | 67.5 |
| P6 | 3 | 1 | 3 | 1 | 5 | 1 | 5 | 2 | 3 | 1 | 82.5 |
| P7 | 3 | 1 | 5 | 1 | 4 | 1 | 5 | 1 | 4 | 1 | 90 |
| P8 | 5 | 1 | 5 | 1 | 4 | 1 | 5 | 1 | 5 | 1 | 97.5 |
| P9 | 5 | 1 | 5 | 1 | 5 | 1 | 5 | 1 | 4 | 1 | 97.5 |
| P10 | 5 | 2 | 2 | 4 | 4 | 1 | 5 | 2 | 3 | 1 | 72.5 |
| P11 | 5 | 1 | 5 | 1 | 5 | 3 | 5 | 1 | 4 | 1 | 92.5 |
| P12 | 3 | 1 | 5 | 1 | 5 | 2 | 5 | 1 | 5 | 1 | 92.5 |
| P13 | 3 | 3 | 2 | 1 | 2 | 1 | 4 | 2 | 4 | 1 | 67.5 |
| P14 | 5 | 1 | 4 | 1 | 5 | 1 | 5 | 1 | 4 | 1 | 95 |
| P15 | 4 | 1 | 5 | 1 | 4 | 2 | 5 | 1 | 3 | 2 | 85 |
| P16 | 5 | 1 | 5 | 1 | 5 | 1 | 5 | 1 | 5 | 1 | 100 |

Tabla 6.3: Validación final: Puntaje SUS.

A la afirmación "Encuentro este Sistema de Coevaluación innecesariamente complejo.", un 68.8 % estuvo muy en desacuerdo, un 6.3 % en desacuerdo, un 18.8 % ni de acuerdo ni en desacuerdo y un 6.3 % de acuerdo. Estos resultados concuerdan con los de la afirmación "Creo que el Sistema de Coevaluación fue fácil de usar." donde un 56.3 % estuvo muy de acuerdo, un 12.5 % de acuerdo, un 18.8 % ni de acuerdo ni en desacuerdo y un 12.5 % en desacuerdo; en donde solo hubo un cambio de persona desde muy de acuerdo a en desacuerdo.

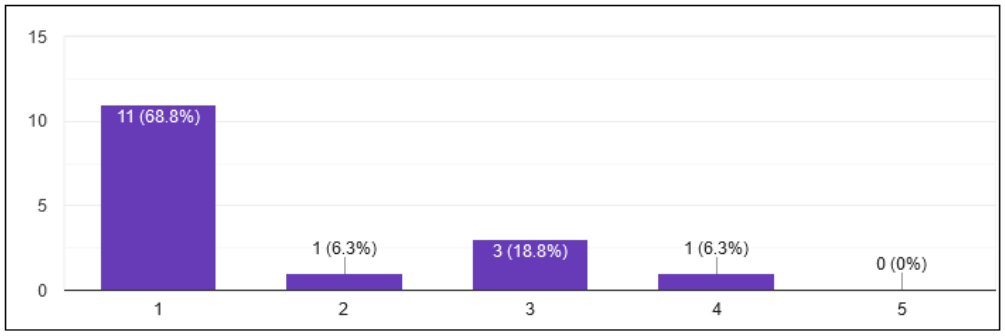



Figura 6.7: Validación final: Respuestas a afirmación "Encuentro este Sistema de Coevaluación innecesariamente complejo."

En general, todas las aseveraciones poseen una fija inclinación hacia respuestas positivas, siendo 68.8 % el mínimo porcentaje de votos a la opción positiva más extrema. El único claro cambio a esta regla es lo que ocurre con la afirmación "Me siento confiado al usar este Sistema de Coevaluación.", donde un 31.3 % responde que está muy de acuerdo, un 37.5 % que está de acuerdo y un 31.3 % se encuentra indeciso. Aunque en este punto la distribución de respuestas es más pareja, sigue la inclinación a las respuestas positivas. Para poder mejorar los resultados a esta pregunta e infundir mayor confianza al usuario, se plantea modificar los mensajes de las notificaciones para que fueran más directos y menos generales en su contenido (por ejemplo, al informar que una encuesta fue contestada, especificar en la notificación sobre quién era el

estudiante evaluado), agregar *tooltips* con información extra sobre funciones ocultas a simple vista (como los clicks en filas) y añadir documentación general del portal.

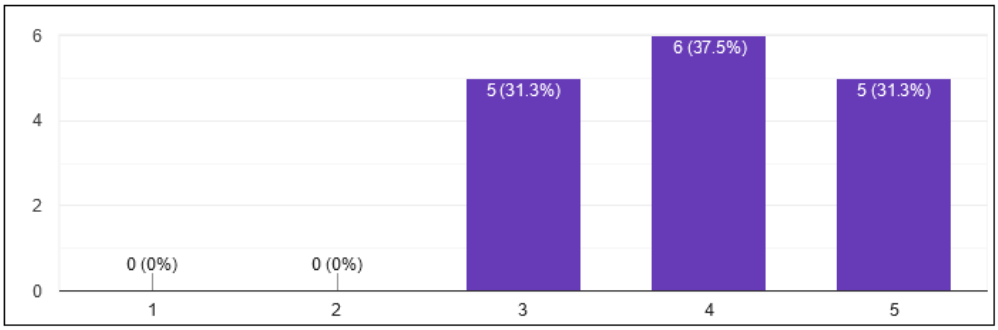



Figura 6.8: Validación final: Respuestas a afirmación "Me siento confiado al usar este Sistema de Coevaluación."

En el apartado para comentar, para la pregunta "¿Qué le parece este nuevo portal con respecto al Sistema de Coevaluación anterior (`http://coevaluaciones.dcc.uchile.cl`)?" existen 12 respuestas, la mayoría de ellas apuntando de forma positiva a la estética del sistema, como por ejemplo "Me parece un poco más sofisticado y limpio." y "(sic) Me resulto mejor esteticamente, pero no logre evidenciar una diferencia muy sustancial", pero tal como apunta este último comentario, no ven una gran diferencia en comparación al anterior, lo cual se puede analizar como algo positivo si se piensa que, si no encuentran diferencias en el proceso de responder una encuesta, el modelo mental ya creado por el sistema en producción se mantiene, suponiendo un cambio más fácil para aquellos usuarios que ya lo conocen.

En esta misma pregunta y con respecto a la complejidad del sistema, existen comentarios para encontrados, de forma positiva se expresa que "Tiene una mejor estructura y detalles sobre las coevaluaciones." y "(sic) Lo encontré mucho más sencillo que el anterior, pero le falto lo bonito que tenía el otro.", mientras que por el lado contrario,"El anterior se veía más amigable, era "más rápido" e intuitivo, este es un poco más complejo, no se necesita ser un experto para saber usarlo, pero si se le entrega a alguien que no sabe, se confundirá más que con el sistema anterior.". Considerando que la prueba de los usuarios alumno fue simplemente el responder una encuesta, se presume que el último comentario se refiere a este proceso, y como se maneja el envío de evaluaciones por separado de cada integrante del grupo versus una con todos. Este cambio en el modo de realizarlo fue una decisión de diseño para que el usuario tuviera mayor seguridad de a quién le está contestando la evaluación, pero se admite que el tener que cliquear uno a uno a cada compañero hace más lento y repetitivo el sistema. Para entender verdaderamente si este resulta ser un verdadero problema, se debería hacer una evaluación con usuarios sin experiencia en el sistema, y analizar como es su adaptación a él, por lo que por el momento no se pueden sacar conclusiones sobre este tópico con respecto a nuevos usuarios.

En línea con lo anterior van los comentarios dejados en el espacio opcional para ellos. Expresiones como "Este formulario es repetitivo en sus preguntas." y "Me gustó mucho que el sistema sea limpio y que para evaluar a cada integrante del grupo hay que seleccionarlo, sin embargo no me gustó el hecho de que se manden por separado (para cada integrante),

encuentro que es una complejidad innecesaria y que no es intuitivo" aportan a la hipótesis en la complejidad del sistema, diferente a lo que se había visto en la primera validación (capítulo 6.2).

También, debido a comentarios como "(sic) cuando hice la coevaluación tenía que poner nota de 1 a 5 pero no salía qué significaban. Asumí que 5 era la mejor nota y 1 la peor. Puede parecer obvio pero me dejó confundido un poco" y "(sic) Me pareció super fácil de usar, pero al ser la primera vez en a veces no estaba segura de que hacer por falta de explicación: no dice algo como responder coevaluación aquí, tampoco explica la escala si 5 es de acuerdo o no (asumí que sí)." se logró identificar la falencia de la falta de instrucciones, especialmente importante para usuarios nuevos del sistema, lo cual deberá ser corregido.

En el conjunto, esta encuesta deja la sensación de que, a pesar de que el sistema tiene buena evaluación SUS y en general comentarios positivos, aún queda por mejorar especialmente enfocando el diseño en usuarios nuevos del sistema, reconociéndose que hubo un cierto sesgo hacia usuarios ya experimentados, debido a que la misma autora de este sistema lo es y a que ninguna de las pruebas se hizo con usuarios que desconocieran completamente el proceso.

# Capítulo 7

# Conclusión

En esta memoria se detalló todo el proceso realizado para llegar a una solución al problema de mantención y usabilidad que posee el sistema de coevaluaciones existente en producción. Para realizarlo, se contaba con un proyecto antecesor, proveniente desde un trabajo de título realizado el año 2019 y que contaba con modelo de datos y algunas de las funcionalidades ya implementadas, y sobre esta construcción fue que se desarrolló el trabajo de memoria, analizándolo, corrigiéndolo y agregándole funcionalidades que permitieran llegar a una versión autónoma de un sistema de coevaluaciones.

Tal como demuestra esta memoria, se considera que tanto los objetivos generales y específicos planteados en un comienzo fueron cumplidos. Aún así, se admite que existen posibilidades de mejorar partes del programa, de tal forma de poder aumentar especialmente la robustez de este.

También, es bueno notar que se logró desarrollar un diseño en donde todas las funciones del sistema quedan a la vista, a diferencia de lo que podía pasar con el sistema en producción. De todas formas, se considera que el diseño podría mejorarse.

El hecho de recibir un sistema ya avanzado provocó que el desarrollo tuviera una cierta limitación en las decisiones a tomar. Esto produjo un aumento en los tiempos de adaptación al sistema, teniendo que además de averiguar sobre el problema en sí, tener que estudiar y conocer el trabajo realizado durante el trabajo dirigido, a lo que ayudó que este contara con documentación. Especialmente difícil se consideró realizar el diseño de las vistas del sistema, al tener que adecuarse a ideas externas, intentando crear un diseño lo mas cohesivo posible.

Otro hito que agregó dificultad a la realización de esta memoria fue el hecho de realizarla completamente de forma online. Se considera que hubo un gran aprendizaje en la forma de sobrellevar las cosas, especialmente durante las entrevistas de validación, donde se aprendió de formas de interactuar en una entrevista con personas por videollamada, ya que es necesario de alguna forma recordarles que están siendo escuchados e incentivarlos a que hablen.

En términos de validaciones, el sistema tuvo en general buen recibimiento, siendo evaluado con altos puntajes SUS y buenos comentarios. Además, todos los entrevistados que debían

realizar tareas con el sistema lograron completarlas.

Al finalizar el proceso, se cuenta con un sistema funcional para dos tipos de usuarios (alumno y profesor) y una alternativa provisional para un tercero. El sistema cuenta con todas las funcionalidades del sistema en producción y se logró implementar las nuevas funciones de autoevaluación, dinamismo de integrantes de grupos y dashboard, las cuales se considera que aumentan el valor del sistema.

## 7.1. Trabajo a futuro

A futuro, sería importante realizar un trabajo más detallado para el usuario Ayudante, ya que se considera como temporal la solución propuesta en esta memoria. Es importante que el usuario ayudante no posea los permisos de creaciones de cursos, ya que esto haría que inmediatamente se le asignara a él como profesor.

Siguiendo esta línea, sería interesante mejorar el portal del administrador, desarrollando algo que desencadene también la lógica que existe por detrás del sistema al, por ejemplo, crear ciertas entidades, lo cual actualmente si se hace desde el administrador, puede producir incongruencias con algunas reglas estipuladas y hacer que la lógica falle.

Entre las funcionalidades implementadas, sería bueno ajustar la función y vista para agregar nuevos integrantes al sistema, en especial cuando son agregados desde una plantilla. Ocurre que actualmente no se revisa si existen conflictos entre usuarios ya existentes en la base de datos, dando preferencia a la última información agregada, es decir, lo que viene desde la plantilla. En el código, la detección de conflictos ya está implementada, mas es necesario agregarlo al frontend.

También, se hace necesario el cambiar la forma en que se crean las autoevaluaciones, las cuales actualmente están siempre relacionadas a una coevaluación, sin la posibilidad de crearla en solitario.

Un tema interesante a abordar, sería la extensión del sistema a otros cursos, incluso pudiendo extenderlo a nivel universidad. Aunque el modelo de datos está pensado de esta forma y orientado hacia una posible expansión de cursos, se puede observar que existen ciertas relaciones que faltan para dimensiones y preguntas. Se esperaría que la unión desde preguntas a coevaluaciones sería directa, pero se cree que el modelo de dimensiones y el cálculo de las notas debería evaluarse con mayor detención, ya que existen cursos que podrían no necesitar dimensiones en sus evaluaciones y por esto, necesitan una versión más simple de lo actualmente implementado.

Además, se puede considerar este sistema como un instrumento para obtener variada información sobre los estudiantes y sus formas de evaluar, la cual podría llegarse a analizar y así encontrar información relevante sobre los alumnos, ya sea de forma individual y sus cambios durante el paso de la carrera, o de forma general, separando el análisis por grupos de interés según edad, género, historial académico, entre otros.

# Bibliografía


[1] Proyectos Agiles. Desarrollo iterativo e incremental. *https://proyectosagiles.org/desarrollo-iterativo-incremental/*, 10 Agosto, 2020.

[2] Nicolás Bevacqua. Dragula. *https://github.com/bevacqua/dragula*, Última visita: 17 Enero, 2021.

[3] Carolina Contreras. Validación coevaluación estudiantes. *https://docs.google.com/presentation/d/167Hs5lX4hpQ8gJhxee9umdJespnSffiaoEUUa8KkJV4/edit?usp=sharing*, 17 Enero, 2021.

[4] Escuela de Python. Django #1: introducción y el patrón mtv. *https://www.escuelapython.com/django-1-introduccion-patron-mtv/*, Última visita: 17 Enero, 2021.

[5] Desarrolloweb.com. Te explicamos de manera general el patrón de arquitectura del software mvc (model - view - controller o modelo - vista - controlador). cómo se separan las distintas capas atendiendo a sus responsabilidades. *https://desarrolloweb.com/articulos/que-es-mvc.html*, 28 Julio, 2020.

[6] MDN Web Docs. Django introduction. *https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/Introduction*, Última visita: 17 Enero, 2021.

[7] MDN Web Docs. Referencia dom de gecko: Introducción. *https://developer.mozilla.org/es/docs/Web/API/Document_Object_Model/Introduction*, Última visita: 28 Marzo, 2021.

[8] Python Everywhere. Host, run, and code python in the cloud! *https://www.pythonanywhere.com/*, 17 Enero, 2021.

[9] D. Galvez, R. López, and M Tomic. Sistema coevaluaciones. `https://github.com/milenkotomic/SistemaCoevaluaciones`, 10 Agosto, 2020.

[10] Editorial Team Geek insta. Difference between mvc and mvt architecture. *https://www.geekinsta.com/difference-between-mvc-and-mvt/*, 28 Diciembre, 2019.

[11] Django Girls. ¿qué es django? *https://tutorial.djangogirls.org/es/djan*

*go/*, Última visita: 17 Enero, 2021.

[12] Rinu Gour. Working structure of django mtv architecture. *https://towardsdatascience.com/working-structure-of-django-mtv-architecture-a741c8c64082*, 16 Abril, 2019.

[13] Nielsen Norman Group. 10 usability heuristics for user interface design. *https://www.nngroup.com/articles/ten-usability-heuristics/*, 10 Agosto, 2020.

[14] Grant McConnaughey. django-field-history. *https://django-field-history.readthedocs.io/en/latest/index.html*, 10 Agosto, 2020.

[15] David Merfield. randomcolor. a color generator for javascript. *https://randomcolor.lllllllllllllllll.com/*, Última visita: 17 Enero, 2021.

[16] Stack Overflow. 2020 developer survey. *https://insights.stackoverflow.com/survey/2020#technology-programming-scripting-and-markup-languages*, Última visita: 28 Marzo, 2021.

[17] Django Project. Signals: Model signals - m2m_changed. *https://docs.djangoproject.com/en/3.0/ref/signals/#m2m-changed*, 10 Agosto, 2020.

[18] Django Project. for... empty. *https://docs.djangoproject.com/en/3.1/ref/templates/builtins/#for-empty*, 17 Enero, 2021.

[19] Django Project. Generating aggregates over a queryset. *https://docs.djangoproject.com/en/3.1/ref/models/querysets/#get-or-create*, 17 Enero, 2021.

[20] Django Project. get_ot_create(). *https://docs.djangoproject.com/en/3.1/ref/models/querysets/#get-or-create*, 17 Enero, 2021.

[21] Pyexcel. pyexcel - let you focus on data, instead of file formats. *http://docs.pyexcel.org/en/latest/*, 07 Diciembre, 2020.

[22] Pyexcel. pyexcel-ods - let you focus on data, instead of ods format. *https://github.com/pyexcel/pyexcel-ods*, Última visita: 17 Enero, 2021.

[23] Pyexcel. pyexcel-xls - let you focus on data, instead of xls format. *https://github.com/pyexcel/pyexcel-xls*, Última visita: 17 Enero, 2021.

[24] Pyexcel. pyexcel-xlsx - let you focus on data, instead of xlsx format. *https://github.com/pyexcel/pyexcel-xlsx*, Última visita: 17 Enero, 2021.

[25] Roberto Riquelme. Sistema de evaluación del desempeño de los miembros de un equipo de desarrollo de software. *Memoria de Ingeniería Civil en Computación, Departamento de Ciencias de la Computación, FCFM, Universidad de Chile*, 2014.

[26] Jeff Sauro. Measuring usability with the system usability scale (sus). *https://measuringu.com/sus/*, Última visita: 28 Marzo, 2021.

[27] Shopify. Examples. *https://shopify.github.io/draggable/examples/*, Última visita: 17 Enero, 2021.

[28] Luis Silvestre. Diseño de equipos de desarrollo de software en escenarios universitarios. *Memoria de Ingeniería Civil en Computación, Departamento de Ciencias de la Computación, FCFM, Universidad de Chile*, 2012.

[29] Jocelyn Simmonds. Oferta de trabajo dirigido (x2). *https://www.u-cursos.cl/uchile/2008/0/COMCADCC/1/foro/o/2380369 8*, 22 Julio 2019.

[30] Stakoverflow. Django get_or_create fails to set field when used with iexact. *https://stackoverflow.com/questions/2929079/django-get-or-create-fails-to-set-field-when-used-with-iexact*, 26 Marzo, 2015.

[31] Sebastián Sánchez. Extensión de un sistema de coevaluación de miembros de equipos de desarrollo de software. *Memoria de Ingeniería Civil en Computación, Departamento de Ciencias de la Computación, FCFM, Universidad de Chile*, 2016.

[32] Nathan Thomas. How to use the system usability scale (sus) to evaluate the usability of your website. *https://usabilitygeek.com/how-to-use-the-system-usability-scale-sus-to-evaluate-the-usability-of-your-website/*, Última visita: 17 Enero, 2021.

[33] DCC UChile. Ingeniería civil en computación. objetivos. *https://www.dcc.uchile.cl/pregrado*, Última visita: 17 Enero, 2021.

[34] JQuery UI. Autocomplete. *https://jqueryui.com/autocomplete/*, Última visita: 17 Enero, 2021.

[35] JQuery UI. Sortable. *https://jqueryui.com/sortable/*, Última visita: 17 Enero, 2021.

[36] Usability.gov. System usability scale (sus). *https://usabilitygeek.com/how-to-use-the-system-usability-scale-sus-to-evaluate-the-usability-of-your-website/*, Última visita: 17 Enero, 2021.

[37] Klaas Van Schelven. django-simple-history. *https://django-simple-history.readthedocs.io/en/latest/index.html*, 10 Agosto, 2020.

[38] Ibis Álvarez. La coevaluación como alternativa para mejorar la calidad del aprendizaje de los estudiantes universitarios: valoración de una experiencia. *Revista Interuniversitaria de Formación del Profesorado*, 2008.

# Apéndices

# Apéndice A

# Lista de preguntas encuesta a estudiantes sobre sistema en producción

A continuación se muestran todas las preguntas presentadas en la encuesta realizada a alumnos del DCC sobre el sistema en producción, referenciada en la sección 3.2.1.3. Desde la pregunta 3 en adelante, se le pidió al estudiante responder considerando su última experiencia con el sistema.

1. ¿En qué cursos ocupó el Sistema de Coevaluación?: Opción múltiple
   - CC4401 - Ingeniería de Software I.
   - CC5401 - Ingeniería de Software II.
   - CC5402 - Proyecto de Software.
2. ¿Cuándo usó el Sistema de Coevaluación por última vez?: Elegir una opción:
   - Otoño 2019.
   - Primavera 2019.
   - Otoño 2020.
   - Otro (escribir cual).
3. ¿Qué tan satisfecho quedó al usar el sistema?: Calificar entre 1 (completamente insatisfecho) a 5(completamente satisfecho).
4. ¿Qué tan fácil le resultó responder una coevaluación?: Calificar entre 1 (extremadamente fácil) a 5 (extremadamente difícil).
5. Los resultados de una coevaluación son agrupados en dimensiones por el sistema, como por ejemplo actitud, colaboración y comunicación. ¿Ha notado esta división en el sistema?: Sí o no.
6. ¿Ha podido hacer seguimiento de la evolución de sus resultados? Es decir, poder comparar su rendimiento entre distintas coevaluaciones: Sí o no.
7. ¿Cómo ha hecho seguimiento de la evolución de sus resultados?: Pregunta abierta.
8. ¿Qué aspectos del sistema le gustan?: Opción múltiple.
   - Diseño.
   - Notificaciones.

- Seguimiento de avance.
- Facilidad de uso.
- Otros (escribir cual).

9. ¿Qué aspectos podría mejorar el sistema?: Mismas opciones de la pregunta anterior.
10. Con respecto al Sistema de Coevaluación, como evaluaría los siguientes aspectos: Evaluar 1 a 5 o "no aplica", con 1 siendo completamente insatisfecho y 5 siendo completamente satisfecho, en los tópicos:
    - Visibilidad del estado de una coevaluación (abierta, cerrada)
    - Claridad y consistencia del sistema (lenguaje, colores, íconos, etc.)
    - Facilidad para contestar una coevaluación
    - Facilidad para editar las respuestas de una coevaluación
    - Diseño estético y minimalista
11. Con respecto a la página de inicio del sistema la considera: Calificar de 1 (completamente insatisfecho) a 5 (completamente satisfecho) en cada una de los siguientes tópicos:
    - Comprensible.
    - Amigable.
    - Atractiva.
    - Útil.
12. ¿Qué te gustó de la interfaz anterior?: Pregunta abierta.
13. ¿Qué NO te gustó de la interfaz anterior?: Pregunta abierta.
14. ¿Alguna vez ha revisado sus resultados en a través del sistema de coevaluación?: Sí o no.
15. En caso de la respuesta ser no, ¿Por qué no revisó los resultados en el Sistema de Coevaluaciones?: Opción múltiple
    - No estaba interesado.
    - No conocía la funcionalidad.
    - Veo el resultado por U-Cursos.
    - Veo el resultado por la notificación que llega a mi correo.
    - Otro (escribir cual).
16. Sobre la interfaz de resultados, la considera: Calificar de 1 (completamente insatisfecho) a 5 (completamente satisfecho) en cada una de los siguientes tópicos:
    - Comprensible.
    - Amigable.
    - Atractiva.
    - Útil.
17. ¿Le gustaría poder ver las notas de todos sus ramos (Ingeniería de Software I, Ingeniería de Software II y Proyecto de Software) en una sola intefaz?: Sí o no.
18. ¿Algún comentario adicional sobre la interfaz de resultados de una coevaluación?: Pregunta abierta.
19. ¿Alguna vez ha visto el resumen de sus notas?: Sí o no.

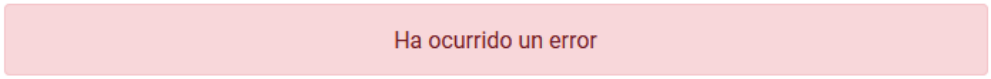
20. En caso de la respuesta ser no, ¿Por qué no revisó el resumen de notas que entrega el sistema de coevaluaciones?: Opción múltiple
    - No estaba interesado.
    - No conocía la funcionalidad.
    - Otro (escribir cual).
21. Sobre la interfaz de resumen de notas, la considera: Calificar de 1 (completamente insatisfecho) a 5 (completamente satisfecho) en cada una de los siguientes tópicos:
    - Comprensible.
    - Amigable.
    - Atractiva.
    - Útil.
22. Con respecto a los gráficos del resumen de notas, ¿los considera adecuados?: Sí o no.
23. ¿Algún comentario adicional sobre la interfaz de resumen de notas?: Pregunta abierta.
24. ¿Algún comentario adicional sobre el Sistema de Coevaluación? Pregunta abierta.

# Apéndice B

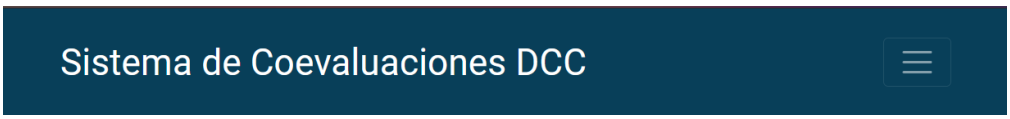
# Lista completa de mejoras para sistema trabajo dirigido

A continuación se muestra en extenso los bugs, funcionalidades faltantes y mejoras necesarias para el sistema del trabajo dirigido.

- **Inicio de Sesión**
  - Iniciar sesión con estudiante sin coevaluaciones: Error porque no existe una coevaluación asociada a él.
  - Iniciar sesión con usuario que no es integrante de nada: Error porque no existe coevaluación asociada a él.
  - Si intenta entrar al portal un usuario no reconocido o con una contraseña incorrecta entrega un mensaje de error que podría ser más claro:




  - En pantallas pequeñas se agrega un botón de menú, pero que no tiene opciones dentro de él:

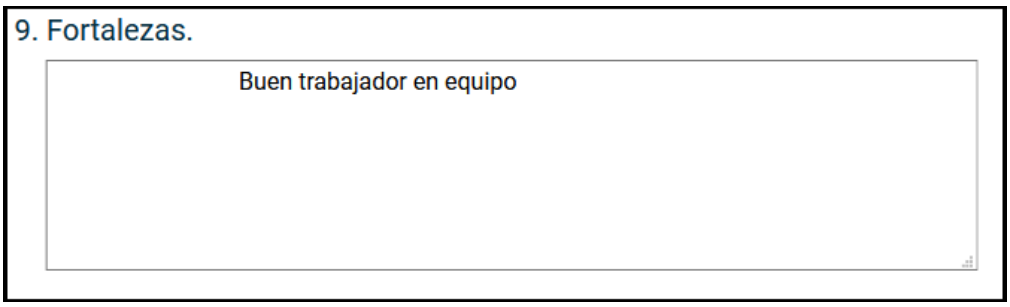



- **Home Page Profesor** (Figura 2.6)
  - Función de agregar cursos no crea nuevos cursos.
  - Agregar coevaluación no maneja la opción de que la coevaluación sea a la vez autoevaluación.
  - En la tabla de "últimas coevaluaciones" la columna para "Detalle" está cortada, pudiendo reemplazarse con un ícono. La columna "Curso" no tiene sus elementos alineados.

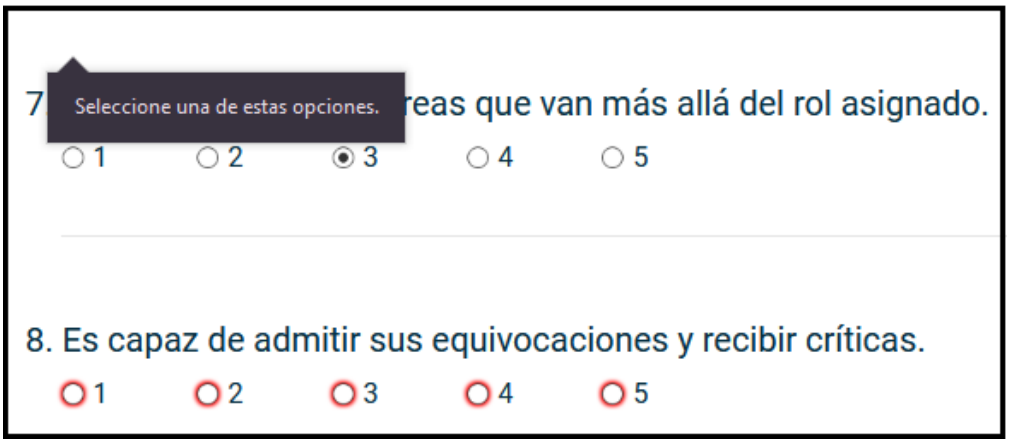

  - Hay que implementar la búsqueda de estudiantes.

- **Detalle de una Coevaluación para Profesor** (Figura 2.7)
  - Hay que implementar las funcionalidades de todos los botones.
  - Detalles relacionados al diseño: En el navbar alinear el nombre del usuario y el botón de "Cerrar sesión", arreglar el botón de "Exportar resultados".
  - Agregar edición del estado de la coevaluación.
  - Error "*unsupported operand type(s) for +=: 'decimal.Decimal' and 'float'*" al calcular la nota de los estudiantes.
  - Se está considerando que una persona contestó todas las coevaluaciones cuando contesta al menos una.

- **Home Page Alumno** (Figura 2.8)
  - La página inicial de alumno es una coevaluación específica, sin opción para ver las demás.
  - Si se intenta ingresar al sistema con un usuario Alumno que no tiene coevaluaciones asociadas a él el sistema levanta un error.
  - El expandir coevaluación a través del click a integrantes de grupos no funciona, se debe forzar la redirección hacia `http://localhost:8000/coevaluacion?coev=<id_coev>&evaluado=<id_evaluado>#responder-coeval`. Además, el sistema deja que se responda la coevaluación para cualquier usuario, incluyendo el profesor y alumnos que no pertenecen ni a su equipo ni a su curso.
  - Detalles de diseño: En el navbar no aparece el nombre del usuario, al responder una coevaluación, esta se identifica con el usuario del evaluado (que será el rut), no con su nombre y apellido.
  - La barra de búsqueda de estudiantes presente en el navbar no es de utilidad para los alumnos.
  - Al volver a editar una coevaluación ya respondida el sistema no muestra las respuestas del checklist (las primeras 8 preguntas), y para las respuestas abiertas se agregan espacios en blanco.




– El sistema sí avisa cuando falta por llenar un campo obligatorio, pero cuando se mueve la página el mensaje se mueve también.




- **Bugs Generales**
  - Hay que restringir tanto los alumnos que pueden pertenecer a un grupo como las coevaluaciones que se les asignan. Todos deben pertenecer al mismo curso.
  - Al cerrar la sesión se puede volver a atrás y ver la sesión abierta, e incluso hacer cosas.
  - Se puede crear una coevaluación desde el usuario Profesor, pero esta no es asignada a los grupos del curso.
  - La página web no tiene un diseño adaptativo (responsive).

# Apéndice C

# Mockups de diseño de solución propuesta

A continuación se muestran los mockups que no fueron incluidos en la sección 4.2.1

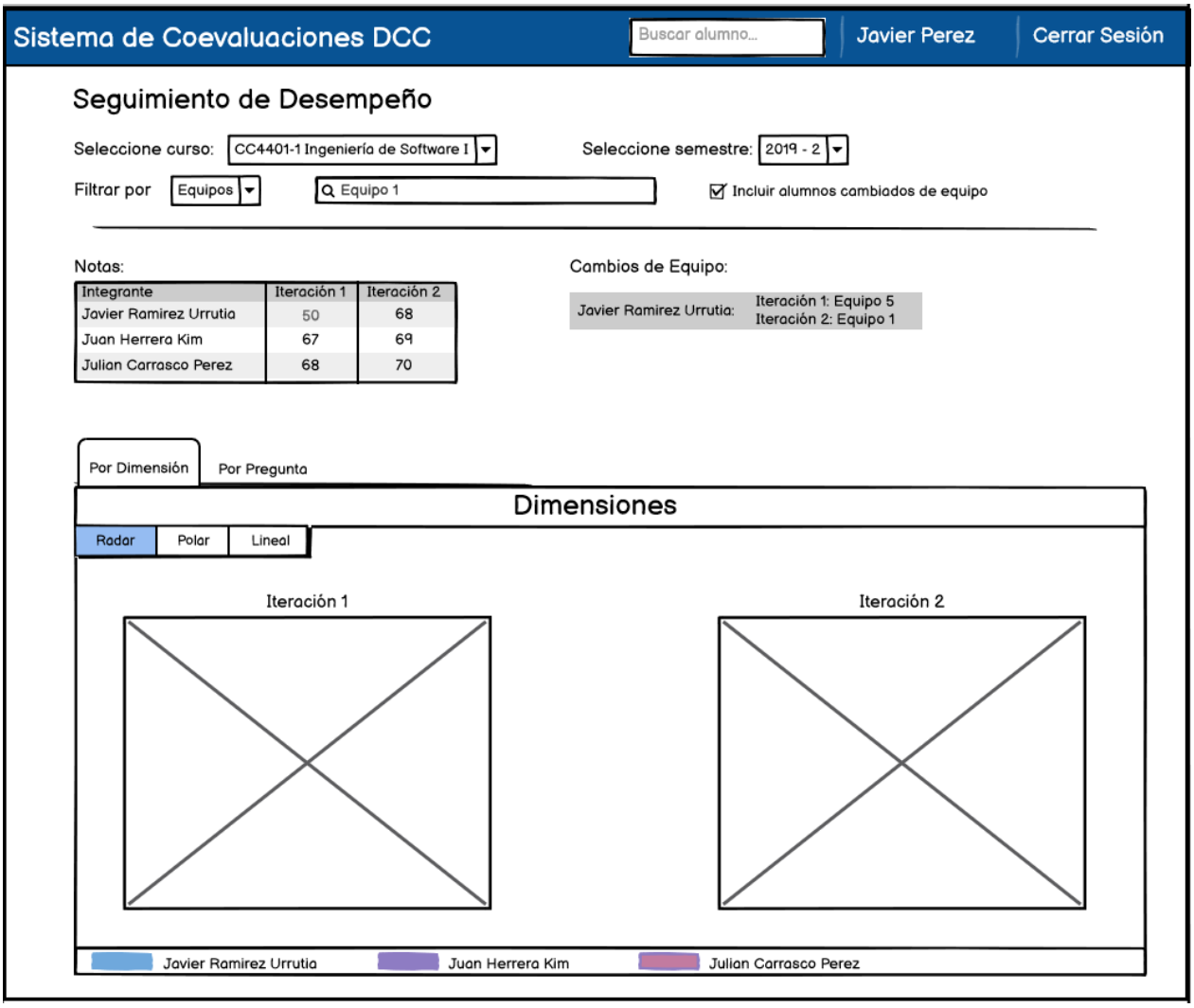



Figura C.2: Seguimiento de equipo

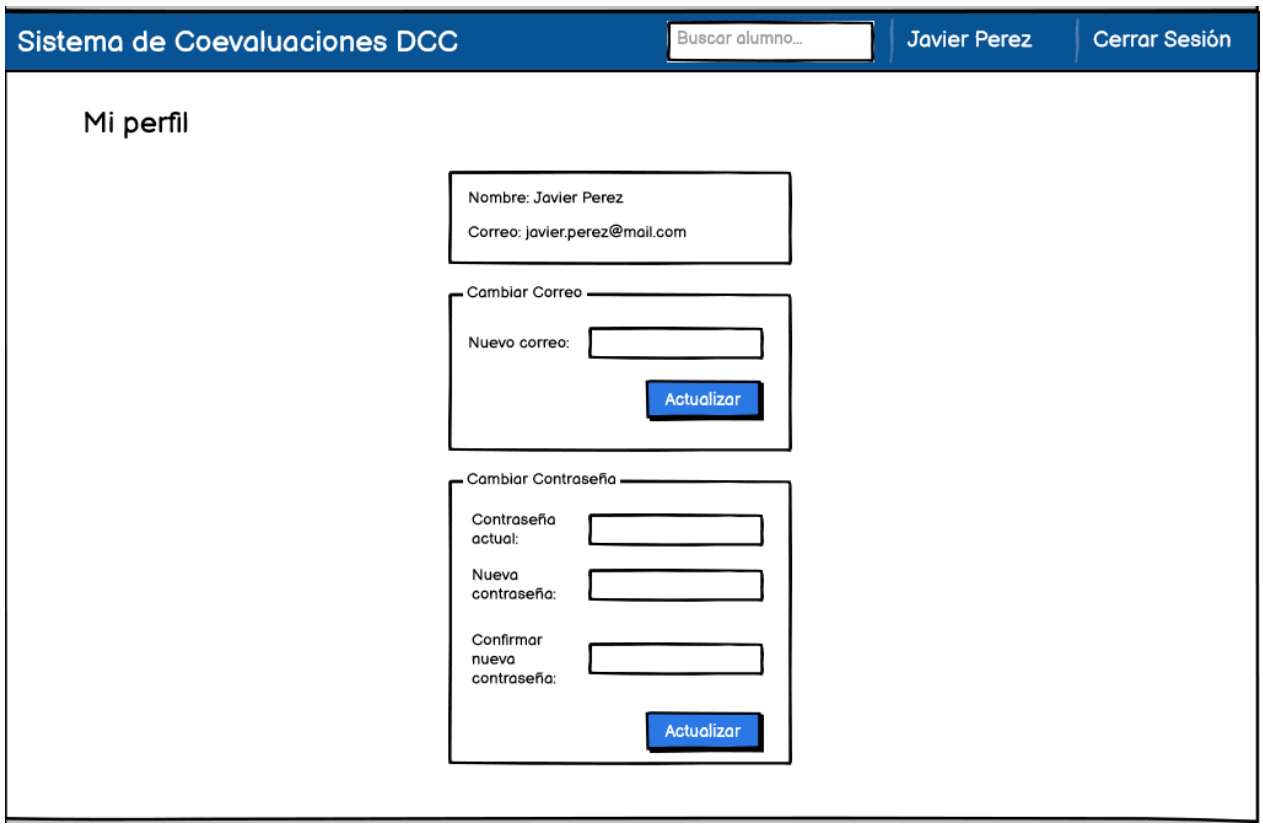



Figura C.1: Perfil profesor

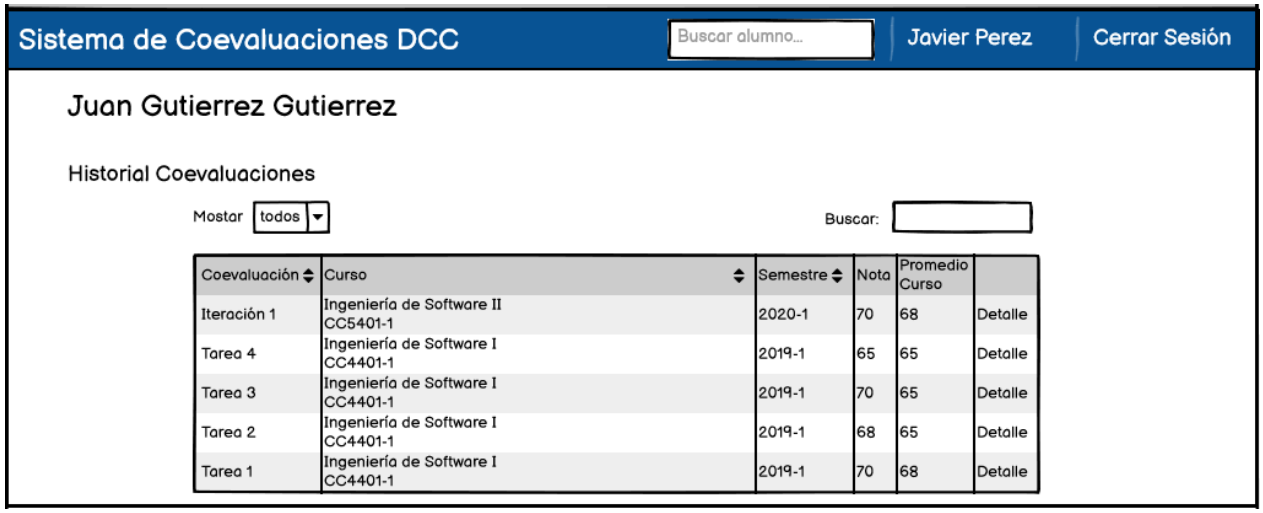



Figura C.4: Historial de coevaluaciones alumno

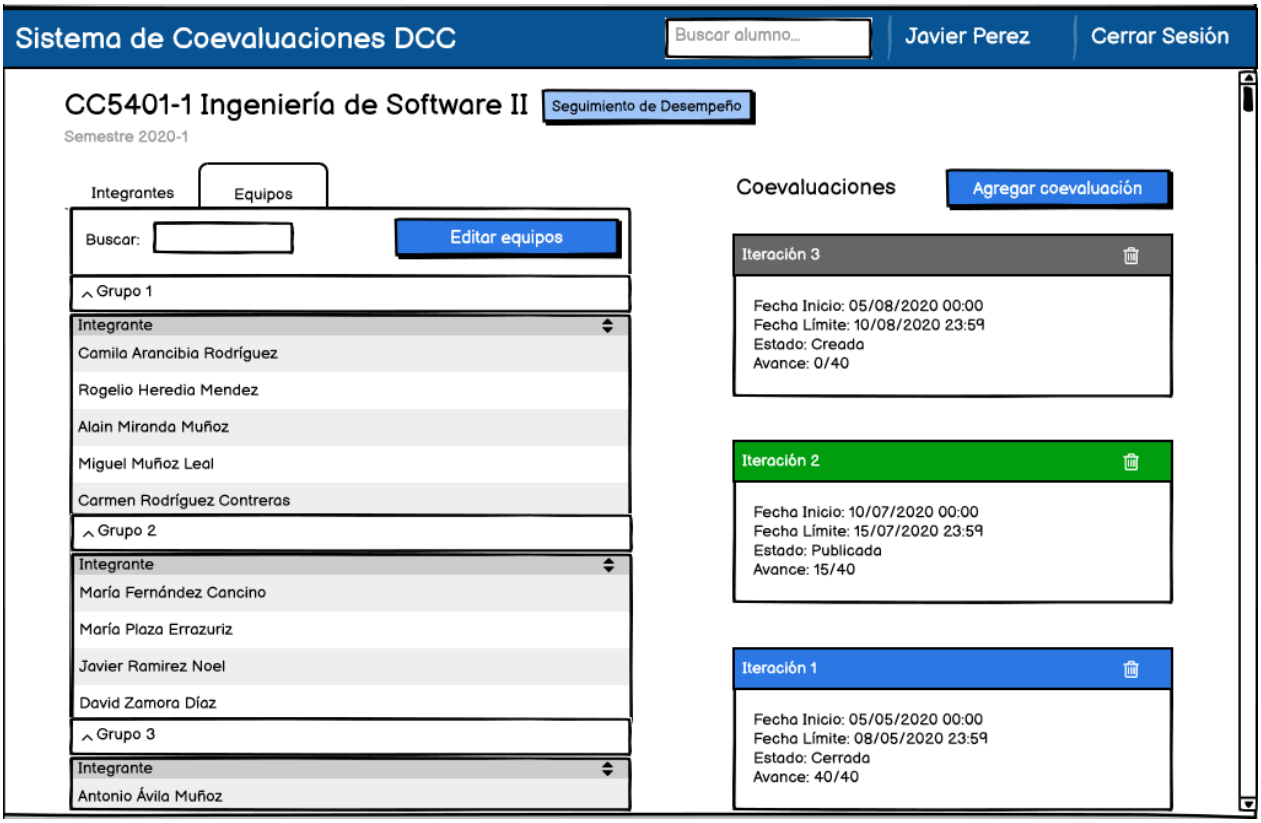
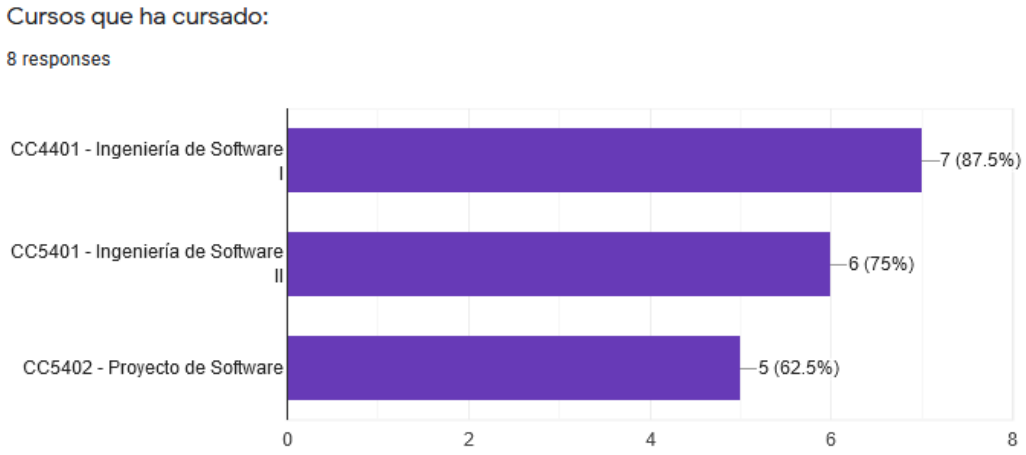


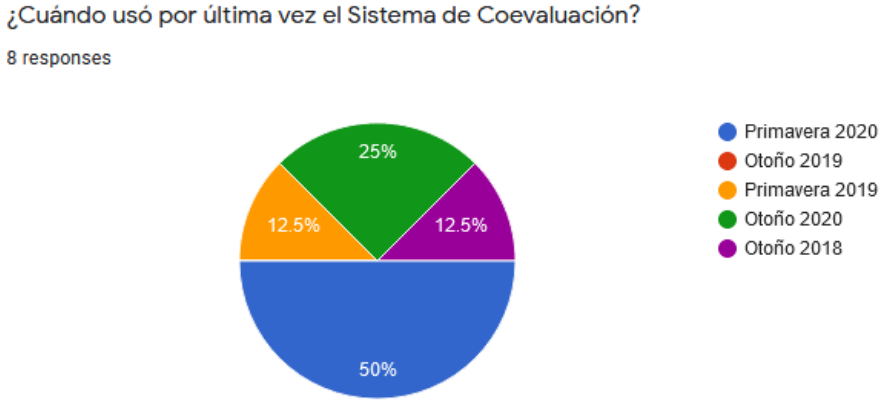

Figura C.3: Detalle curso - Lista equipos

# Apéndice D

# Ficha de resultados primera validación de respuestas a coevaluaciones

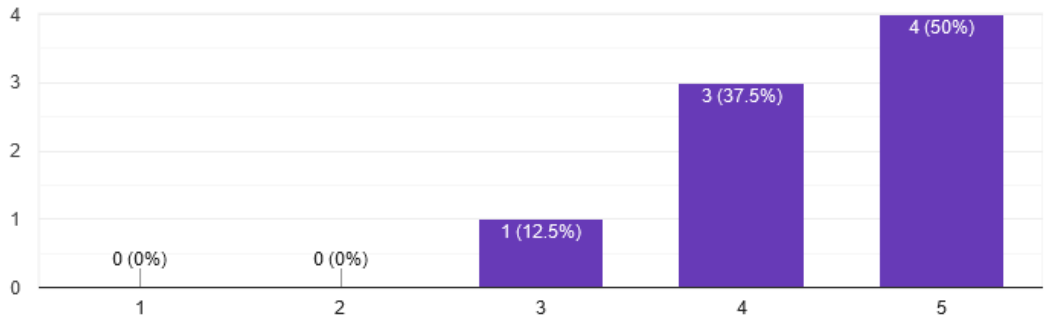
En esta sección se muestran las preguntas y los resultados en gráficos de la primera validación con alumnos sobre la funcionalidad de respuesta a coevaluaciones. Esto es mencionado en la sección 6.2.



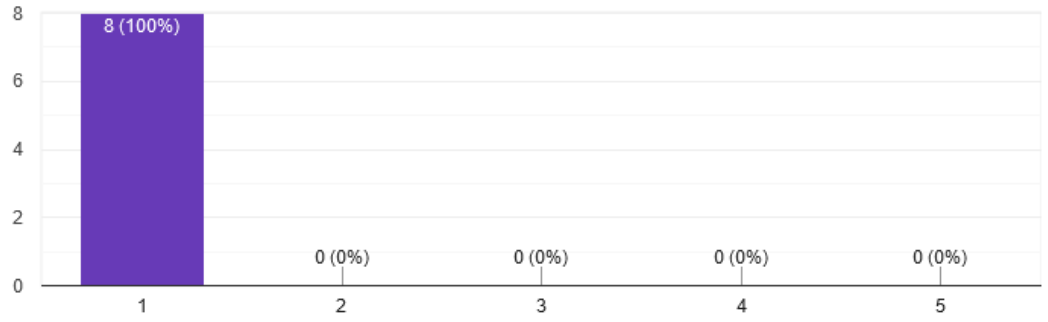
1. Creo que usaría este Sistema de Coevaluación frecuentemente.

8 responses



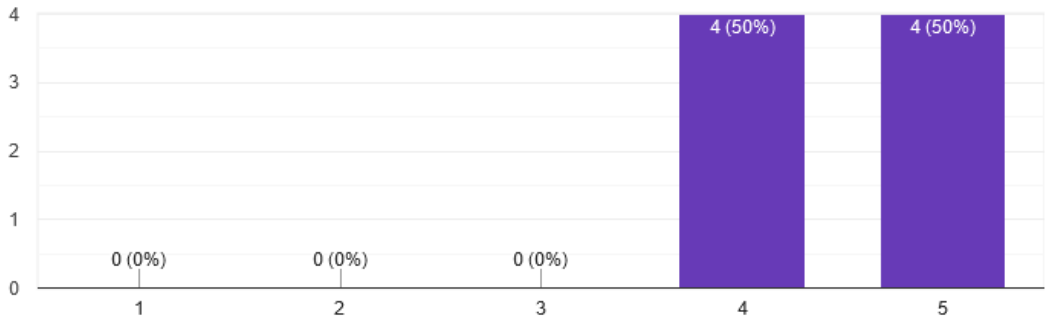

2. Encuentro este Sistema de Coevaluación innecesariamente complejo.

8 responses



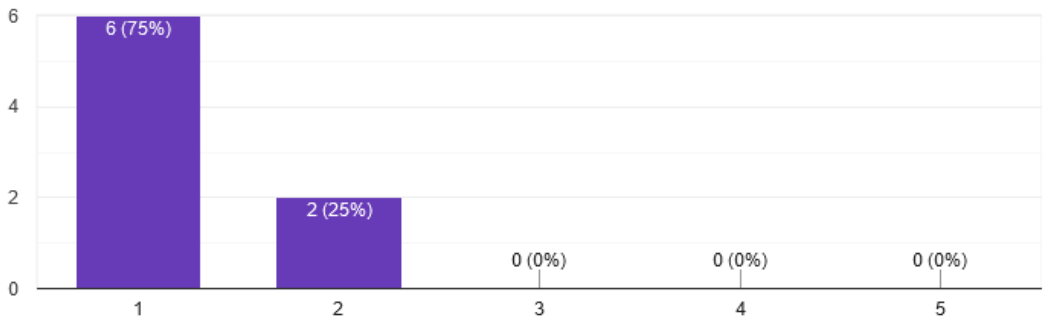

3. Creo que el Sistema de Coevaluación fue fácil de usar.

8 responses



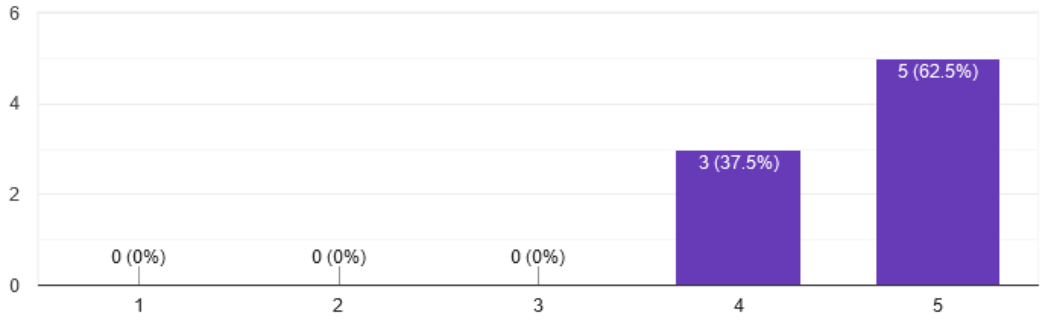
4. Creo que necesitaría ayuda de una persona con conocimientos técnicos parausar este Sistema de Coevaluación.

8 responses



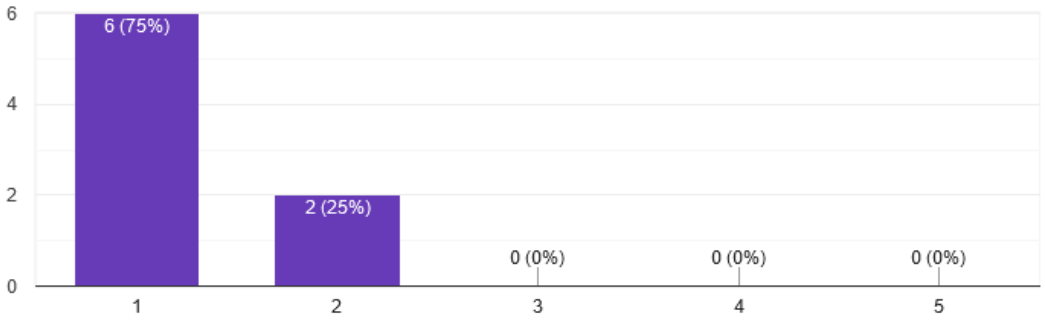

5. Las funciones de este Sistema de Coevaluación están bien integradas.

8 responses



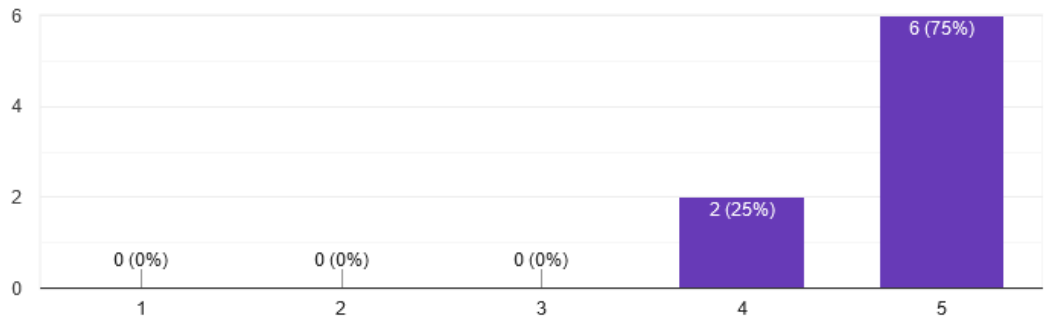

6. Creo que el Sistema de Coevaluación es muy inconsistente.

8 responses



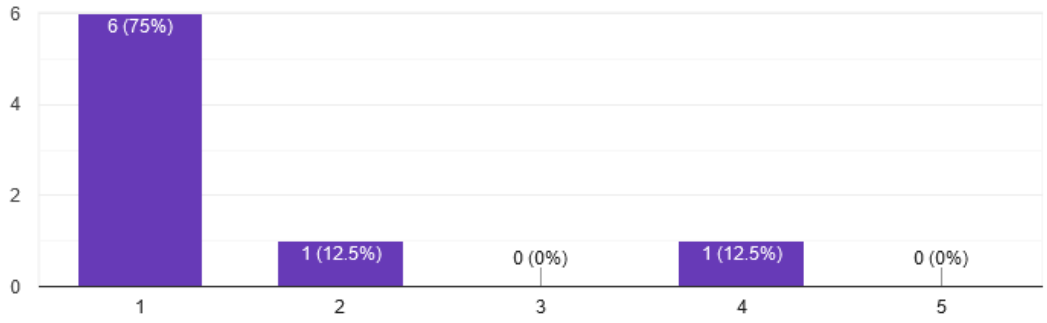
7. Imagino que la mayoría de la gente aprendería a usar este Sistema de Coevaluación en forma muy rápida.

8 responses



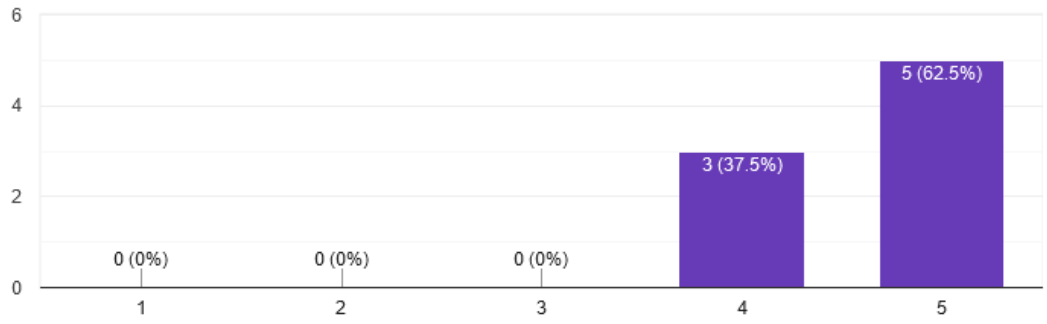

8. Encuentro que el Sistema de Coevaluación es muy difícil de usar.

8 responses



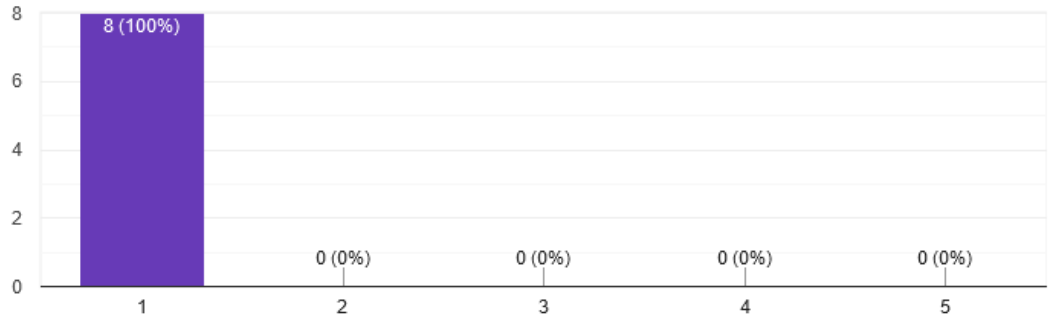

9. Me siento confiado al usar este Sistema de Coevaluación.

8 responses



10. Necesité aprender muchas cosas antes de ser capaz de usar este Sistema de Coevaluación.

8 responses



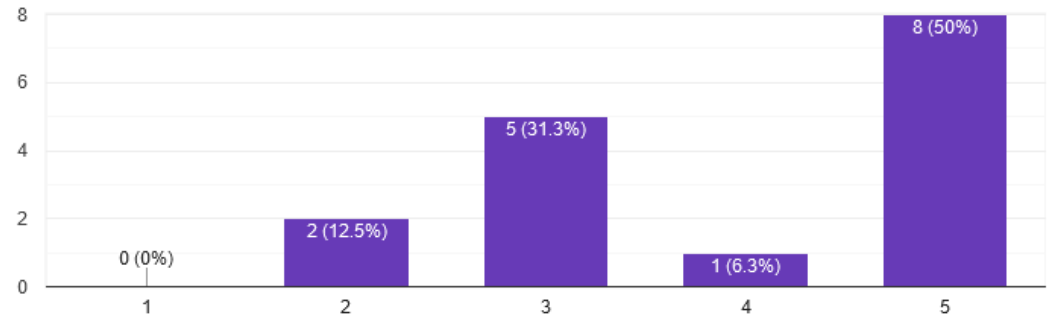
# Apéndice E

# Ficha de resultados validación final con alumnos del curso CC4401

A continuación se muestran las preguntas y sus resultados de la encuesta aplicada a los alumnos de CC4401 del semestre Primavera 2020. En el informe se referencia esta parte en la sección 6.4

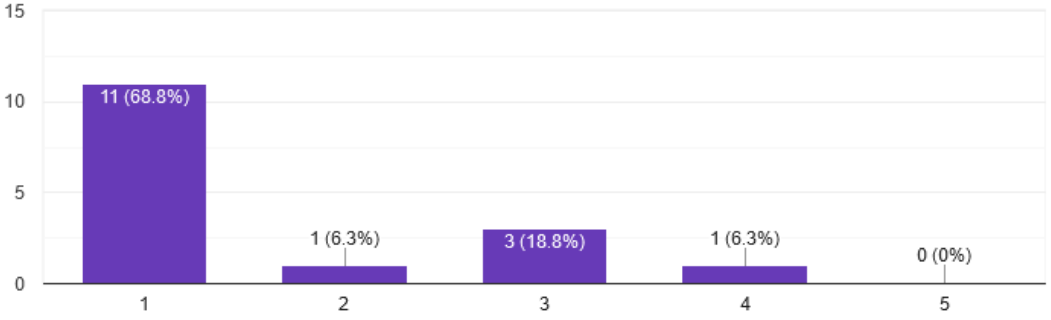
1. Creo que usaría este Sistema de Coevaluación frecuentemente.

16 responses



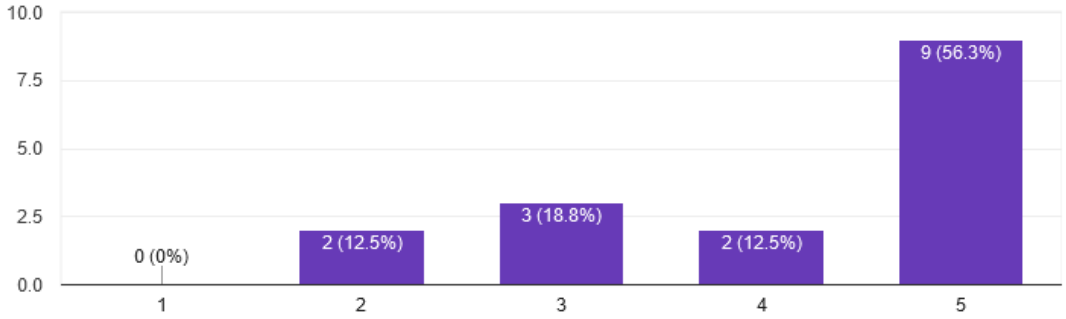

2. Encuentro este Sistema de Coevaluación innecesariamente complejo.

16 responses



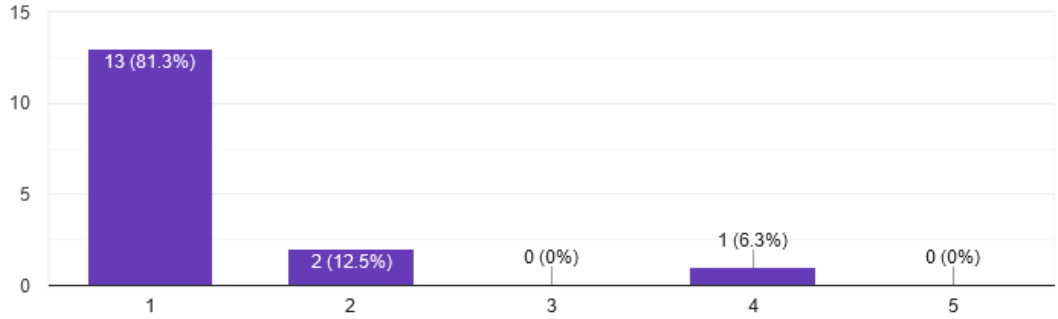
3. Creo que el Sistema de Coevaluación fue fácil de usar.

16 responses



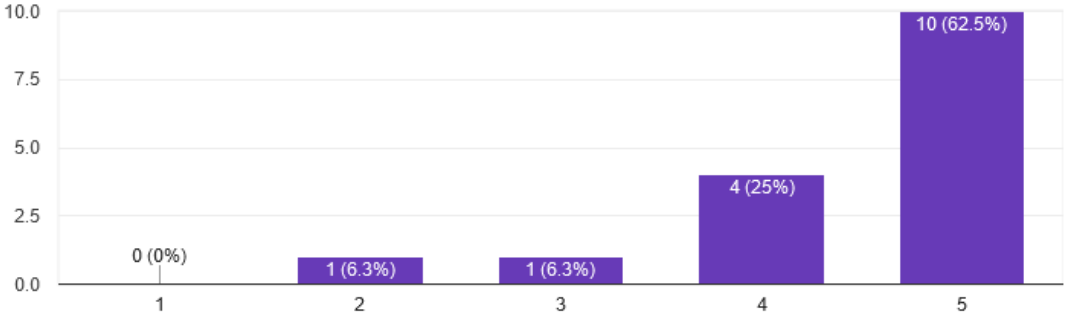

4. Creo que necesitaría ayuda de una persona con conocimientos técnicos parausar este Sistema de Coevaluación.

16 responses



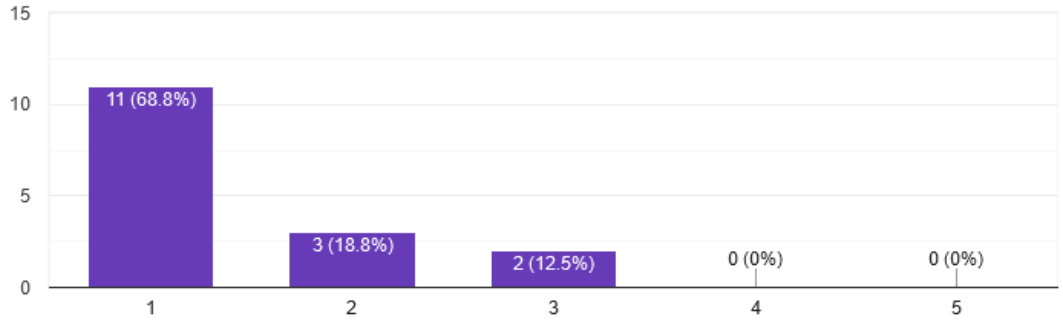

5. Las funciones de este Sistema de Coevaluación están bien integradas.

16 responses



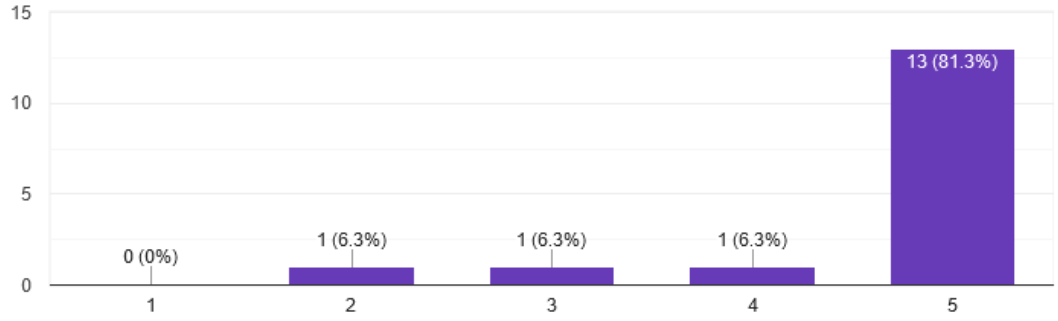
6. Creo que el Sistema de Coevaluación es muy inconsistente.

16 responses



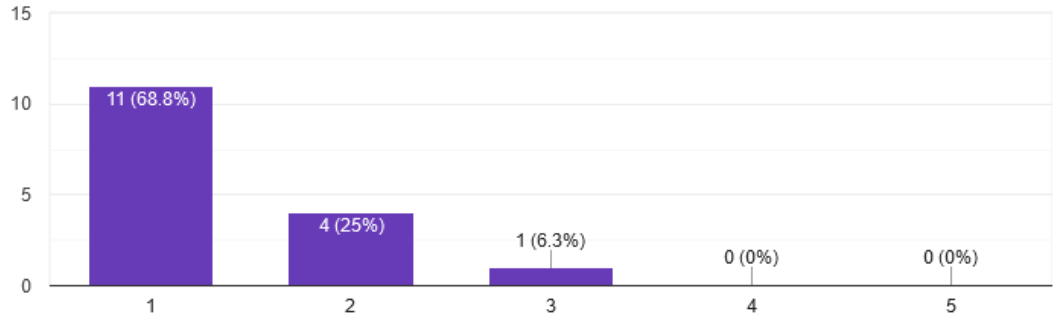

7. Imagino que la mayoría de la gente aprendería a usar este Sistema de Coevaluación en forma muy rápida.

16 responses



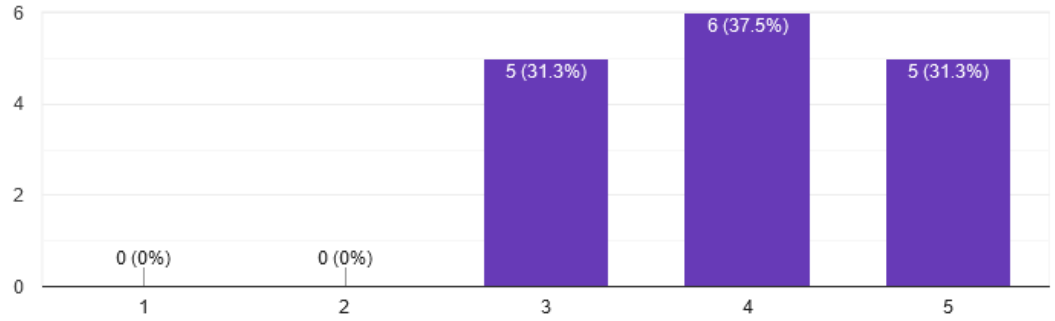

8. Encuentro que el Sistema de Coevaluación es muy difícil de usar.

16 responses



9. Me siento confiado al usar este Sistema de Coevaluación.

16 responses




10. Necesité aprender muchas cosas antes de ser capaz de usar este Sistema de Coevaluación.

16 responses