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

REESTRUCTURACIÓN Y DESARROLLO DEL SISTEMA DE GESTIÓN DE BECAS Y TRABAJOS PARA LATINITY

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERA CIVIL EN COMPUTACIÓN

CAMILA SOLEDAD QUILAPE PAILLAL

PROFESOR GUÍA:
JOCELYN SIMMONDS WAGEMANN

MIEMBROS DE LA COMISIÓN:
JUAN ÁLVAREZ RUBIO
PABLO GONZÁLEZ JURE

SANTIAGO DE CHILE
2022

Resumen

Latinity es una conferencia enfocada en promover la participación femenina en computación y tecnología, que se desarrolla una vez al año en algún país latinoamericano. Durante la conferencia se exponen diferentes presentaciones, pósters y círculos de mentoría, conocidos en conjunto como trabajos. Tanto los trabajos como las becas de asistencia se seleccionan a través de procesos de postulación.

Actualmente la administración de estos procesos se realiza mediante formularios y planillas de Google, sin embargo a lo largo de los últimos años la cantidad de postulaciones ha incrementado considerablemente, provocando dificultades y demoras en la toma de decisiones. De aquí surge la necesidad de crear un sistema a la medida, que facilite la gestión de los procesos de postulación a becas y trabajos de la conferencia.

El software comenzó a desarrollarse el 2019 y ha pasado por tres equipos del ramo Ingeniería de Software 2, los que avanzaron en el módulo de postulación a becas, el módulo de postulación a trabajos y el módulo de revisión de trabajos. No obstante, estos módulos estaban incompletos y no se relacionaban bien entre ellos, razones que hacían indispensable continuar con el trabajo.

El problema abordado, por lo tanto, fue extender lo desarrollado, lo cual abarca a grandes rasgos, solucionar problemas, reestructurar secciones y añadir implementaciones faltantes. Así como también unificar desarrollos y mejorar el diseño del front-end, con el fin de alcanzar un producto usable y unificado.

La solución propuesta consistió en implementar un conjunto de funcionalidades, como la reestructuración del módulo de becas y la incorporación de la revisión de becas. Por otra parte, incluye la carga de revisores mediante un archivo csv, y la edición y desactivación de revisores. Asimismo se agregaron resúmenes de apoyo que visibilizan cifras relevantes para los administradores.

Teniendo en cuenta el objetivo general, que resumidamente es “reestructurar y completar el sistema de gestión de Latinity con el propósito de lograr por primera vez un software usable, funcional y completo”. Junto al plan de trabajo y las validaciones realizadas, muestran que se alcanzó el producto mínimo viable, debido a que se cumplieron todos los objetivos y se obtuvieron buenos resultados de las pruebas de usuario.

Dedicado a mis papás, a quiénes les debo todo.

Agradecimientos

A mis amigas del liceo, Javiera, Katherine y Steffi por apoyarme y preocuparse por mi.

A Andrea y Carolina, por su compañía en las alegrías, frustraciones y penas. Los almuerzos juntas son de las cosas que más atesoro y extraño.

A Raúl, por estar cerca este último tiempo y darme ánimos para terminar esta etapa.

A la profesora Jocelyn, por su ayuda y buena disposición. Estoy muy agradecida.

A todas las personas que conocí en la universidad y que de algún modo u otro fueron un aporte a mi vida, sobre todo a Andrés, Bryan, Felipe, Renata y Rocío.

A mis abuelos, tíos y primos, por animarme a seguir adelante.

A mis papás José y Lidia, por darme las facilidades y herramientas para llegar hasta aquí y por motivarme siempre.

Por último, a mi hermano Víctor, por todas las risas y momentos de distracción.

Tabla de Contenido

Índice de Tablas

Índice de Ilustraciones

Índice de Códigos

Capítulo 1

Introducción

1.1. Contexto

Latinity, acrónimo de Latin American Women in Technology, es una conferencia en torno a computación y tecnología orientada a mujeres, que se realiza una vez al año en algún país de Latinoamérica. Surge el año 2015, con el objetivo de fomentar la participación y difusión de estas disciplinas en la comunidad femenina latinoamericana.

Durante la conferencia, cuya duración es de dos días, se presentan diversas charlas, pósters y círculos de mentoría, conocidos en conjunto como trabajos. Mujeres destacadas en el mundo de la computación relatan sus experiencias, con el propósito de motivar a los asistentes a cumplir sus sueños. Académicas exponen los proyectos que han realizado o en los que están trabajando. Asimismo se generan discusiones sobre temas de género y sociedad, tales como deserción estudiantil, compatibilidad de responsabilidades, etc.

Los trabajos se seleccionan mediante un proceso de postulación, a partir de una convocatoria abierta a estudiantes, profesionales y académicos, que considera en la evaluación aspectos como relevancia del tema y originalidad. Por otro lado, existe igualmente un proceso de postulación a becas de apoyo para asistir a Latinity, que buscan favorecer a quienes perciban un mayor beneficio o impacto. Desde otro punto de vista, la conferencia también involucra un proceso de gestión interna, que incluye la definición de plazos, la distribución y revisión de postulaciones, y la toma de decisiones finales, es decir, si una postulación se acepta o rechaza.

A lo largo de los últimos años las postulaciones mencionadas se han estado realizando a través de formularios de Google. Se habilitan por un cierto periodo de tiempo y permiten, tanto a los interesados en presentar un trabajo como a los solicitantes de becas, ingresar postulaciones. Los formularios solicitan lo necesario para contactar a los postulantes y conocer el contexto del trabajo o situación, en el caso de becas.

Una vez terminado el plazo, se asignan revisores a postulaciones manualmente, considerando dominio del área e idioma. Luego, se recopilan los resultados de esas revisiones en una planilla compartida entre los revisores. Finalmente, el equipo organizador se reúne a decidir, en base a las revisiones más otros criterios, qué postulaciones aceptar, notificando los veredictos mediante correo electrónico.

1.2. Problema y relevancia

Lo anterior funcionó bien en un inicio, dado el bajo número de postulaciones, aproximadamente 100 para trabajos y 200 para becas. Sin embargo últimamente la conferencia ha ganado popularidad, provocando que la cantidad de postulaciones haya crecido considerablemente. En la última versión del año 2019 se recibieron cerca de 300 postulaciones a trabajos y 400 postulaciones a becas. Por lo que, se hizo indispensable un sistema que facilitara la gestión y comunicación de las postulaciones a becas y trabajos, disminuyendo el trabajo manual y reduciendo las probabilidades de cometer errores.

Antes de tomar la decisión de efectivamente desarrollar un software, se estudiaron diferentes alternativas, entre las que destacan las plataformas EasyChair, HotCRP, PaperCall y el software de Anita Borg para Grace Hopper Celebration. Cada plataforma posee ventajas y desventajas, las que se detallarán en el capítulo Estado del Arte, pero principalmente se descartaron por tener solo funcionalidades para trabajos, los costos y las curvas de entrada en cuanto a usabilidad.

El sistema comenzó a llevarse a cabo a mitad del año 2019 y fue desarrollado por tres equipos del curso Ingeniería de Software 2, los cuales avanzaron en la implementación de diferentes módulos. En específico, el equipo Primavera 2019 [1] se encargó de postulaciones a becas, el equipo Otoño 2020 [2] de postulaciones a trabajos y el equipo Primavera 2020 [3] de revisión de trabajos. La estudiante formó parte de este último equipo.

Como consecuencia, el software estaba separado en tres partes que no se relacionaban como debían, ni eran usables, ya que estaban incompletas. Por ejemplo, el módulo de postulaciones a becas estaba completamente aislado de los módulos de postulación y revisión de trabajos. Cada una de estas partes poseía decisiones de modelos de datos, flujos lógicos, diseño de interfaces y secciones de administración diferentes, lo que dificultaba conectarlas. Debido a esto surge la necesidad de continuar con el desarrollo, implementando las funcionalidades faltantes, corrigiendo las que no andan bien y unificando componentes.

El trabajo contempla extender lo desarrollado, lo cual consiste en solucionar los problemas de validación y navegación de los módulos de trabajos y revisión de trabajos, reestructurar por completo el módulo de becas, ajustando modelos y diseño, y añadir la parte de revisión de becas. También unificar las secciones de administración y mejorar el diseño del front-end.

1.3. Objetivos

El objetivo general fue reestructurar y completar el sistema de gestión de Latinity, con el propósito de lograr por primera vez un software usable, funcional y completo, que facilitara la administración de todas las tareas relacionadas a las postulaciones a becas y trabajos. Esto contemplaba la integración de funcionalidades presentes con otras corregidas y nuevas.

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

1. Analizar el sistema, a nivel de código, con el fin de entender cómo estaba diseñado e implementado. También para determinar si reestructurar el módulo de becas modificándolo o reimplementándolo, dado que el recibido no satisfacía las necesidades de la conferencia.
2. Reestructurar el módulo de becas, comenzando con el diseño de un nuevo modelo de datos.
3. Uniformar las interfaces, que eran diferentes en cada módulo e incluso dentro del mismo.
4. Desarrollar la revisión de becas con el propósito de completar la gestión del proceso de postulación a becas.
5. Implementar funciones de gestión, tales como la unión de las secciones de administración, la carga de revisores mediante archivo y la visualización de resúmenes de apoyo. Esto es esencial para el correcto funcionamiento de los procesos que involucra la conferencia.
6. Validar la usabilidad y experiencia de usuario del sistema implementado, mediante pruebas a posibles postulantes, a revisores y a administradores. Recopilando de estas validaciones niveles de satisfacción y oportunidades de mejora.

1.4. Descripción de la solución

La solución propuesta consistió en implementar un conjunto de funcionalidades, las que se ordenaron de acuerdo a su importancia para el sistema. Lo primero, fue solucionar los problemas de validación y navegación, con el objetivo de evitar la caída del software al apretar cierto botón, o debido a que no se completaron todos los campos requeridos en un formulario. Esto le da continuidad a la navegación y es indispensable para alcanzar un sistema robusto.

Siguió la reestructuración del módulo de becas para cumplir con los requerimientos de la conferencia, lo cual incluyó desde un nuevo modelo de datos hasta un diseño de interfaces usable. Relacionado a esto, fue imprescindible desarrollar la parte de revisión de becas, con el fin de que la gestión del proceso de postulación a becas estuviera terminada. Contempla a grandes rasgos el manejo de asignaciones postulación-revisor, las revisiones y la toma de decisiones finales.

A continuación, se trabajó en la carga de revisores al sistema mediante la utilización de un archivo tipo csv. Este archivo se debe completar con los datos de los revisores como nombre, email, experiencia e idioma, entre otros, de modo que al subirlo se generan automáticamente los objetos revisores con esa información. Junto a los revisores se originan sus respectivos tokens de acceso y se envían por correo.

La función previa mejora la versión anterior, donde para crear revisores se debía completar un formulario por cada uno, con sus datos, en el sitio de administración del framework usado. También se implementó la edición y desactivación de revisores.

Luego, se llevó a cabo la unión de las secciones de administración de trabajos y de revisión de trabajos, y la incorporación de las secciones desarrolladas para becas y revisión de becas, organizando las funcionalidades de manera conveniente, con el propósito de que el administrador no tenga que autenticarse en cada una. Esto igualmente involucra la anexión de feedback al usuario y la uniformización del diseño.

Por último, debido a la cantidad de postulaciones y de revisiones, la sección de administración unificada requirió de resúmenes de apoyo. Tarjetas que muestran, por ejemplo, la cantidad de postulaciones con veredicto y de ellas cuántas han sido aceptadas, rechazadas o están en lista de espera. Asimismo para becas se expone la distribución por tipo de beca solicitada.

1.5. Metodología

Referente a la metodología de trabajo que se siguió para llevar a cabo la solución, se acerca a una metodología de desarrollo incremental. No obstante, no cabe del todo en esa categoría, ya que todas estas formas de trabajo se enfocan en el trabajo en equipo. Lo que se hizo en la práctica fue intentar seguir el plan de trabajo establecido, se separaron las etapas en tareas pequeñas y se distribuyeron a lo largo de las semanas.

Por otro lado, se usó la técnica Pomodoro [4] para mejorar la administración del tiempo. Esta técnica consiste en trabajar continuamente en una determinada actividad por 25 minutos, intervalo de tiempo conocido como pomodoro, tomando descansos de 5 minutos entre pomodoros. Una vez pasados 4 pomodoros, viene un descanso más largo de 10 o 20 minutos. El tiempo de desarrollo diario se fue contabilizando con pomodoros, lo que facilitaba saber si realmente se estaban cumpliendo las horas semanales de dedicación al proyecto.

1.6. Validación y resultados

El testing se enfocó en evaluar la usabilidad del sitio y la experiencia de usuario. Para ello, se separó la validación del software de acuerdo a los perfiles de usuario disponibles (postulantes, revisores y administradores), los que van a utilizar el sitio de postulaciones a becas y trabajos, el sitio de revisiones y el sitio de administración respectivamente.

Para el primer perfil, se reclutaron cinco personas con diferentes características, las que simularon postulaciones a becas y trabajos, contestando al finalizar el test System Usability Scale (SUS) [5]. El sitio de revisiones fue analizado por una revisora de Latinity, la cual probó las diferentes funcionalidades y también respondió el test SUS. Por último, la sección de administración fue validada por una organizadora de Latinity, en forma subjetiva y asíncrona.

Tomando en cuenta los objetivos y los resultados de las pruebas de usuario, se considera que se alcanzó el producto mínimo viable planificado para esta memoria. Dado que se cumplieron todos los objetivos planteados y los resultados, tanto a nivel cuantitativo como cualitativo, fueron satisfactorios. En general, la mayoría de los comentarios se refieren a cambios pequeños, tales como textos, colores y ubicación de elementos.

1.7. Estructura del informe

A modo de resumen de la estructura de este informe, a continuación se listan los capítulos que lo conforman, junto a una breve síntesis de sus contenidos.

- Capítulo 1: Introducción. Contiene el contexto, motivación y problema a abordar. Además de los objetivos y la solución desarrollada.
- Capítulo 2: Estado del Arte. Se presentan los sistemas existentes que fueron estudiados para usarse en la gestión de la conferencia, y sus ventajas y desventajas. También se describe lo que había del sistema, justo antes de iniciar la memoria.
- Capítulo 3: Análisis y Diseño. Se profundiza en el análisis de cada módulo del software y en la solución propuesta. Asimismo se exponen los diagramas de casos de uso, algunos modelos de datos y la arquitectura del sistema.
- Capítulo 4: Implementación. Se detallan las principales tareas desarrolladas, las dificultades enfrentadas y se muestran algunas páginas de la plataforma.
- Capítulo 5: Validación. Se relata el proceso de ejecución de la validación, los instrumentos utilizados y los resultados obtenidos.
- Capítulo 6: Conclusiones. Se presenta el recuento de los objetivos logrados, análisis y reflexiones. También las lecciones aprendidas y el trabajo futuro.

Capítulo 2

Estado del Arte

En cuanto a sistemas de gestión vinculados al tema de esta memoria, actualmente existen varios, entre los que destacan cuatro: EasyChair, HotCRP, PaperCall y el software de Anita Borg para Grace Hopper Celebration. Sistemas que fueron considerados como opciones para la conferencia. A continuación se describen en mayor profundidad.

2.1. EasyChair

EasyChair [6] es una plataforma de administración de conferencias, que de acuerdo a su sitio, ofrece los servicios de convocatorias, gestión de conferencias y soluciones para conferencias virtuales e híbridas. Además de sistemas de registros y pagos, y mecanismos de publicación de presentaciones y actas.

Referente a la gestión de conferencias, ésta engloba la administración de comités, el envío de trabajos, las revisiones, y la comunicación y seguimiento. Con respecto a las revisiones en si, los miembros del comité son quiénes revisan y pueden haber varios comités. El sistema toma en cuenta sus preferencias de revisión al momento de generar las asignaciones revisor-trabajo.

EasyChair posee licencias para organizaciones, las cuales dependen del número de conferencias y propuestas, y licencias para conferencias individuales, que se dividen en cuatro tipos, gratis, profesional, ejecutiva y grupal. Las características disponibles se incrementan de gratis a grupal.

En cuánto al precio, éste va aumentando conforme a la cantidad de postulaciones recibidas, las que se agrupan por rangos. Para Latinity el precio estaría entre 430.000 y 954.000 pesos chilenos, dejando fijo el rango y variando el tipo, de profesional a grupal.

2.2. HotCRP

HotCRP [7] es un software de gestión de conferencias, especialmente académicas, que permite el envío de trabajos y la administración del proceso de revisión. Entre sus características resaltan las funcionalidades de búsqueda y etiquetado de trabajos, la posibilidad de descarga de trabajos y la visualización de estadísticas.

En relación a las asignaciones de revisiones, pueden ser automáticas o a mano. Al igual que EasyChair los miembros del comité son quiénes revisan, pero existe la opción de asignaciones a revisores externos. Cabe mencionar que, los integrantes del comité pueden especificar sus preferencias por trabajo.

HotCRP es de código abierto, por lo que cualquier usuario puede descargarlo, configurarlo y ejecutarlo. La otra alternativa es contratar una versión ya alojada, la cual tiene un costo de cerca de ocho dólares por propuesta, lo que se traduce en unos 1.825.000 pesos chilenos. Para académicos hay una versión gratuita para usar en la sala de clases, sin soporte.

2.3. PaperCall

PaperCall [8] es una plataforma de gestión de eventos y oradores, que facilita el manejo de convocatorias a papers y presentaciones de charlas. Incluye funciones tales como llamado y envío de postulaciones, y comunicación entre organizadores y oradores potenciales. Asimismo incorpora la administración de notificaciones y la toma de decisiones finales sobre trabajos.

PaperCall tiene tres versiones: una gratuita, otra profesional y una personalizada. Las dos primeras presentan límites a la cantidad de propuestas, al número de organizadores y a la cantidad de preguntas configurables. En cambio, la versión personalizada posee casi todos los aspectos ilimitados más soporte constante. En la Tabla 2.1 se muestran los detalles.

Propiedad	Gratuita	Profesional	Personalizada
Cantidad de propuestas	200	Ilimitada	Ilimitada
Cantidad de organizadores	5	20	Ilimitada
Cantidad de preguntas configurables	No se menciona	5	Ilimitada

Tabla 2.1: Principales características de las versiones de PaperCall

El plan profesional cuesta alrededor de $ 380.000 y el personalizado no se especifica, dado que el precio depende de las características que se soliciten y de la cantidad de trabajos. Este último es el que más se ajusta a los requerimientos de Latinity.

2.4. Software de Anita Borg

Anita Borg fue una científica informática estadounidense, que en 1994 junto a Telle Whitney cofundaron Grace Hopper Celebration, con el fin de ofrecer un espacio en el que las

mujeres pudieran mejorar sus habilidades técnicas y generar redes de contacto. Unos años después, en 1997, Anita Borg fundó el Instituto de Mujeres y Tecnología (IWT), con el objetivo de aumentar la representación femenina en el área tecnológica y fomentar la creación de más tecnología por parte de mujeres. La institución pasó a llevar su nombre luego de su fallecimiento y actualmente se encarga de organizar GHC.

El software del Instituto Anita Borg [9] es un sistema a medida que se usa para gestionar Grace Hopper Celebration, en cuanto a postulaciones a becas y trabajos, razón por la que se incluye en este capítulo. No obstante, como está hecho acorde a las necesidades de GHC no se acomoda a lo requerido por Latinity. Grace Hopper Celebration es una conferencia a una escala gigantesca, cuyo tamaño supera varias veces el de Latinity. Como referencia, en 2019 GHC tuvo alrededor de 25.000 asistentes, mientras que Latinity cerca de 600.

2.5. Discusión sobre los sistemas existentes

El principal inconveniente de los primeros tres sistemas abordados es que solo se enfocan en trabajos, es decir, no contemplan implementaciones o configuraciones para becas. Por lo que, solo resolverían la mitad del problema de la gestión de postulaciones a Latinity. Es preciso señalar que solo EasyChair y PaperCall se intentaron usar en versiones anteriores de Latinity.

A modo de contexto, el presupuesto de Latinity varía entre 22 y 45 millones de pesos aproximadamente, de esto alrededor de la mitad se gasta en pasajes y hospedajes de becadas e invitadas, mientras que otro porcentaje considerable se destina a alimentación. De manera que, no se dispone de dinero suficiente para absorber este tipo de gasto cada año.

Además de lo anterior, en particular se notó que EasyChair era complicado de configurar y difícil de ocupar, sobre todo pensando en personas no familiarizadas con este tipo de plataformas, como revisores de la industria o estudiantes. Por ejemplo el administrador debe hacer click en cada postulación para ver si se encuentra revisada o no. En consecuencia, se descartó por su baja usabilidad y complejidad.

Debido a lo poco amigable de EasyChair, se analizó la idea de utilizar HotCRP, sin embargo se desechó por su orientación a papers. Los papers son lo fundamental para este sistema. Además tenía la dificultad asociada a habilitar un sitio que no se conocía completamente y a mantenerlo operativo por el tiempo que durara el proceso de selección de trabajos.

Por último, PaperCall resultó ser el más caro, en vista de que cobra por trabajo revisado, escapándose del presupuesto de la conferencia. Por otro lado, se advirtió que no está enfocado a conferencias científicas y que solo permitía cinco cuentas de revisores.

A modo de resumen, se presenta a continuación una tabla comparativa con los tres primeros sistemas (ver la Tabla 2.2), de acuerdo a algunos atributos relevantes. Cabe señalar que el de Anita Borg no se incluyó, dado que existe poca información de sus características específicas.

Atributo	EasyChair	HotCRP	PaperCall
¿Implementaciones para becas y trabajos?	Solo trabajos	Solo trabajos	Solo trabajos
Costo aproximado	430.000-954.000	1.825.000	No se especifica
¿El costo depende de la cantidad de postulaciones?	Sí	Sí	Sí
¿Revisores externos a la organización?	No	Sí	Sí, pero limitado a 5

Tabla 2.2: Tabla comparativa entre EasyChair, HotCRP y PaperCall, considerando la versión alojada de HotCRP

2.6. Situación actual

Como se mencionó previamente, no existía un sistema que lograra cumplir con todos los requerimientos para llevar a cabo la administración de la conferencia, en cuanto a becas y trabajos. Motivo por el que se hizo necesario comenzar a construir uno. Al momento de partir este tema solo habían partes que correspondían al MVP (Producto Mínimo Viable) definido en cada semestre, con implementaciones parciales del software.

El proyecto Latinity justo antes de empezar la memoria se puede dividir en tres grandes módulos. Estos abarcaban un porcentaje importante de lo requerido para gestionar la conferencia, no obstante, no eran utilizables por si solos. Faltaban funcionalidades claves, conexión de secciones y uniformidad de diseños, tanto a nivel de modelos de datos como de interfaces, que permitieran alcanzar un sistema robusto.

El módulo de becas fue desarrollado por el equipo Latinity Primavera 2019, conformado por Bárbara Castro, Sebastián Cifuentes, Kyra Cossio, Daniel Gálvez, Felipe Lizama, Josefa Robert y Miguel Sepúlveda. Se encargaba del proceso de postulación a becas y contemplaba las tareas que se listan a continuación.

- Ingreso de postulantes.
- Postulación a través de un formulario.
- Almacenamiento de las respuestas en la base de datos.
- Descarga de la información de las postulaciones, por parte del administrador.

El módulo de becas fue el primero de la serie de proyectos. Luego de un primer análisis, se notó que se encontraba diseñado, tanto los modelos como las interfaces, de una manera

que no se ajustaba a las necesidades de este proceso. Esto hacía que el módulo no fuera escalable. Lo anterior se debe principalmente a la interrupción por el estallido social y a la menor experiencia con respecto a los formularios que se querían.

Las interfaces no presentaban suficiente feedback al usuario, no eran intuitivas y en algunos casos exponían información errónea. Por ejemplo, todas las postulaciones enviadas se mostraban luego como rechazadas. Por otro lado, en el caso de postulantes estudiantes, a los que se les solicita una carta de recomendación, se enfrentaban a los inconvenientes de que al momento de subirla no se especificara la extensión requerida, ni se informara si realmente se había subido. Desde el punto de vista del modelo de datos, los modelos contenían demasiados campos y no permitían borrar ni modificar las preguntas de los cuestionarios de postulación.

Entre las funcionalidades que le faltaban a este módulo destaca que no se podían asignar revisores a postulaciones, por lo que no era posible revisar las postulaciones. También, que no se podían tomar decisiones finales de asignación de becas, considerando los puntajes de los formularios de revisión, ya que no existían dichos formularios. Solo era posible tomar veredictos directamente, por parte del administrador. Dicho de otra manera, faltaba la parte de revisión de becas.

El módulo de trabajos fue desarrollado por el equipo Latinity Otoño 2020, integrado por Mauricio Araneda, Marcelo Becerra, Javiera Díaz, Sebastián Donoso, Israel Peña y Valentina Pinto. Es responsable del proceso de postulación a trabajos, lo que abarca lo exhibido en la lista de abajo.

- Acceso de postulantes.
- Postulación mediante un formulario, según el tipo de trabajo.
- Almacenamiento de las respuestas en la base de datos.
- Visualización de un resumen de postulaciones, por parte del postulante.
- Posibilidad de realizar una postulación grupal.
- Opción de descarga, para el administrador, de toda la información de las postulaciones.

El módulo de trabajos presentaba problemas de diseño de interfaces y usabilidad, por ejemplo existían campos en los formularios que debían ser obligatorios y no lo eran; y otros que sí eran obligatorios, pero no se les indicaban al usuario, provocando que en algunos casos el sistema fallara.

Otros inconvenientes a resaltar son que una vez enviada la postulación se podía seguir editando, incluso si el proceso de postulaciones se encontraba cerrado. Además, si una persona enviaba varias postulaciones, todas quedaban con el mismo nombre de trabajo. Por lo que, si se modificaba el nombre de uno de estos trabajos, cambiaba el de todos.

El módulo de revisión de trabajos fue desarrollado por el equipo Latinity Primavera 2020, cuyos miembros fueron Roberto Aguilera, Valentín Espina, Diego Ortego, Camila Quilape, Felipe Sanhueza, Jorge Toloza y Sebastián Vega. Se encarga, tal como dice su nombre, del manejo y revisión de postulaciones a trabajos. Incluye las funciones de la siguiente lista.

- Asignación de revisores a postulaciones.
- Ingreso de revisores al sistema.
- Revisión de postulaciones a través de formularios.
- Almacenamiento de las respuestas de las revisiones.
- Presentación al administrador de la información de los revisores, como nombre, correo, idioma, etc.
- La posibilidad del administrador de realizar veredictos, vale decir aceptar o no una postulación.

Entre los problemas que tenía el módulo de revisión de trabajos sobresalía que la carga de revisores fuera de forma manual, o sea solo era posible subir la información uno a uno mediante el sitio de administración de Django, el framework utilizado. Este módulo también requería de más feedback en las páginas dedicadas al administrador, que mejoraran la experiencia del usuario.

De lo mencionado en los párrafos anteriores se desprende la importancia de continuar con el desarrollo, con el objetivo de alcanzar un producto funcional, unificado y usable, que facilite la gestión de los procesos de becas y trabajos pertenecientes a la conferencia Latinity. Se puede separar el trabajo de la memoria, por lo tanto, en corregir problemas, reestructurar el módulo de becas e implementar funcionalidades faltantes.

El software Latinity está desarrollado en Django y su código permanece en un repositorio de GitHub [10] cuyo nombre es `latinity2020`. Django es un framework diseñado para agilizar el desarrollo de aplicaciones web, se encuentra escrito en Python y es de código abierto. Además se enfoca en aspectos como el reuso, el desarrollo rápido y la conectividad. Así como también en la extensibilidad y seguridad.

Django organiza los proyectos en aplicaciones, directorios con una determinada estructura, que contienen archivos típicos como `admin.py, models.py, views.py`, y las carpetas `migrations` y `templates`.

La carpeta `migrations` tiene las migraciones, unos archivos especiales que reflejan los cambios en la base de datos. Mientras que `templates` incluye los archivos templates: un conjunto de plantillas basadas en etiquetas con herencia, que dan lugar a los html's que se muestran al usuario.

Cabe mencionar que un modelo corresponde a la representación de una tabla, va en el archivo `models.py`, y que una vista es una función o clase que se encarga de gestionar los datos y proveer de información al template. Las vistas se escriben en `views.py`.

A continuación, en la Tabla 2.3 se exhiben las aplicaciones iniciales del proyecto, junto a un resumen de sus principales características.

	Aplicación	# modelos	# vistas	# templates	# líneas
General	home_page	0	4	4	371
	site_conf	5	0	0	160
Becas	grants_admin	12	6	4	1193
	grants_applications	0	5	10	1200
Trabajos	works_admin	13	8	5	1623
	works_applications	3	8	14	2476
Revisión de trabajos	review_admin	7	7	4	1190
	review_reviewers	3	3	5	935

Tabla 2.3: Resumen del proyecto inicial, sin considerar los archivos de carga de datos de prueba

Capítulo 3

Análisis y Diseño

Se considera que los resultados de la revisión inicial del sistema, presentados en Introducción y Situación Actual, respaldaron la necesidad de conectar las implementaciones, resolver errores y continuar avanzando. Esto con el propósito de lograr por primera vez un software unificado, uniforme, funcional y completo.

En consecuencia, el problema abordado fue extender lo desarrollado. Se planificó solucionar los problemas de validación y navegación de los módulos de trabajos y revisión de trabajos, reestructurar por completo el módulo de becas, tanto a nivel de modelos de datos como diseño de interfaces, y añadir la parte de revisión de becas. También unificar las secciones de administración, agregar la gestión de revisores y mejorar el diseño del front-end.

En este capítulo se describe la solución desarrollada, se especifican los casos de uso que se querían cumplir y se exponen los hallazgos del análisis detenido al software recibido. Asimismo se incluyen las decisiones de diseño tomadas, los nuevos modelos de datos, y una sección orientada a arquitectura.

Antes de continuar, es relevante precisar el tipo de personas que van a usar este sitio. Se clasifican en tres categorías, postulantes a becas y/o trabajos, los cuales pueden ser estudiantes, profesionales o académicos. Revisores, que son quienes evalúan las postulaciones, por lo general académicos, y organizadores o administradores, las personas que administran todos los procesos y que determinan los veredictos finales de las postulaciones.

3.1. Descripción general

La solución propuesta consistió en implementar un conjunto de funcionalidades, las que se exhiben a continuación ordenadas de acuerdo a su importancia para el sistema. Estas funcionalidades surgieron como resultado de la exploración mencionada en el capítulo anterior, sumado a lo requerido por la profesora guía, quién además es parte del equipo organizador de Latinity.

Lo primero, fue solucionar los problemas de validación y navegación de los módulos de trabajos y revisión de trabajos, con la finalidad de evitar que el software fallara al apretar cierto botón, o debido a que no se completaron todos los campos obligatorios en un formulario. Dándole así continuidad a la navegación.

Posteriormente se sopesó cómo llevar a cabo la reestructuración del módulo de becas, es decir, si se partía de cero o se modificaba lo ya escrito. Esto tomando en cuenta que el modelo de datos se iba a diseñar de nuevo, al igual que las interfaces. El modelo de datos inicial no cumplía con las necesidades de la conferencia, en concreto, no permitía un manejo adecuado de los datos y encapsulaba demasiada información en los objetos.

Relacionado a lo anterior, para completar la gestión del proceso de postulación a becas fue pertinente implementar la revisión de becas. Para ello se evaluó si era posible extender el módulo de revisión de trabajos o si se necesitaba un módulo nuevo enfocado a becas. La revisión de becas abarca la asignación de revisores a postulaciones, la evaluación de las postulaciones asignadas mediante formularios y la toma de decisiones finales, por parte de los administradores, en base a las revisiones.

Dado que el sistema requería de una manera más eficiente para cargar revisores, se optó por utilizar un archivo tipo `csv`, en el que se pudieran ingresar todos los datos, como nombre, email, experiencia, idioma, entre otros. De modo que al subirlo, se crearan los objetos revisores junto a sus respectivos tokens de acceso. Estos tokens después son enviados por correo electrónico.

La carga usando archivos dio origen a una nueva sección, la de gestión de revisores, donde también se incorporó la edición y la activación/desactivación de revisores. Cabe destacar que la plataforma debía desarrollarse de tal forma que permitiera a un revisor evaluar postulaciones a becas y trabajos mediante un solo token de acceso, si es que tenía asignaciones de ambos tipos.

Cada uno de los módulos iniciales poseía una sección de administración, vale decir un grupo de páginas dedicadas exclusivamente a los organizadores. En la de trabajos, los administradores podían ver y descargar información de las postulaciones, y configurar plazos. Por otra parte, en la de revisión de trabajos, podían asignar revisores a postulaciones a trabajos, ver las revisiones realizadas y tomar veredictos.

De aquí surgió la necesidad de crear una sección de administración unificada, conectando las secciones existentes con las nuevas, y organizándolas de manera conveniente. Igualmente se añadió feedback para el usuario y se uniformizó el diseño de las páginas. Todo esto con el fin de que el administrador no tuviera que autenticarse en cada una de las secciones para acceder a las funcionalidades provistas por dichas partes.

Por último, debido al gran número de postulaciones y revisiones que maneja Latinity, la sección de administración requirió de resúmenes de apoyo. Los resúmenes son tarjetas que muestran, por ejemplo, la cantidad de postulaciones por tipo de trabajo y el número de postulaciones completas o incompletas. Asimismo la cantidad de postulaciones con veredicto y de ellas cuántas han sido aceptadas, rechazadas, están pendientes o en lista de espera.

3.2. Diagrama de casos de usos

A continuación, en las Figuras 3.1 a 3.4, se muestran los diagramas de casos de uso, que resumen el estado del proyecto previo a la memoria y lo que se esperaba realizar. El color blanco indica un caso incompleto, el verde una implementación completa, el rojo que no estaba desarrollado y el naranjo que existía, pero se iba a reestructurar. El propósito era que al final de este trabajo de título todos los casos estuvieran verdes.

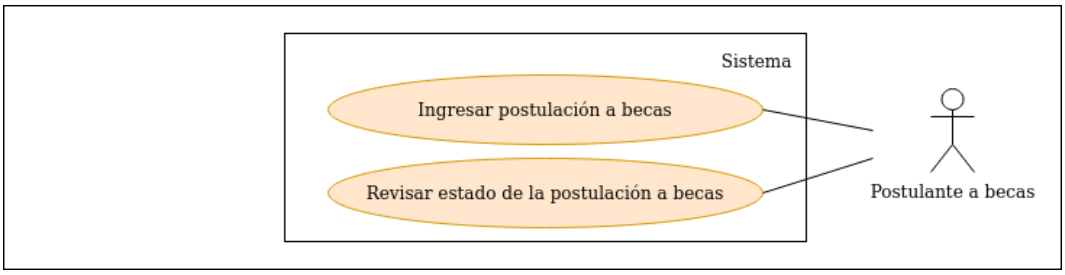

Figura 3.1: Diagrama de casos de uso para postulante a becas

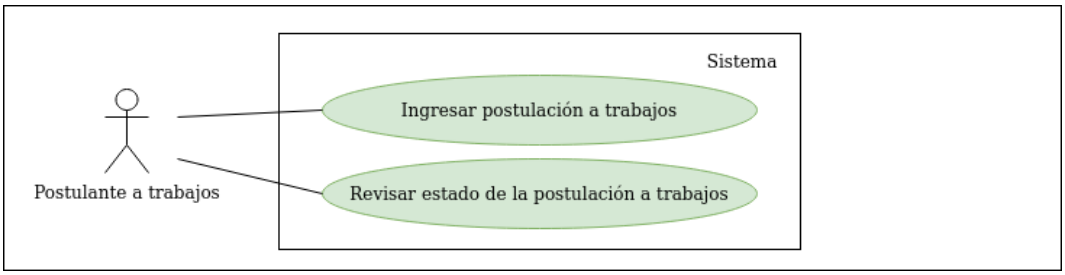

Figura 3.2: Diagrama de casos de uso para postulante a trabajos

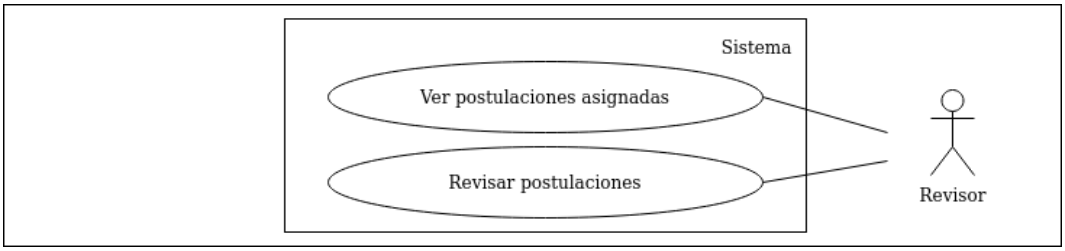

Figura 3.3: Diagrama de casos de uso para un revisor

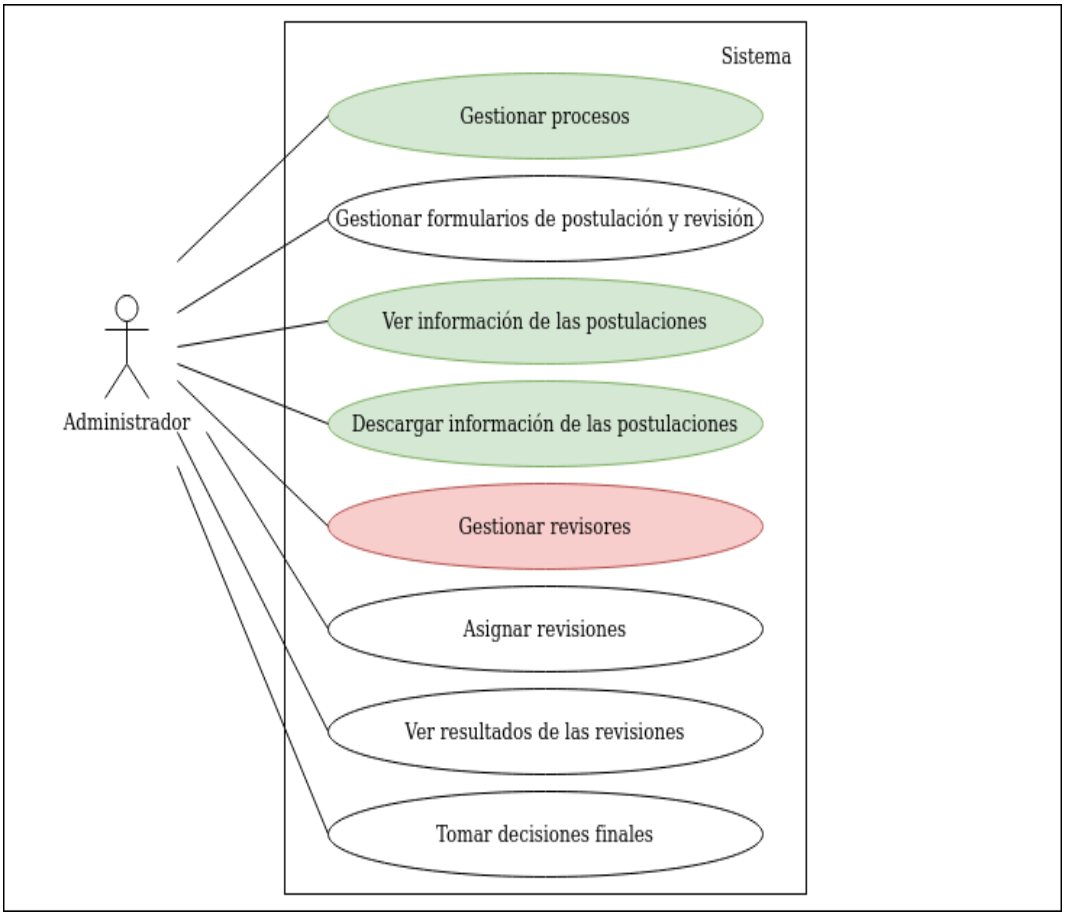

Figura 3.4: Diagrama de casos de uso para un administrador

3.3. Análisis del sistema legado

A continuación se presentan los resultados de leer y estudiar el código del software recibido. Esta sección se divide en cuatro subsecciones, las primeras tres destinadas a becas, trabajos y revisión de trabajos respectivamente y la última a la discusión. En cada una de las subsecciones orientadas a módulos se describe la estructura correspondiente, las principales funcionalidades, y los problemas detectados, junto a ejemplos de esos problemas. Además, en revisión de trabajos se incluye el análisis de cómo utilizar lo desarrollado para revisión de becas.

La subsección discusión contiene comentarios generales y el estudio de los estados pertenecientes a los modelos más esenciales. Además, se ahonda en los inconvenientes de los modelos del módulo de becas y en el diseño del nuevo modelo de datos, sumado a observaciones respecto a la reestructuración del mismo. Igualmente se dispone una breve conclusión y reflexiones.

En concreto, se partió revisando el módulo de becas, luego el de trabajos y finalmente revisión de trabajos. El análisis se enfocó en el back-end, por lo que en cada módulo se examinaron detenidamente los archivos `admin.py`, `forms.py`, `models.py`, `urls.py` y `views.py`. Los

demás archivos se observaron superficialmente, solo con el objetivo de entender sus usos y sus dependencias.

En cuanto al procedimiento seguido, se anotó cada problema encontrado junto a las líneas de código asociadas, con el propósito de identificarlos rápidamente en el futuro. En algunas ocasiones, también fue necesario entender cómo el inconveniente hallado afectaba al front-end. Por este motivo se mencionan de igual forma detalles de las interfaces que no se notaron inicialmente.

Al terminar de revisar un módulo se sintetizaban los problemas detectados, agrupando los similares o relacionados, con el fin de facilitar la redacción de este capítulo. Cabe aclarar que no se profundiza en los errores pequeños, tales como los de sintaxis o palabras mal escritas.

3.3.1. Becas

El módulo de becas tenía dos submódulos, `grants_admin y grants_applications`. El primero orientado a la sección de administración del módulo y el segundo a las funcionalidades a las que el postulante requiere acceder. El modelo de datos se exhibe en la Figura 3.5.

El submódulo `grants_admin` mostraba la configuración de fechas del proceso de postulación a becas, el detalle de las postulaciones recibidas y la opción, para el administrador, de asignar veredictos a las postulaciones. Por otro lado, en `grants_applications` estaba implementado el login para el postulante, el formulario de postulación, y una página con el estado de la postulación.

Lo primero que se hizo en este módulo fue solucionar un par de problemas que impedían que el usuario accediera al formulario de postulación. Uno de ellos ocurría al momento de solicitar un token. Al introducir un correo y seleccionar enviar, el sistema se caía y no llegaba el correo con el token de acceso. Investigando, se descubrió que el correo vinculado a Latinity, el que envía los tokens y al que llegan los emails de contacto, había dejado de funcionar, debido a cambios realizados por Google a las configuraciones de seguridad. Por lo que, se tuvieron que cambiar los ajustes de Gmail para solucionar este inconveniente.

El otro era un problema de redireccionamiento. Cuando se ingresaba con el token generado, el sitio siempre redirigía a una página que exponía el estado de la postulación y el estado del proceso, independiente de si había efectivamente una postulación y del estado en que estuviera el proceso. Siempre aparecía la postulación como rechazada y el proceso cerrado. De modo que, se corrigió la redirección inicial.

Desde el punto de vista del código, se notó que el modelo Proceso tenía dos opciones de estado, abierto asociado al número 1 y cerrado al 2. Sin embargo, en el archivo `views.py` de `grants_admin y grants_applications` se asumía que los valores eran 0 y 1 respectivamente, lo cual provocaba condiciones erróneas en las vistas.

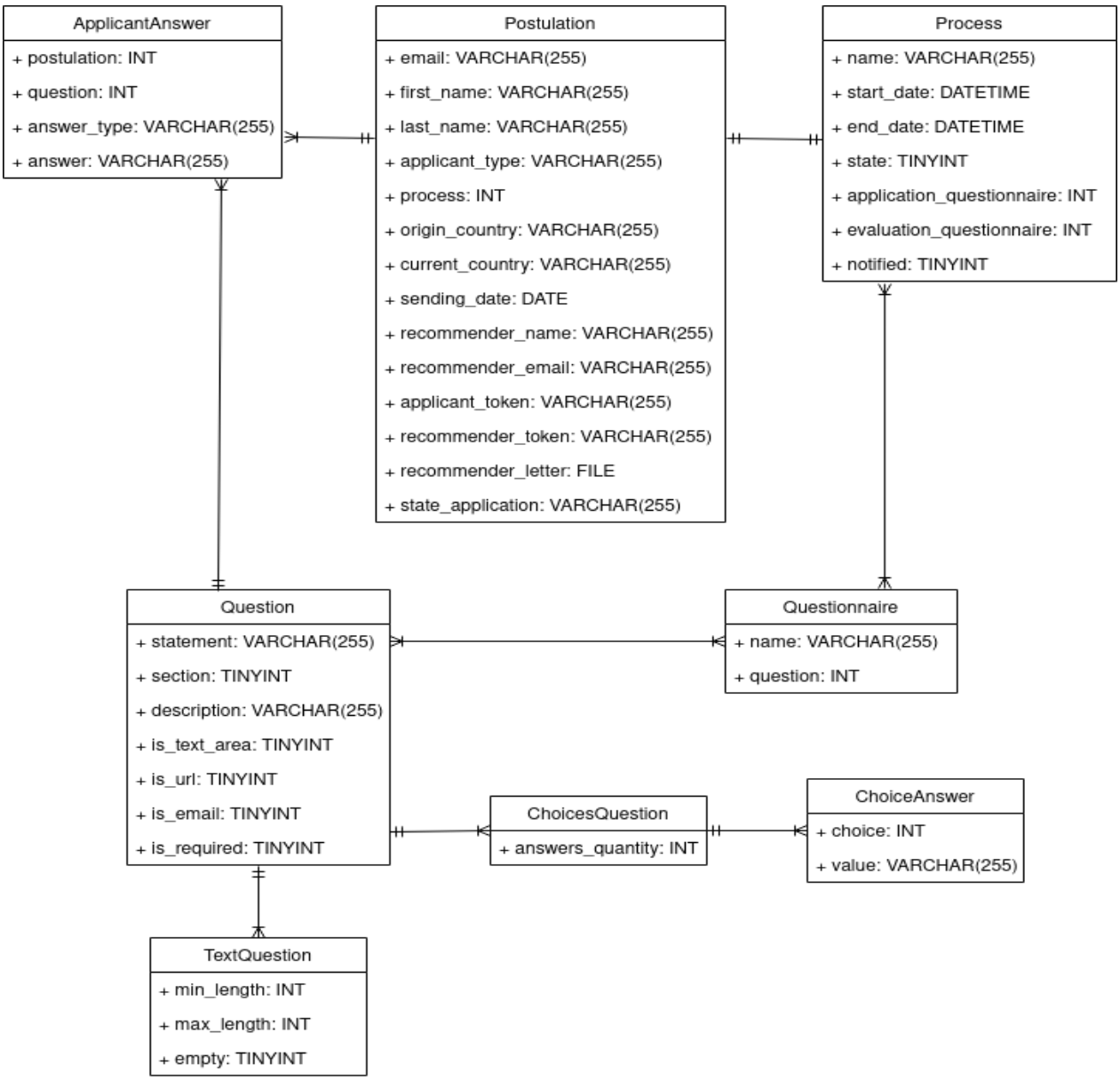

Figura 3.5: Modelo de datos de becas

Un ejemplo concreto de lo anterior es que se podían pedir tokens con el proceso finalizado, lo que no tenía mucho sentido. El sistema está diseñado de tal forma que el token de un usuario sea por proceso, por lo que el token solicitado cuando el proceso está cerrado no sirve para el del año siguiente.

Continuando con las vistas, se advirtió que la encargada de cambiar los plazos del proceso de postulación a becas no funcionaba bien. Variar la fecha inicial y/o fecha final no modificaba el estado del proceso, vale decir si estaba abierto o cerrado. Esto también generaba inconvenientes en otras vistas, en las que revisaban este estado antes de realizar ciertas acciones.

Otro problema grave era el tema de los filtros. Como se definió antes, las vistas se encargan de recuperar información de la base de datos estableciendo condiciones, esas condiciones son los filtros. Cuando se verificaba si un correo ya existía en el sistema, faltaba agregar como filtro el proceso, de lo contrario una persona solo iba a poder postular una vez. La idea es que no hayan dos postulaciones o más de un mismo usuario en un determinado proceso, pero sí podría postular a varios distintos.

En general, se advirtió una mezcla de español e inglés en el software, y la presencia de texto de relleno del estilo *Lorem ipsum* [11]. Por otra parte, algunos botones y links llevaban a páginas que no debían. Por ejemplo, como consecuencia de un refactoring, los templates de becas quedaron con el mismo template base que los de trabajos, provocando que al presionar el botón cerrar sesión se redirigiera al login de la sección de administración de trabajos, lo cual no estaba bien en becas ni trabajos.

A modo de ejemplo de los problemas de diseño, se muestra cómo se veía el formulario para subir una carta de recomendación (ver la Figura 3.6). La imagen corresponde al paso posterior a cargar una carta. Se advirtió que la única diferencia con lo que estaba antes era el recuadro y ticket verde, por lo que, sumado a que no se indicaba el nombre del archivo, no quedaba claro si se había subido realmente o no. Otro factor que contribuía a esta incertidumbre era que no se exponía el tipo de archivo requerido, como se mencionó en Situación Actual.

Figura 3.6: Formulario para subir la carta de recomendación

3.3.2. Trabajos

El módulo de trabajos tiene dos submódulos: uno dedicado a administración `works_admin` y otro al postulante `works_applications`. El submódulo `works_admin` tenía opciones para descargar las postulaciones de acuerdo al tipo de trabajo y la funcionalidad de configurar las fechas del proceso. Igualmente incluía resúmenes y detalles de las postulaciones por tipo.

El submódulo `works_applications` se encarga de mostrar información al usuario, como el resumen al enviar una postulación y el listado de postulaciones enviadas (en trabajos

se pueden ingresar varias postulaciones). Asimismo implementa los formularios de login, de olvido de token y de postulaciones. En relación al modelo de datos, corresponde a los modelos en blanco de la figura 3.7.

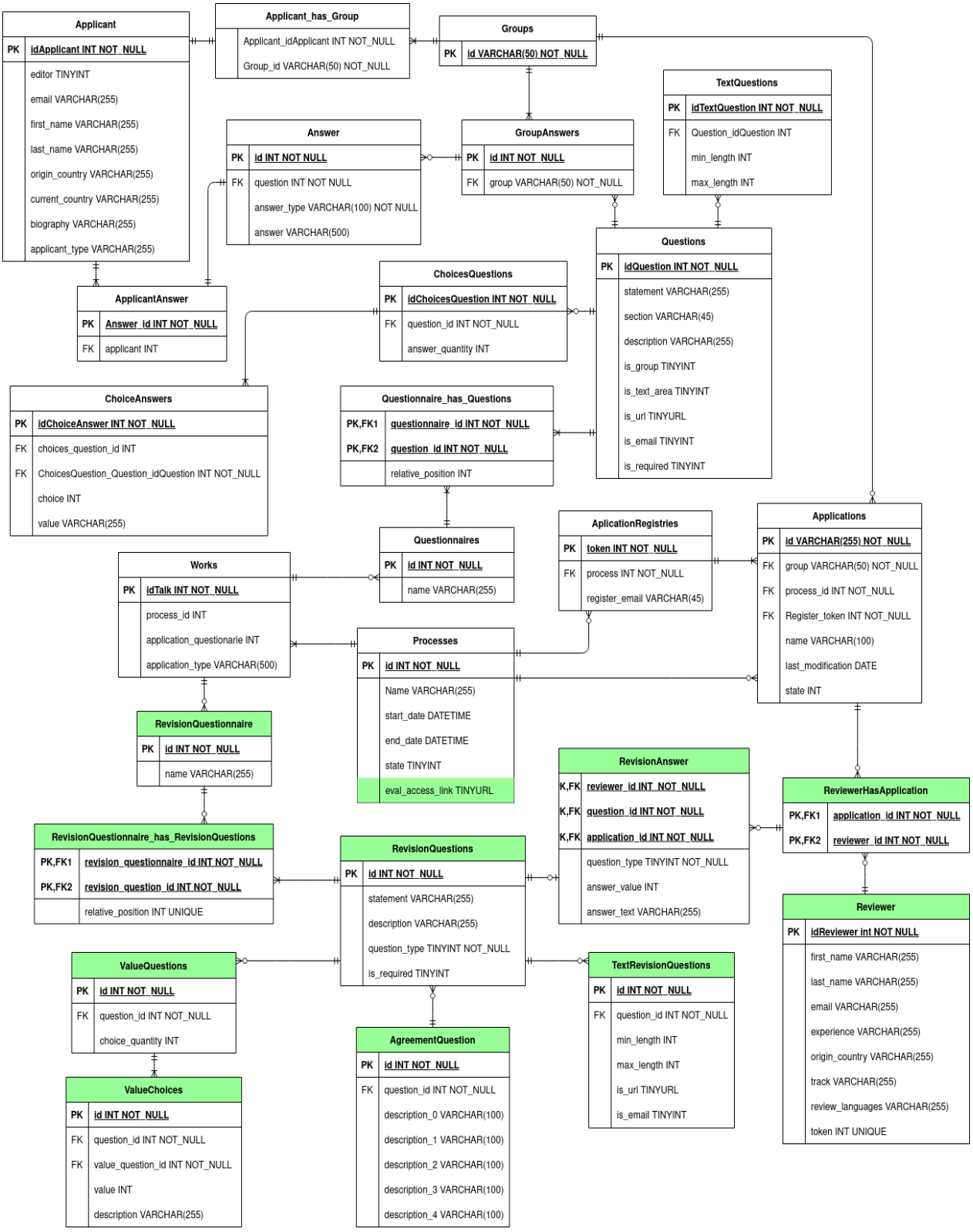

Figura 3.7: Modelo de datos de trabajos y de revisión de trabajos

Aquí también se revisó el cambio de plazos del proceso de postulación a trabajos, el que funcionaba parcialmente. Se notó un desfase de cuatro horas y que la fecha de inicio no se actualizaba correctamente en el front-end, pero sí a nivel de base de datos. Con respecto al estado, tampoco se modificaba de inmediato, debido a una función guardar mal ubicada.

En este módulo habían tres estados posibles para el proceso: pre-abierto, abierto y cerrado, cuyos valores eran 0, 1 y 2 respectivamente. No obstante, en las vistas solo se consultaba si el estado era verdadero o falso, lo cual no consideraba todos los casos y producía, por ejemplo, que se pudieran editar postulaciones cuando el proceso ya estaba cerrado.

En general, a nivel de código se advirtió la presencia de un número considerable de líneas que sobraban, ya sea porque estaban repetidas o porque involucraban variables o funciones que se definían y luego no se empleaban. Por ejemplo, en el submódulo `works_admin` se implementaba un archivo forms, con la definición de formularios, que en la práctica no se utilizaba. Igualmente se observó una falta de optimización, condiciones malas debido al uso equivocado de or/and, y links sin funcionalidad o con lógica errada.

Por otro lado, existían varios filtros donde faltaba añadir el proceso. Esto podía provocar que se mandaran correos a todas las personas registradas en el sistema, que se mezclaran postulaciones de años diferentes o que se utilizaran cuestionarios de versiones anteriores de Latinity. Asimismo se tenían consultas donde se asumía que iban a retornar un objeto, pero estaban devolviendo más.

Por último, se detectó que en la parte de administración, los archivos descargados de las páginas de inicio, resumen y detalles contenían la misma información, la correspondiente al tipo de trabajo seleccionado. Vale decir que si por ejemplo se apretaba descargar presentación en las tres secciones, esos tres archivos eran iguales.

La captura de pantalla de la Figura 3.8 expone cómo se encontraba el primer paso del formulario de postulación, donde se selecciona el tipo de trabajo, es decir si se trata de un póster, una presentación o un círculo de mentoría. Destaca el diseño de la interfaz, el header solo estaba compuesto del logo de Latinity y del botón cerrar sesión. Por otro lado, el footer aparte de hallarse flotando a media página, no alcanzaba el ancho total.

La imagen de la Figura 3.9 representa el segundo paso del formulario de postulación, el cual permite ingresar los datos personales de el o los postulantes. Estos datos eran nombre, correo, tipo de postulante, país de origen, país actual, biografía y nombre del trabajo. Todos los campos excepto biografía y nombre del trabajo eran obligatorios, pero no se le señalaba esto al usuario, ocasionando, como se mencionó antes, que el sistema se cayera. Por otra parte, se advirtió que nombre del trabajo debía ser obligatorio, ya que es uno de los aspectos más relevantes de una postulación.

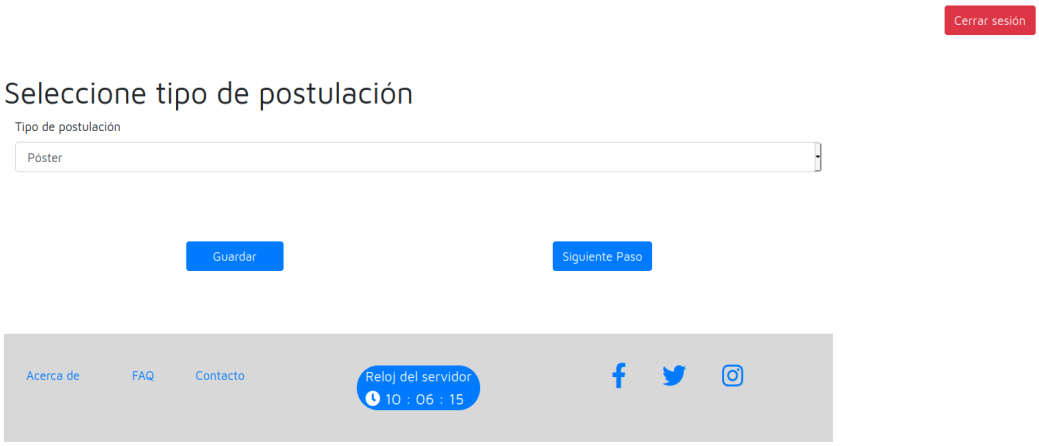

Figura 3.8: Selección del tipo de trabajo a postular

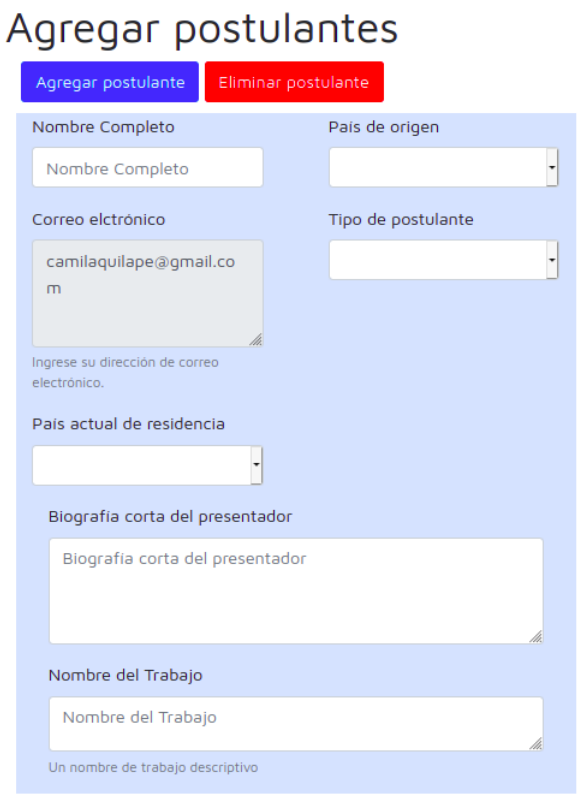

Figura 3.9: Formulario de postulación grupal

3.3.3. Revisión de trabajos

El módulo de revisión de trabajos engloba dos submódulos, llamados `review_admin` y `review_reviewers`. El primero orientado a la sección de administración del módulo y el segundo a las funciones que requiere el revisor. El modelo de datos de ambos submódulos se presenta de color verde en la Figura 3.7.

El submódulo `review_admin` tenía dos secciones. Una el panel, donde se mostraba el listado de revisores, y las postulaciones junto a los revisores asignados. Así como también la funcionalidad de descargar y subir un archivo para cargar las asignaciones de revisión, archivo que especifica los revisores que van a evaluar cada postulación.

Mientras que en la otra sección, la de veredicto, se exponía el detalle de una determinada postulación, como las respuestas a las preguntas de postulación y de revisión, y la opción de tomar un veredicto. Por otra parte, en `review_reviewers` estaba implementado el login del revisor, el panel con las postulaciones a revisar y el formulario de revisión.

Al igual que con los módulos anteriores, éste igualmente tenía problemas con los filtros. Cabe señalar que estos inconvenientes no se detectaron antes debido a que siempre se probó el sistema con un solo proceso, una versión de Latinity. Por otro lado, se advirtió que en algunas partes del código se recuperaba información consultando por el id de una pregunta dada, lo cual puede provocar problemas si dicha pregunta cambia de contenido.

Volviendo a la sección de administración, se notó que en esta parte se asumía que todas las postulaciones a trabajos habían sido enviadas, ocasionando que se pudieran asignar revisores a postulaciones incompletas. Esta situación se puede observar en la Figura 3.10. Era necesario separar las postulaciones completas o enviadas de las incompletas, con el fin de evitar confusiones y facilitar el envío de correos. Por ejemplo, para avisar a los postulantes que no han terminado de postular, que queda poco tiempo.

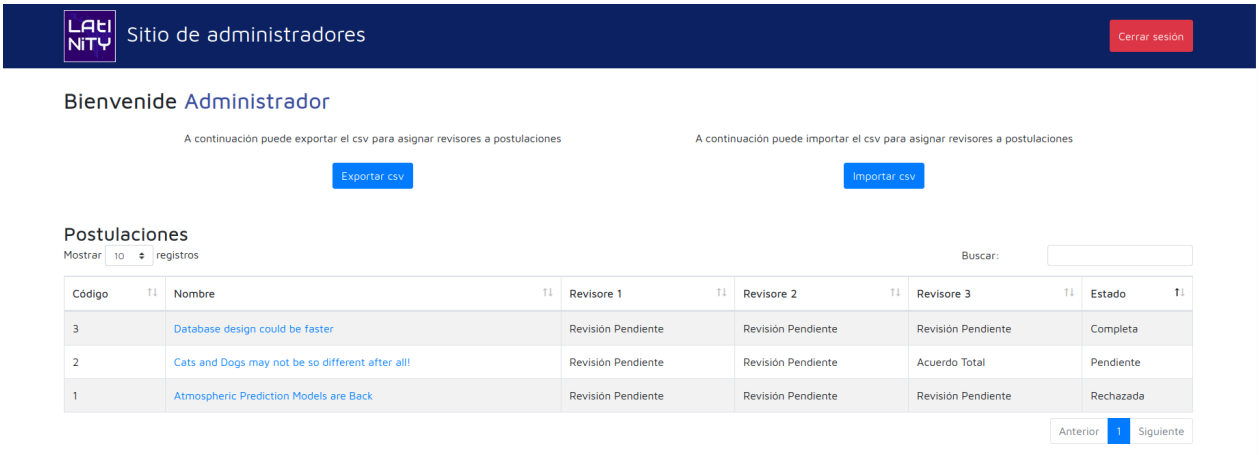

Figura 3.10: Panel del administrador

Aquí mismo, pero en la página de toma de decisiones finales (ver la Figura 3.11), se reparó que la primera selección de un veredicto no quedaba visible. Es decir, presionar el botón del estado escogido para una determinada postulación no producía ningún cambio visual, pero sí uno en la base de datos.

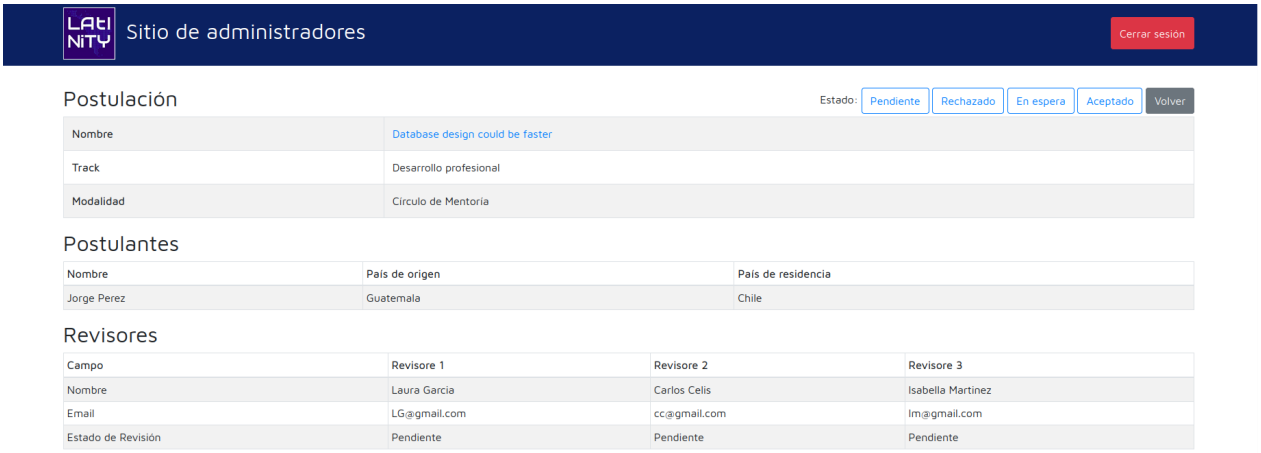

Figura 3.11: Toma de veredictos más detalle de una postulación a trabajos

Del análisis de este módulo, se desprende la necesidad de establecer un modelo proceso de revisiones, que incorpore un periodo de revisión a través de los campos fecha de inicio y fecha de término. Esto con el propósito de prevenir que los revisores editen sus respuestas mientras se estén tomando veredictos sobre las postulaciones.

Además se estudió cómo utilizar lo ya implementado en revisión de trabajos para revisión de becas. Se requerían un par de modelos nuevos, uno para asociar un revisor a una postulación a becas y otro para almacenar la respuesta a cada pregunta de revisiones de becas. Asimismo era preciso desarrollar vistas nuevas, excepto la que se encargaba de obtener las postulaciones a revisar por un revisor, la que debía extenderse para incluir las postulaciones a becas. Desde el punto de vista de los templates, se podían emplear los mismos aplicando pequeños cambios.

3.3.4. Discusión

De los resultados anteriores, se concluye que lo más grave era la inconsistencia de los estados y los problemas con los filtros. A modo de resumen, a continuación se exhiben los diagramas de estados realizados para los principales modelos del software inicial (ver Figuras 3.12 y 3.13).

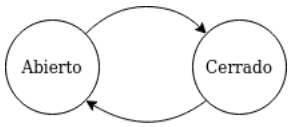

Figura 3.12: Proceso de postulación a becas

Figura 3.13: Proceso de postulación a trabajos

Referente al proceso de postulación a becas (ver la Figura 3.12), se mantuvieron los estados, pero se modificaron los valores asociados, de modo que fueran 0 y 1. De manera similar, en el proceso de postulación a trabajos (ver la Figura 3.13), se eliminó el estado pre-abierto, ya que no aportaba valor al proceso y se ajustaron los otros dos estados para que también fueran 0 y 1.

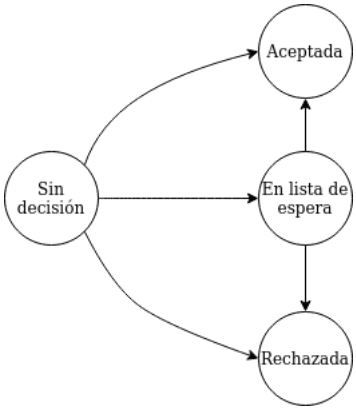

Figura 3.14: Diagrama de estados de postulación a becas

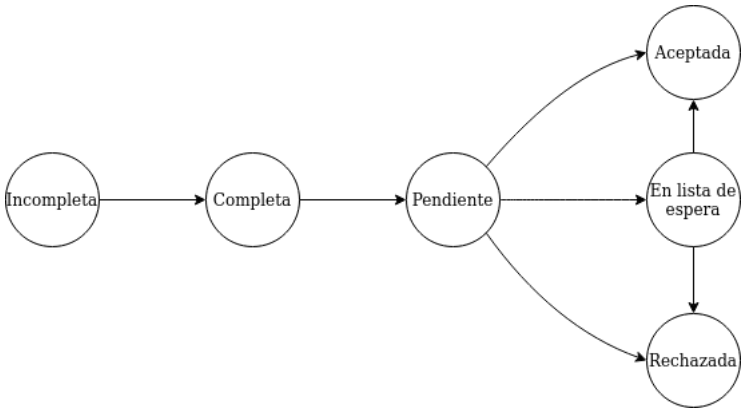

Figura 3.15: Diagrama de estados de postulación a trabajos

Se advierte en el diagrama de la Figura 3.14 la falta de estados intermedios, que reflejaran si una postulación había sido enviada o revisada por ejemplo. La mayor parte del tiempo el estado era sin decisión, por lo que no se proporcionaba suficiente información. En concreto, se verificaba indirectamente si una postulación estaba enviada, chequeando el atributo fecha de envío, el cual solo tenía una fecha en ese caso.

En consecuencia, el objetivo era que este diagrama representara los mismos estados que el modelo de postulación a trabajos. Igualmente es importante nombrar que el modelo de postulación a becas era el único donde los estados eran de tipo `char`, de manera que asimismo se realizó el cambio a flags.

Figura 3.16: Estados de la revisión de trabajos, modelo ReviewerHasApplication

En la Figura 3.16 se muestra el estado del modelo `ReviewerHasApplication`, donde se señala si cierto revisor ha revisado una determinada postulación a trabajos. En otras palabras, se pasa de pendiente a completa al enviarse el formulario de revisión.

Por otra parte, en relación a postulación a trabajos (ver la Figura 3.15), su estado parte en incompleta y cambia a completa cuando el postulante la envía. Luego, varía a pendiente cuando todos los revisores que deben evaluarla lo hacen, para ello se comprueba el estado de los `ReviewerHasApplication` correspondientes. En pendiente la postulación queda lista para que el administrador tome un veredicto. Cabe mencionar que había un estado llamado revisada, que no se incluyó dado que no se estaba usando y que posteriormente se eliminó.

Con respecto al modelo de datos de becas, las principales complicaciones se vinculaban con la cantidad de información que encapsulaban los modelos, y con las decisiones de diseño tomadas. Como se ve en la Figura 3.17, el modelo `Postulación` contenía datos del postulante, de la postulación (proceso, fecha de envío y estado) y del recomendador, cuyos campos nombre, correo, carta y token solo se completaban si el postulante era estudiante.

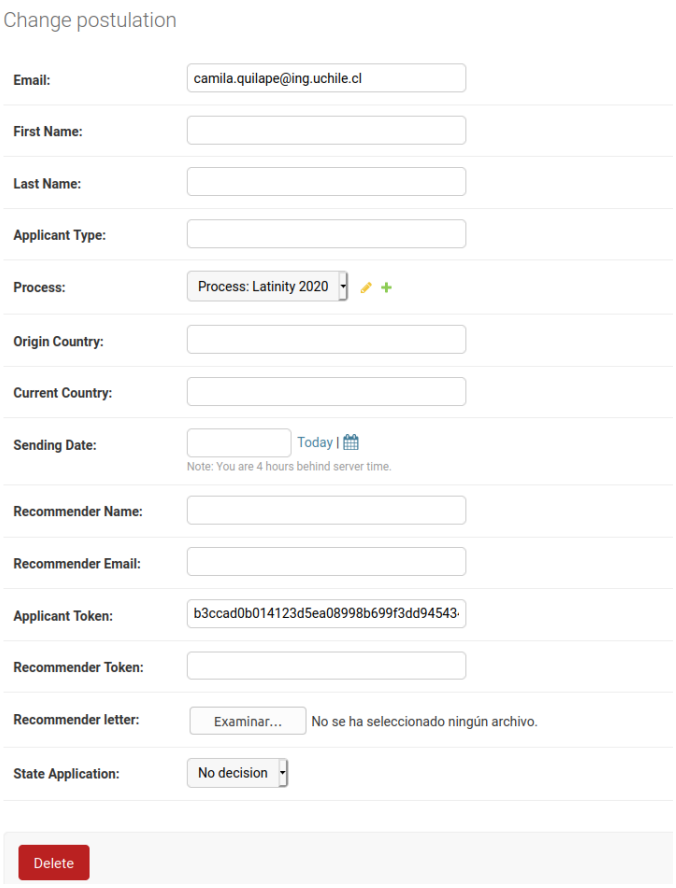

Figura 3.17: Modelo postulación a beca, desplegado en la sección de administración de Django

En la captura que se muestra en la Figura 3.18, el modelo `Proceso` además de incorporar las características que definen al proceso en sí, es decir nombre, fecha de inicio, fecha de término, estado y notificado, incluía los cuestionarios de postulación y de evaluación. Este último no se estaba utilizando.

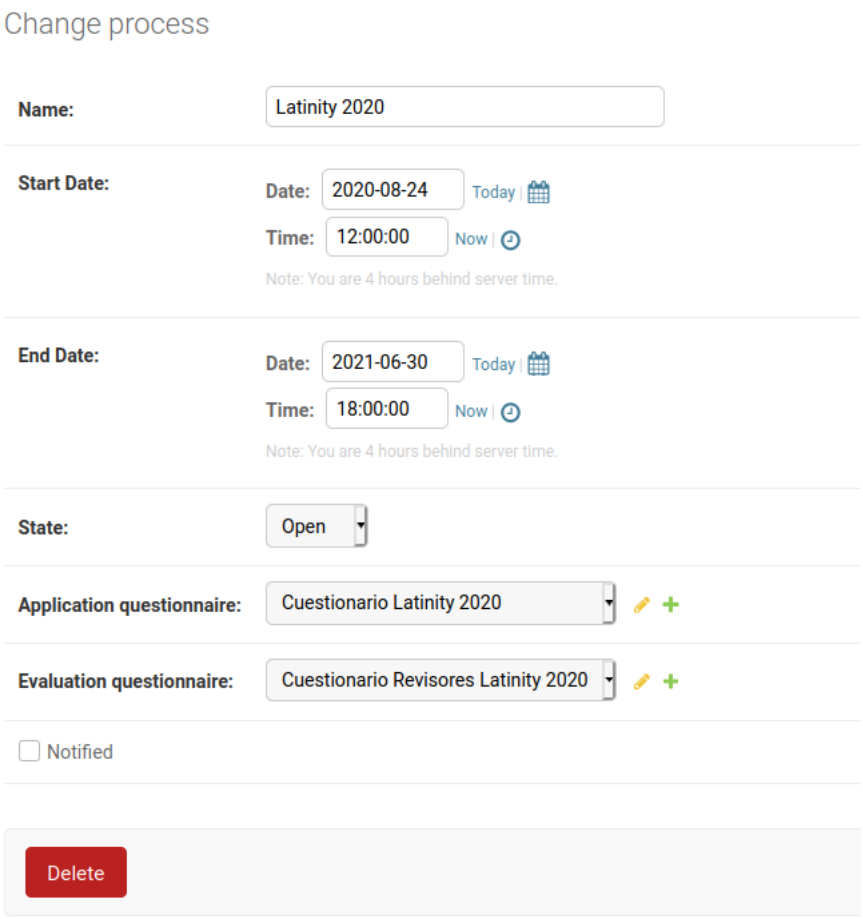

Figura 3.18: Modelo proceso de postulación a becas, desplegado en la sección de administración de Django

En la imagen que se muestra en la Figura 3.19 se aprecia que `Cuestionario` solo poseía los atributos `name` y `question`. Con `question` configurado como `many to many`, lo que provocaba que no se pudieran borrar ni seleccionar preguntas de las cargadas en la plataforma. Las preguntas se debían agregar una a una, una vez creado el cuestionario. Esto junto a lo hallado durante el análisis reafirma la necesidad de una reestructuración.

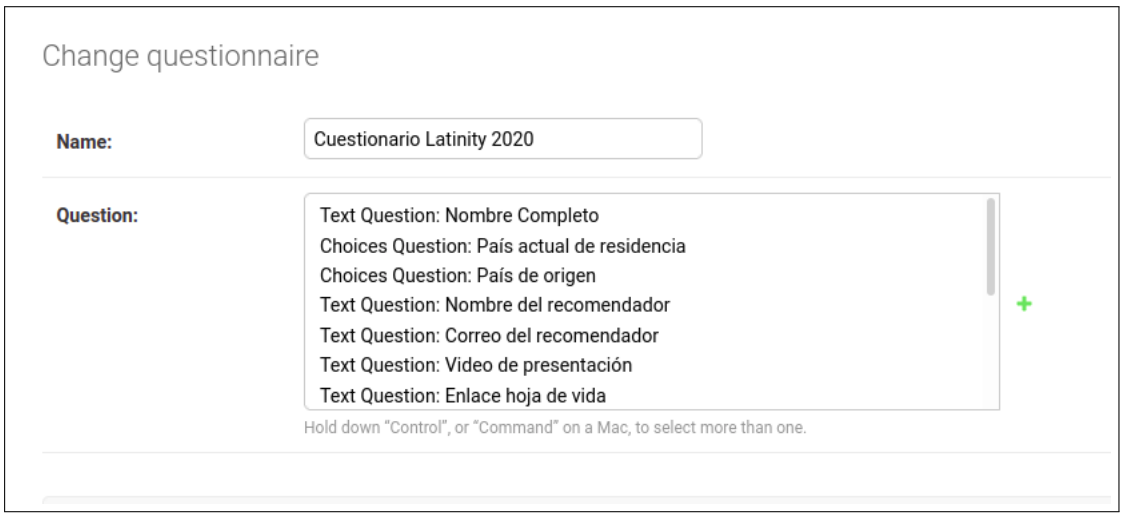

Figura 3.19: Modelo cuestionario de postulación a becas, desplegado en la sección de administración de Django

3.4. Diseño

En la Figura 3.20 se muestra el nuevo modelo de datos para el módulo de becas, el que fue diseñado por la estudiante y revisado por la profesora guía, acompañado de la descripción de las entidades. En el proceso se tomaron en cuenta todos los inconvenientes vistos previamente.

- **Process:** Representa el proceso de postulación a becas, contempla fechas, nombre y estado.
- **ApplicationRegistry:** Corresponde al modelo que almacena el token asignado al postulante, junto al proceso asociado.
- **Application:** Es la postulación en sí, tiene un estado, la fecha de la última modificación y el proceso al cual pertenece, entre otros campos.
- **Recommender:** Almacena al recomendador de una postulación y la ruta de la carta proporcionada.
- **Questionnaire:** Se trata del cuestionario habilitado para ingresar postulaciones.
- **QuestionnaireHasQuestion:** Modelo que permite vincular preguntas al cuestionario en cuestión.
- **Question:** Pregunta con campos como declaración, sección o paso en el que se mostrará la pregunta y descripción, para aportar mayor información.
- **TextQuestion:** Extiende el modelo `Question` agregando límites a las respuestas en las que se solicita texto.
- **ChoicesQuestion:** Extiende el modelo `Question` señalando la cantidad de opciones que tiene una determinada pregunta de selección.
- **ChoiceAnswer:** Especifica una alternativa a una pregunta de selección.

- **GrantProcess:** Modelo que vincula el proceso con sus cuestionarios de postulación y revisión.
- **Answer:** Almacena la respuesta a una pregunta dada.
- **Applicant:** Contiene los datos personales del postulante.

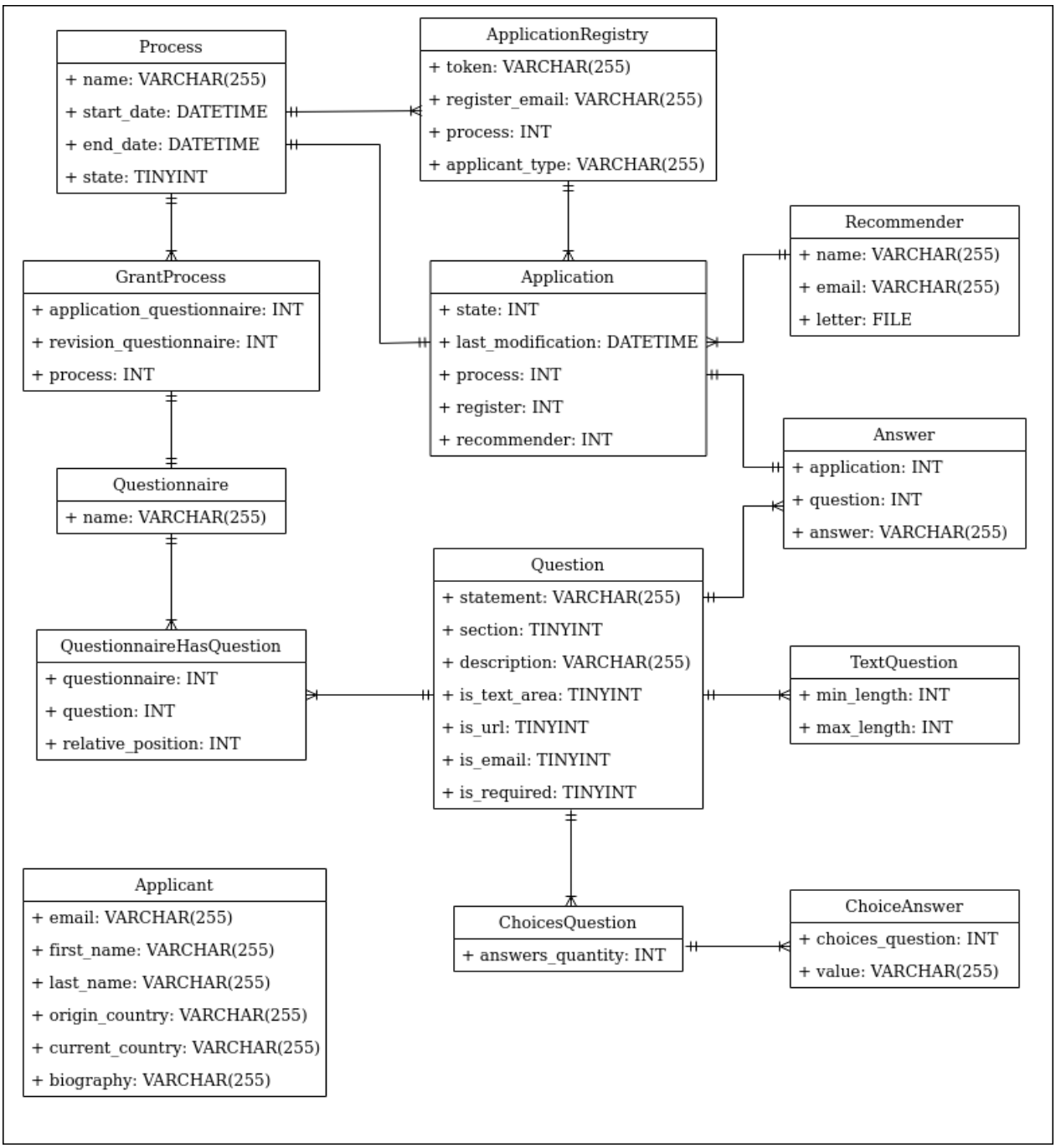

Figura 3.20: Nuevo modelo de datos para el módulo de becas

De manera similar se diseñaron los modelos faltantes, tanto para el módulo de revisión de trabajos como para la futura parte de revisión de becas. Estos modelos son `ReviewProcess`, `RevisionScholarshipAnswer` y `ReviewerHasScholarshipApplication`, y se presentan en la imagen 3.21.

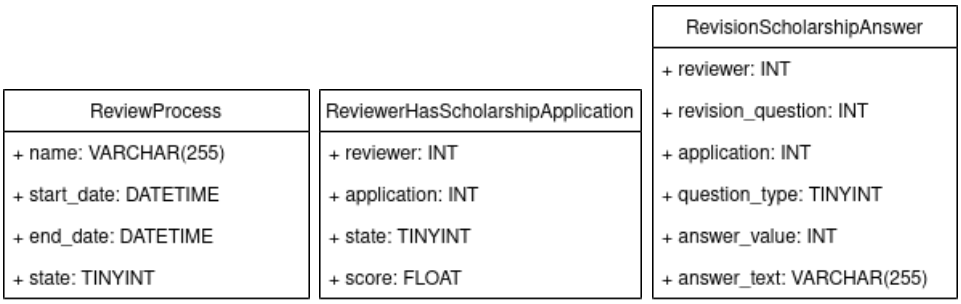

Figura 3.21: Modelos ReviewProcess, ReviewerHasScholarshipApplication y RevisionScholarshipAnswer

- **ReviewProcess:** Representa al proceso de revisiones. Abarca fechas, nombre y estado.
- **RevisionScholarshipAnswer:** Almacena la respuesta a una determinada pregunta de revisión de postulación a becas.
- **ReviewerHasScholarshipApplication:** Asocia un revisor con una postulación a becas por revisar.

Referente a la reestructuración del módulo de becas, se tomó la decisión de reimplementarlo, ya que cambiar el modelo de datos implicaba que las vistas dejaran de funcionar, y podía ser confuso ir reparándolas sobre lo ya implementado. El plan era hacer un nuevo módulo siguiendo la idea del anterior y conservando las buenas decisiones de desarrollo.

En la subsección de análisis y ésta se abordaron los primeros dos objetivos específicos. Lográndose el de análisis y evaluación completamente y el segundo de reestructuración del módulo de becas parcialmente con el diseño del nuevo modelo de datos.

Se concluye de esta etapa que la labor realizada fue útil para determinar qué era necesario agregar, ajustar y/o eliminar, contribuyendo de esta forma a una implementación más eficiente. También se espera que los resultados faciliten la mantención del sistema en el futuro.

3.5. Arquitectura del sistema

3.5.1. Arquitectura física

La arquitectura física del sistema comprende un modelo cliente servidor, donde el usuario accederá a la plataforma mediante un navegador web. Se va a conectar vía internet al servidor web Nginx, que se encontrará corriendo en un Amazon Web Service, tal como se muestra en la Figura 3.22. Se verán más detalles en el siguiente capítulo.

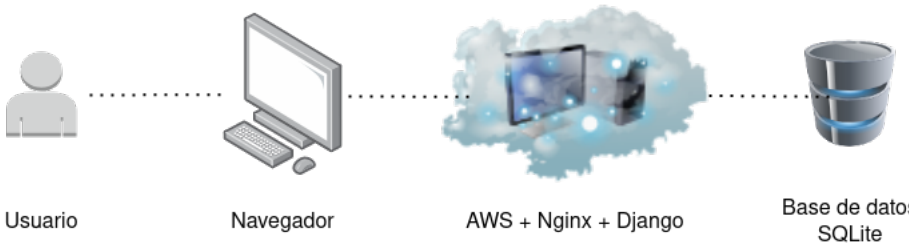

Figura 3.22: Modelo cliente servidor

3.5.2. Arquitectura lógica

Con respecto a la arquitectura lógica, el framework Django sigue el patrón de diseño Modelo Vista Controlador (MVC). MVC se usa para separar el código de las aplicaciones en tres componentes lógicas, tal como su nombre lo indica, modelo, vista y controlador. Cada una con una tarea definida.

- Modelo: Representa la información que maneja el sistema. En esta capa se trabaja con los datos, por lo que contiene mecanismos o funciones para acceder, crear, modificar o eliminar datos. Aquí va la lógica de negocio.
- Vista: Contiene el código que va a generar la visualización de las interfaces de usuario, de modo que no accede directamente a los datos. Incorpora la lógica de presentación o front-end.
- Controlador: Es una capa que sirve de enlace entre el modelo y la vista. Incluye el código preciso para responder a las acciones que se solicitan en el sitio. Aquí va la lógica de la aplicación.

MVC surge ante la necesidad de producir software más robusto y con separación de conceptos, por ende su objetivo es ayudar a generar aplicaciones de mejor calidad. Entre sus ventajas resalta el facilitar el mantenimiento y la reutilización de código.

El flujo de trabajo, que se exhibe en el diagrama de la Figura 3.23, es el siguiente. El usuario realiza una solicitud al sitio, como por ejemplo acceder a cierta página, solicitud que llega al controlador. Luego, el controlador ordena ejecutar la operación de datos correspondiente a los modelos. Una vez que los modelos terminan su labor, el controlador le pasa los datos a la vista si compete y le solicita la renderización, generando de esta manera la respuesta que se le envía al usuario.

En particular, Django utiliza una variación de MVC, donde los modelos se mantienen, pero el controlador es llamado vista y la vista template. Además existe un paso previo, donde la url de petición del usuario se compara a la de los archivos `urls.py`. Estos archivos detallan la vista encargada de atender cada tipo de solicitud.

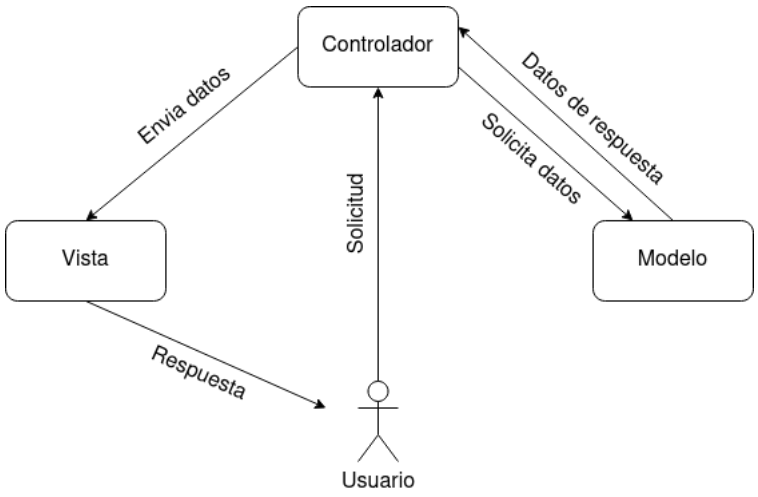

Figura 3.23: Modelo Vista Controlador

Capítulo 4

Implementación

En este capítulo se va a detallar la implementación del software, basada en el análisis y el diseño del capítulo anterior. Las secciones que siguen, contemplan desde la metodología y resolución de problemas, hasta la puesta en marcha en un ambiente formal.

En concreto, las secciones de este capítulo son: Metodología, Resolución de problemas, Reestructuración de becas, Revisión de becas, Unificación de las secciones de administración, Gestión de revisores, Resúmenes e implementaciones varias, Uniformización, Fixtures y Despliegue.

A continuación se precisan las decisiones de diseño, las funcionalidades desarrolladas y la conexión entre lo que ve el usuario y el back-end. Además se especifican los inconvenientes presentados en el proceso, junto a aspectos difíciles de llevar a cabo. Adicionalmente, se incluyen fragmentos de código, para ejemplificar el uso de las tecnologías, y capturas de partes relevantes del proyecto.

4.1. Metodología

Referente a la metodología de trabajo seguida para implementar la solución, se acerca a una metodología de desarrollo incremental, pero no cabe del todo en esta categoría, ya que todas estas formas de trabajo se orientan al trabajo en equipo.

Lo que se hizo en la práctica fue intentar seguir el plan de trabajo establecido, desagregando las etapas en tareas pequeñas y distribuyéndolas a lo largo de las semanas. Por otro lado, el tiempo de desarrollo diario se fue contabilizando con pomodoros, lo que contribuyó a cumplir las horas semanales de dedicación a la memoria.

En cuanto a seguimiento del trabajo, se tuvieron reuniones semanales con la profesora guía, quién además es una de las organizadoras de Latinity. Fue la persona a la cual se le pudieron realizar las preguntas y validar las implementaciones, por lo que se tuvo retroalimentación constante.

4.2. Resolución de problemas

Como parte de la etapa de resolución de problemas, se corrigieron las condiciones y los estados de los procesos, de acuerdo a lo señalado en el capítulo anterior. También se ajustaron las consultas a la base de datos, ejecutadas mediante filtros de Django, agregándoles los argumentos faltantes. En el Listado 4.1 se muestra un ejemplo de filtro sobre el proceso.

```
work = Work.objects.get(application_type=application_type, process_id=
    process)
```

Listado 4.1: Ejemplo de filtro de Django

Asimismo se realizó una limpieza general del proyecto. A nivel de interfaces se redujó el número de templates base, actualmente hay uno por módulo, para becas, trabajos y revisiones, y otro para la sección de administración unificada. Un template base es un archivo que mezcla código HTML con código Python incrustado a través de etiquetas de Django, que contiene las dependencias y componentes comunes a un grupo de templates, como el header y el footer.

Igualmente se eliminaron las dependencias que no se usaban y se cargaron en el proyecto las que sí, agregando los archivos correspondientes. De este modo se evitaron los links a páginas externas, que podían cambiar con el tiempo. Estos posibles cambios podían provocar alteraciones visuales, acciones inesperadas o incluso que la plataforma dejara de funcionar.

A nivel de datos, se limpiaron los modelos existentes en trabajos y revisión de trabajos, eliminando los campos que no estaban siendo utilizados. También se añadieron características a algunos campos o columnas de los modelos, precisando si podían quedar en blanco o si existían valores por defecto.

Por ejemplo en el Listado 4.2, el campo `is_required` de `RevisionQuestion`, modelo que representa a una pregunta de la revisión de postulaciones, es `True` por defecto.

```
class RevisionQuestion(PolymorphicModel):
    statement = models.CharField(
        _('Statement'),
        max_length=255,
        null=False,
    )
    description = models.CharField(
        _('Description'),
        max_length=255,
        null=True,
        blank=True,
    )
    question_type = models.IntegerField(
        _('Question Type'),
        choices=[
            (0, "Value Question"),
            (1, "Agreement Question"),
            (2, "Text Question")
        ],
        null=False,
    )
```

```
22      is_required = models.BooleanField(
23          _('Is Required'),
24          null=False,
25          default=True,
26      )
```

Listado 4.2: Modelo `RevisionQuestion`

Una de las tareas más desafiantes de esta fase fue corregir los errores del paso dos de postulación a trabajos, en el que se deben especificar los integrantes del grupo al cual pertenece el trabajo, o que van a participar de él (ver la Figura 3.9). Para lograrlo fue necesario partir con el estudio de la documentación de formsets de Django, debido a que era el concepto que se estaba aplicando y que la memorista no lo manejaba. Un formset es una capa de abstracción que permite trabajar con múltiples formularios en una misma página.

Posteriormente, luego de leer el código asociado a este paso, se notó que faltaban parámetros iniciales al momento de llamar a las funciones, los que debían controlar la cantidad de formularios, y los mínimos y máximos (ver el Listado 4.3).

```
1 formset = formset_factory(SaveApplicationForm, extra=8, min_num=1,
      validate_min=True, max_num=8, validate_max=True)
2 group_formset = formset(form_kwargs={'questions': questions}, initial =
      initial_data)
3 app_form = SaveApplicationForm(questions=questions)
4 formset.empty_form = app_form
```

Listado 4.3: Parte del contenido de la función get de la vista `ApplicantMembers`

Por otra parte, asimismo se advirtió que no se estaba usando el campo apellido cuando se guardaban los datos de un postulante. En su lugar, se utilizaba el campo nombre para almacenar el nombre completo, de manera que se aprovechó de corregir eso.

Estudiando el comportamiento de los formularios del módulo de trabajos, también se reparó que cada vez que se pasaba por el segundo paso de postulación, usando los botones `paso anterior`/ `paso siguiente`, se eliminaban todos los objetos integrantes y se creaban de nuevo, incluso si no se editaban los campos. Esto hacía al sistema lento e ineficiente en términos de almacenamiento.

Lo anterior se solucionó separando las posibles acciones del usuario en casos (ver el Listado 4.4), los que son agregar miembros (línea 9), editar miembros (línea 7) o eliminar miembros (línea 11). Entonces en el back-end se compara el número de miembros en un grupo dado con la cantidad de formularios que vienen desde el template, con el propósito de determinar el tipo de operación efectuada por el postulante. Por tanto, ahora solo se generan, borran o modifican los objetos precisos.

```
if group_formset.is_valid():

    applicants = GroupHasApplicant.objects.filter(group_id=group).
values_list("applicant_id", flat=True)
    members = len(applicants)
    forms = int(request.POST.get("form-TOTAL_FORMS"))

    if members == forms:
        # Se mantuvieron o editaron los integrantes
    elif forms > members:
        # Se mantuvieron o editaron members integrantes y se agregaron
forms-members integrantes
    else:
        # Se mantuvieron o editaron forms integrantes y se eliminaron
members-forms integrantes
```

Listado 4.4: Pseudocódigo de la lógica de la función post de la vista encargada del paso 2 de postulación a trabajos

Otro problema que tenía el módulo de trabajos ocurría cuando ya se habían contestado ciertos campos de los formularios. Para mostrar la información, la vista recuperaba las respuestas de la base de datos y las pasaba al template mediante un diccionario o una lista. Luego, una vez cargado el formulario en la página, se ejecutaba código JavaScript para desplegar los datos en las ubicaciones correspondientes.

La implementación previa, a pesar de ser ineficiente, no causaba inconvenientes localmente. No obstante, cuando se corrió el software en la máquina de Amazon se observó que pasaban varios segundos entre que se dibujaban los formularios y aparecían las respuestas.

El problema se corrigió utilizando funcionalidad nativa de Django. Al momento de instanciar formularios se agregó el campo extra `initial` con los datos iniciales incluidos en una lista de diccionarios, tal como aparece en la línea 2 del Listado 4.3. La forma en que se define la variable lista `initial_data` se exhibe en el Listado 4.5. De modo que es Django quién se encarga de presentar la información de manera adecuada y al mismo tiempo que los formularios.

```
group_has_applicant = GroupHasApplicant.objects.filter(group_id=group)
initial_data = []
for i, member_has_group in enumerate(group_has_applicant):
    member = member_has_group.applicant_id
    dict_applicant = {}
    for q in questions:
        applicant_answer = ApplicantAnswer.objects.get(application=
application, question=q, applicant=member)
        dict_applicant[f'{q.statement}'] = applicant_answer.answer

    initial_data.append(dict_applicant)
```

Listado 4.5: Ejemplo de definición de variable con datos iniciales. En este caso la del paso 2 de postulación a trabajos

4.3. Reestructuración de becas

Con respecto a la reestructuración de becas, lo primero que se hizo fue generar los submódulos `scholarship_applications` y `scholarship_admin`. Posteriormente, se plasmaron en los archivos `models.py` los modelos de datos diseñados(ver la Figura 3.20) y se construyeron las vistas para cada sección a la que puede acceder el postulante. En concreto, para el login (ver la Figura 4.1), la página de inicio (ver la Figura 4.2), el formulario, la página de resumen (ver la Figura 4.6) y la de recuperación del token.

Para ingresar una postulación se debe completar el formulario de postulación, el que aparece al hacer click sobre el botón `Nueva postulación` en la página de inicio (ver la Figura 4.2). Este formulario está asociado principalmente a las tablas `Application`, `Answer`, `Applicant` y `Recommender`.

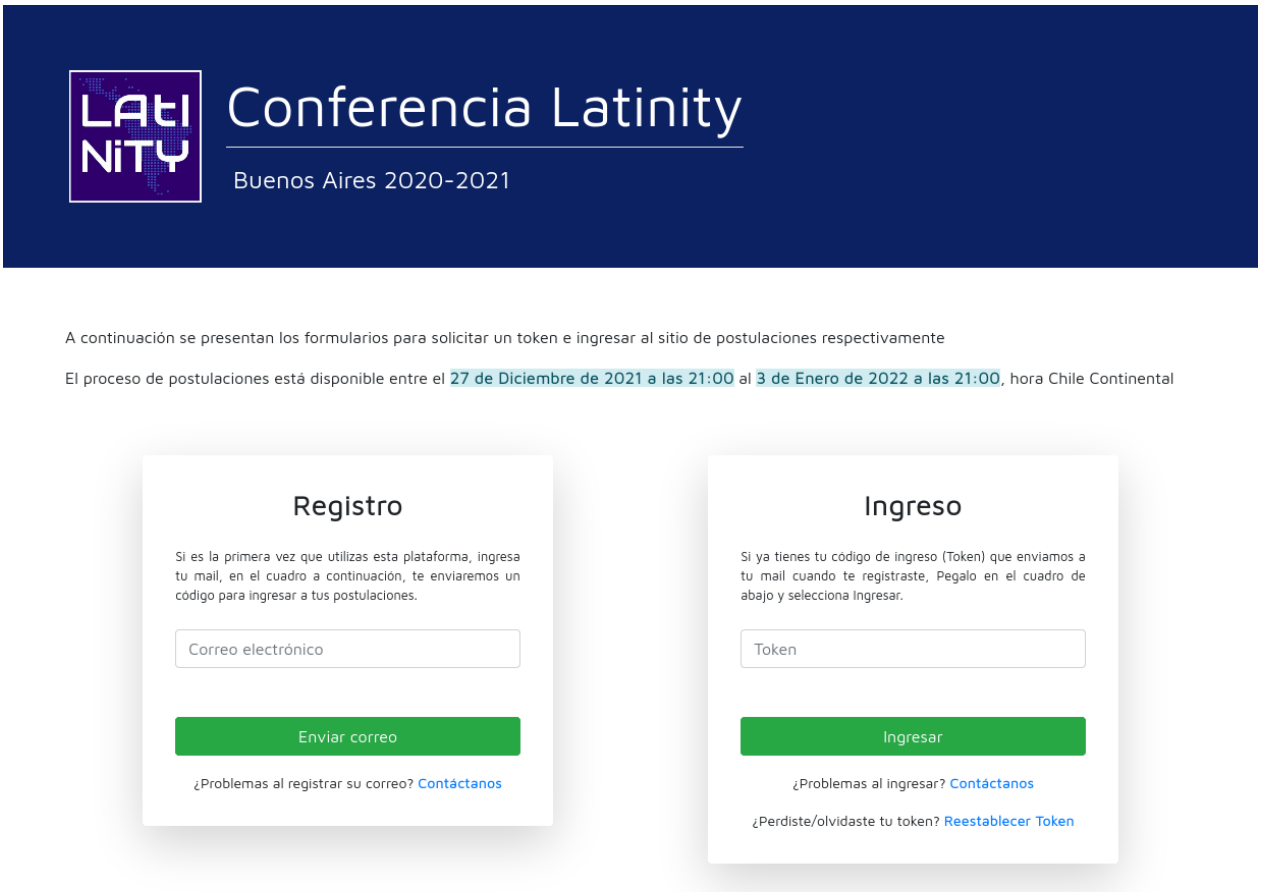

Figura 4.1: Página de autenticación de postulación a becas

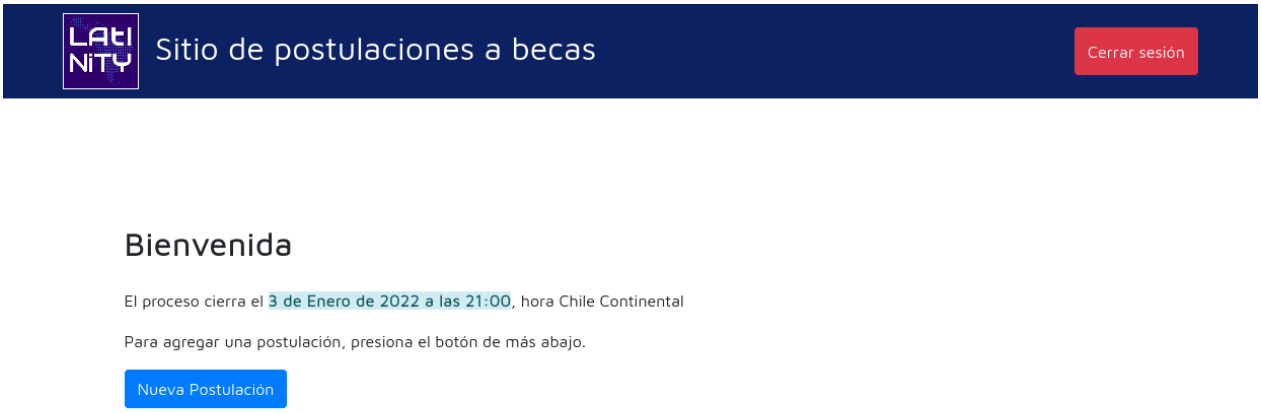

Figura 4.2: Página de inicio de postulación a becas

Es importante señalar que en becas solo se puede postular una vez por proceso, por lo que al iniciarse la postulación desaparece el botón `Nueva postulación`. En su lugar se expone una tabla con el identificador, el correo del postulante, la fecha de la última modificación y el estado, ya sea incompleta o completa. Luego de que se toma un veredicto, la celda estado exhibe el resultado de la postulación, es decir si fue aceptada, rechazada o está en lista de espera.

Cuando el usuario presiona el identificador, el sistema lo redirige al formulario. Ahí puede ver lo respondido hasta el momento, y completar o editar la postulación, todo esto mientras el proceso de postulación a becas se encuentre abierto. En cambio, si está cerrado, seleccionar el identificador despliega una tabla resumen con las respuestas ingresadas.

El formulario de becas tiene tres pasos. En el primero se solicitan los datos personales (ver la Figura 4.3), en el segundo va la información del recomendador, esto es nombre, correo y carta (ver la Figura 4.4), y en el tercero se deben especificar los detalles que respalden la solicitud, tales como motivo, hoja de vida y vídeo de presentación (ver la Figura 4.5). La etapa dos solo es requerida si el postulante es estudiante de pregrado o de una carrera técnica.

Al hacer click sobre el botón `Enviar` del paso 3, se genera una solicitud de tipo post a la vista del formulario. La vista valida que todos los campos obligatorios de los tres pasos estén completos y si es así, lleva a la página de resumen (ver la Figura 4.6). En la página resumen se presenta el detalle de cada respuesta y es posible enviar efectivamente la postulación presionando `Confirmar envío`.

Lo más complicado de esta etapa fue la implementación de la vista que se encarga de estos pasos, debido a los casos que se debían tomar en cuenta, a las validaciones necesarias y al funcionamiento de los botones. Esta vista, llamada `ApplicationFill`, selecciona las preguntas a mostrar a partir de los modelos `Questionnaire`, `QuestionnaireHasQuestion` y `Question`, estableciendo la condición campo `section` igual a paso.

Por otro lado, `ApplicationFill` maneja el almacenamiento y la edición de las respuestas ingresadas por el postulante, creando o actualizando los objetos correspondientes. Un ejemplo de su funcionamiento es que, si en el paso 1 se responde `Tipo de postulante` y se presiona `Paso siguiente`, la vista debe guardar y revisar la respuesta, con el fin de determinar si exhibe o no el paso 2.

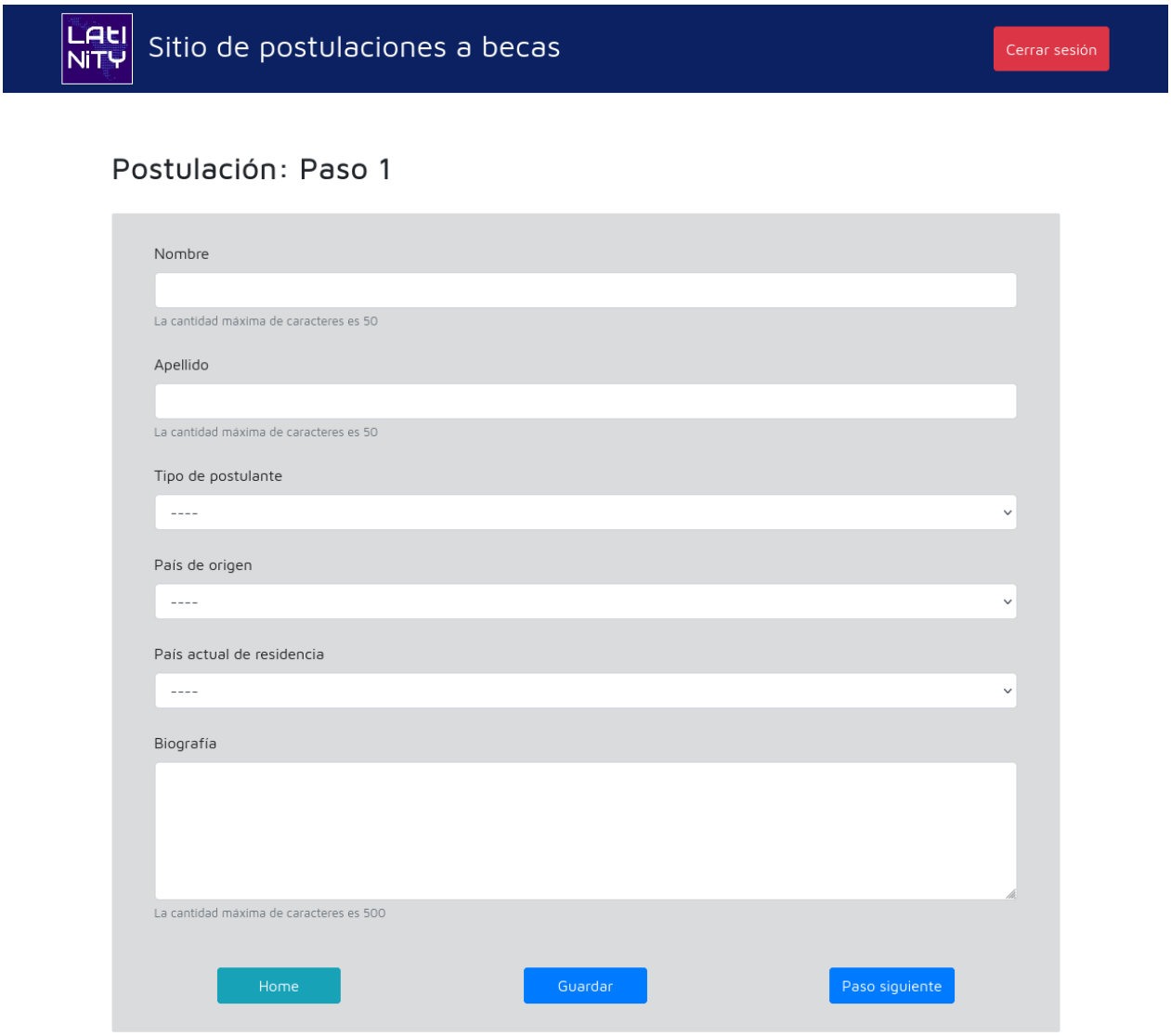

Figura 4.3: Paso 1 del formulario de postulación a becas

Figura 4.4: Paso 2 del formulario de postulación a becas

El identificador de la postulación, el número del paso y el token se pasan de una página a otra a través de la url `.../fill/<str:token>/<int:step>/<str:application_id>`. Corresponden a los argumentos de las funciones `get` y `post` de la vista `ApplicationFill`.

Además de los problemas mencionados en el capítulo de Análisis y Diseño, antes se rompía el flujo de navegación en el paso 2. Vale decir, para subir la carta de recomendación se llevaba al usuario a una página aparte, donde solo se encontraba ese formulario (ver la Figura 3.6). Ahora el formulario de la carta es parte del formulario principal (ver la Figura 4.4).

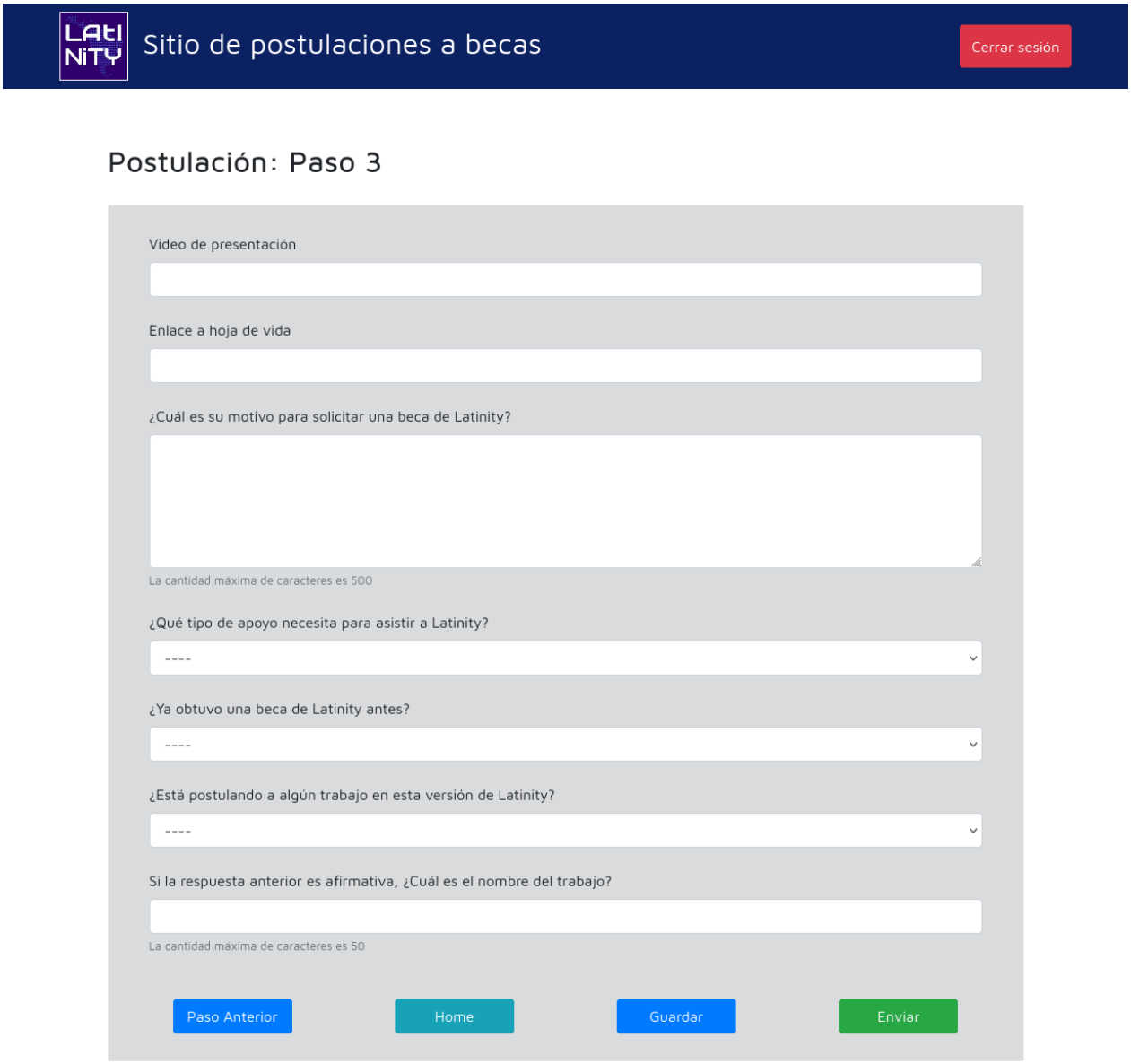

Figura 4.5: Paso 3 del formulario de postulación a becas

Relacionado a lo anterior, se configuró el modelo `Recomendador` (ver el Listado 4.6) para que también guardara la ruta de la ubicación del archivo carta y su nombre original. Asimismo, en el directorio de archivos del proyecto, se estableció que la carta se almacenara con el nombre del postulante concatenado al identificador de la postulación. Estos cambios surgieron una vez avanzada la implementación, ante la necesidad de desplegar la carta de manera exitosa a los diferentes usuarios de la plataforma.

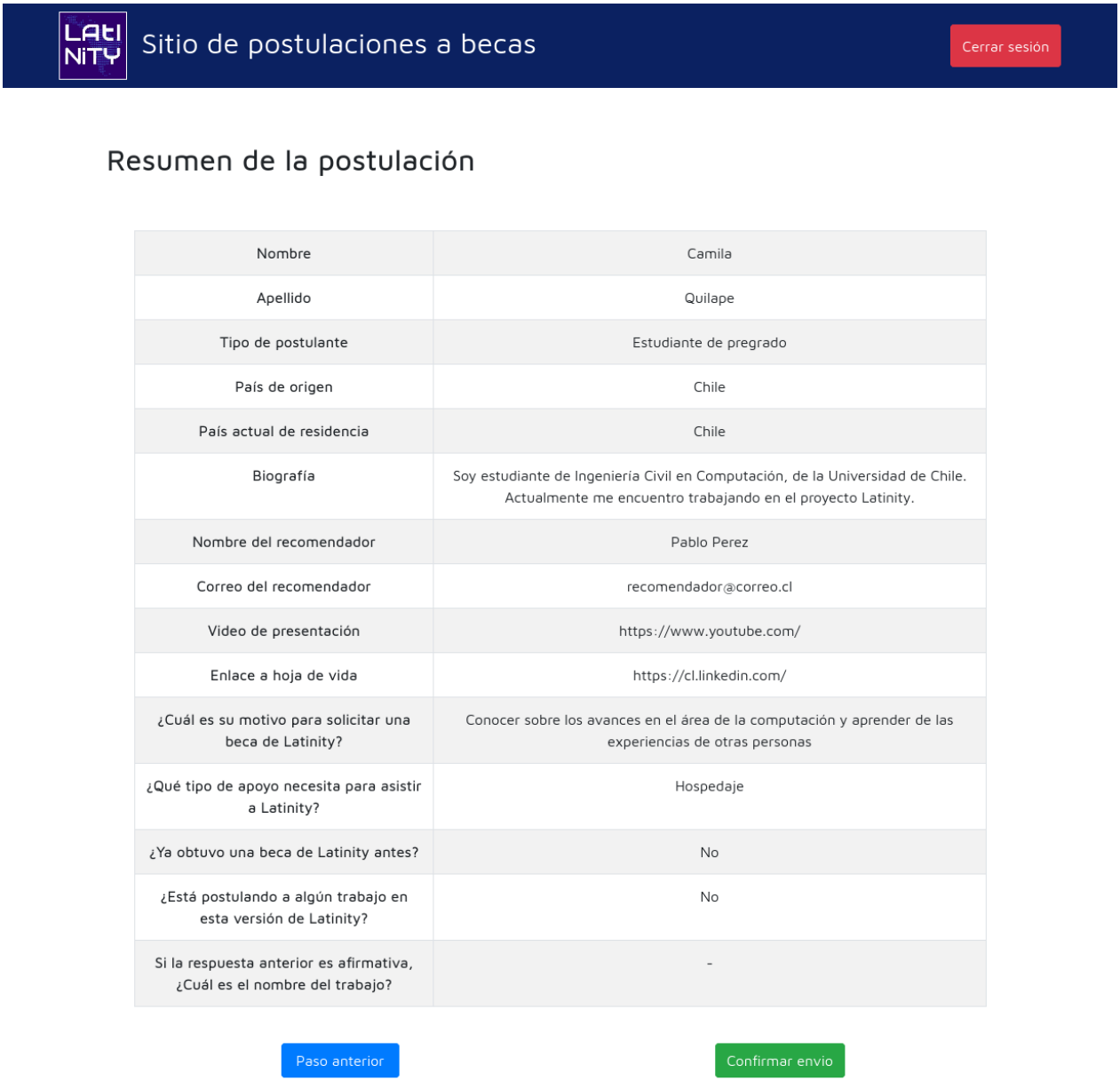

Figura 4.6: Página de resumen de postulación a becas

```
class Recommender(models.Model):
    name = models.CharField(
        _('Name'),
        max_length=255
    )
    email = models.EmailField(
        _('Email'),
        null=False,
```

```
        default=None
    )
    letter = models.FileField(
        upload_to='letters/',
        validators=[FileExtensionValidator(["pdf"])],
        default = '',
    )
    original_letter = models.CharField(
        _('Original letter'),
        max_length=255,
        null=True,
        blank=True
    )
```

Listado 4.6: Modelo Recomendador

Junto al desarrollo del formulario de becas, se corrigió el de trabajos. Las implementaciones en ambos módulos fueron iterativas, es decir, se partió con funcionalidades simples, a las que se les fueron incorporando nuevas características y corrigiendo otras, a medida que avanzaban las semanas. Al principio, para pasar de un paso a otro, o hacer efectiva la acción de un botón, se debían completar todos los campos obligatorios del paso correspondiente.

Luego, lo anterior evolucionó a un guardado parcial, donde ya no era necesario rellenar todo. Se quería esta función para que el postulante pudiera conocer íntegramente lo que se le solicita, sin la obligación de completar un paso dado para ver el siguiente. No obstante, se perdieron las validaciones. Si no se ingresaba lo que se debía, por ejemplo un correo o enlace, simplemente no se guardaba, sin entregar feedback al usuario.

Posteriormente, se consiguió solucionar el tema de las validaciones, pero seguían bloqueados los botones `Home`, `Paso anterior` y `Paso siguiente`. Esto significa que presionar cualquiera de ellos, provocaba que el software revisara el contenido de las respuestas, si es que había alguna.

`Home` y `Paso anterior` se dejaron totalmente libres, redirigiendo a las páginas respectivas. Por otro lado, `Paso siguiente` quedó parecido a `Guardar`, con la diferencia que `Guardar` hace que se recargue la misma página, mientras que `Paso siguiente` lleva a la que sigue.

Desde el punto de vista de la administración, en el submódulo `scholarship_admin` se incluyó una página de resumen de las postulaciones a becas (ver la Figura 4.7). Esta página contiene una tabla con los datos más relevantes de las postulaciones, tales como estado, tipo de postulante y tipo de apoyo. La tabla se generó usando DataTables [12], un plug-in de la librería JQuery que permite paginar tablas, ordenar columnas, y realizar búsquedas.

La tabla resumen presenta información de todas las postulaciones del proceso actual, independiente de sus estados. Dicha información se recupera en su mayoría de la tabla `Answer`. Asimismo la tabla tiene asociada la opción de descargar todos los datos, ya sea en formato texto plano, csv o Excel.

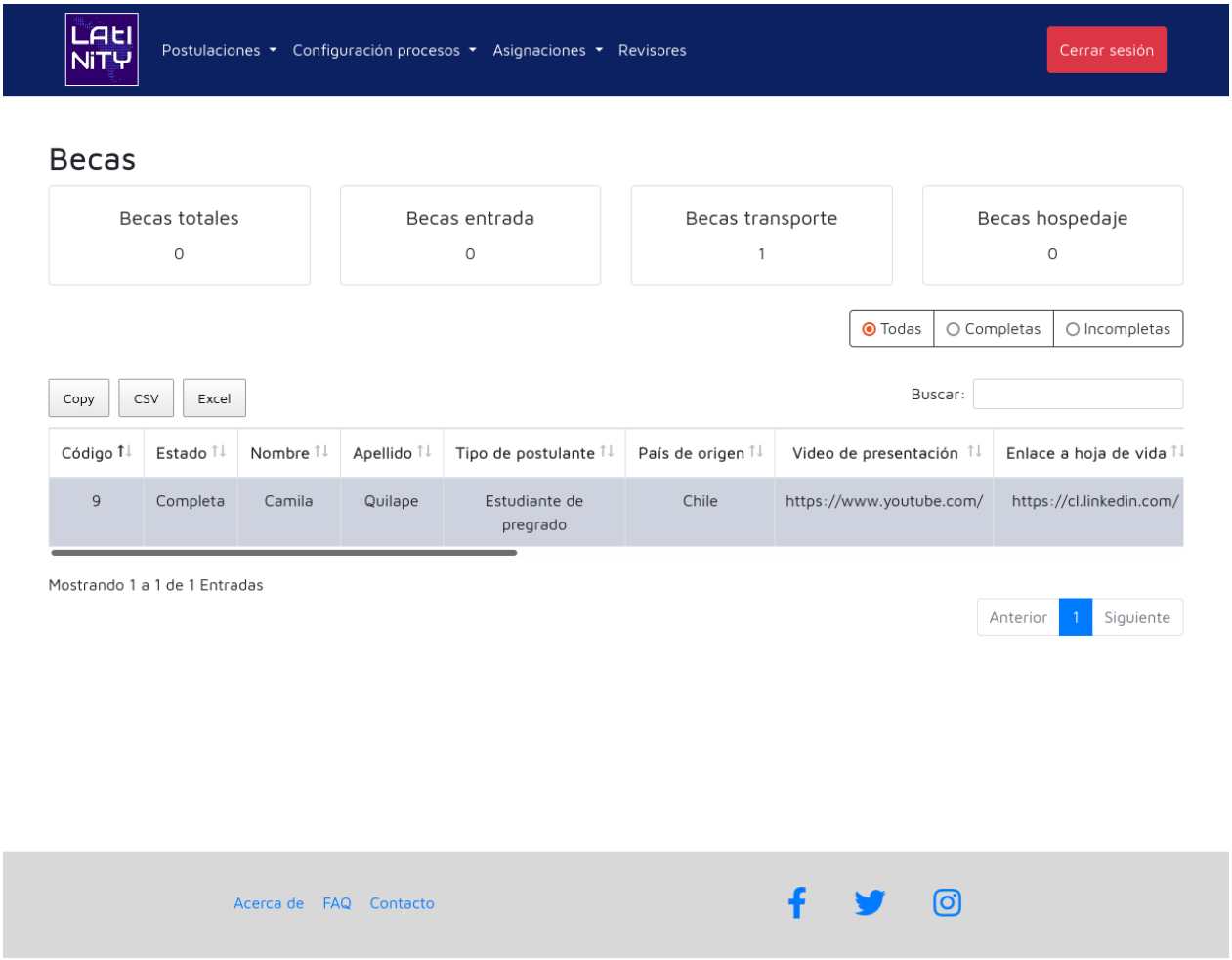

Figura 4.7: Página de resumen de postulaciones a becas, vista por el administrador

También se incorporó en este submódulo la posibilidad de configurar las fechas del proceso de postulación a becas (ver la Figura 4.8). Tanto para la de inicio como para la de fin se puede desplegar un calendario que facilita la selección del día, mientras que la hora solo puede ser cambiada manualmente. Esta página se dejó similar a la contenida en el módulo de trabajos.

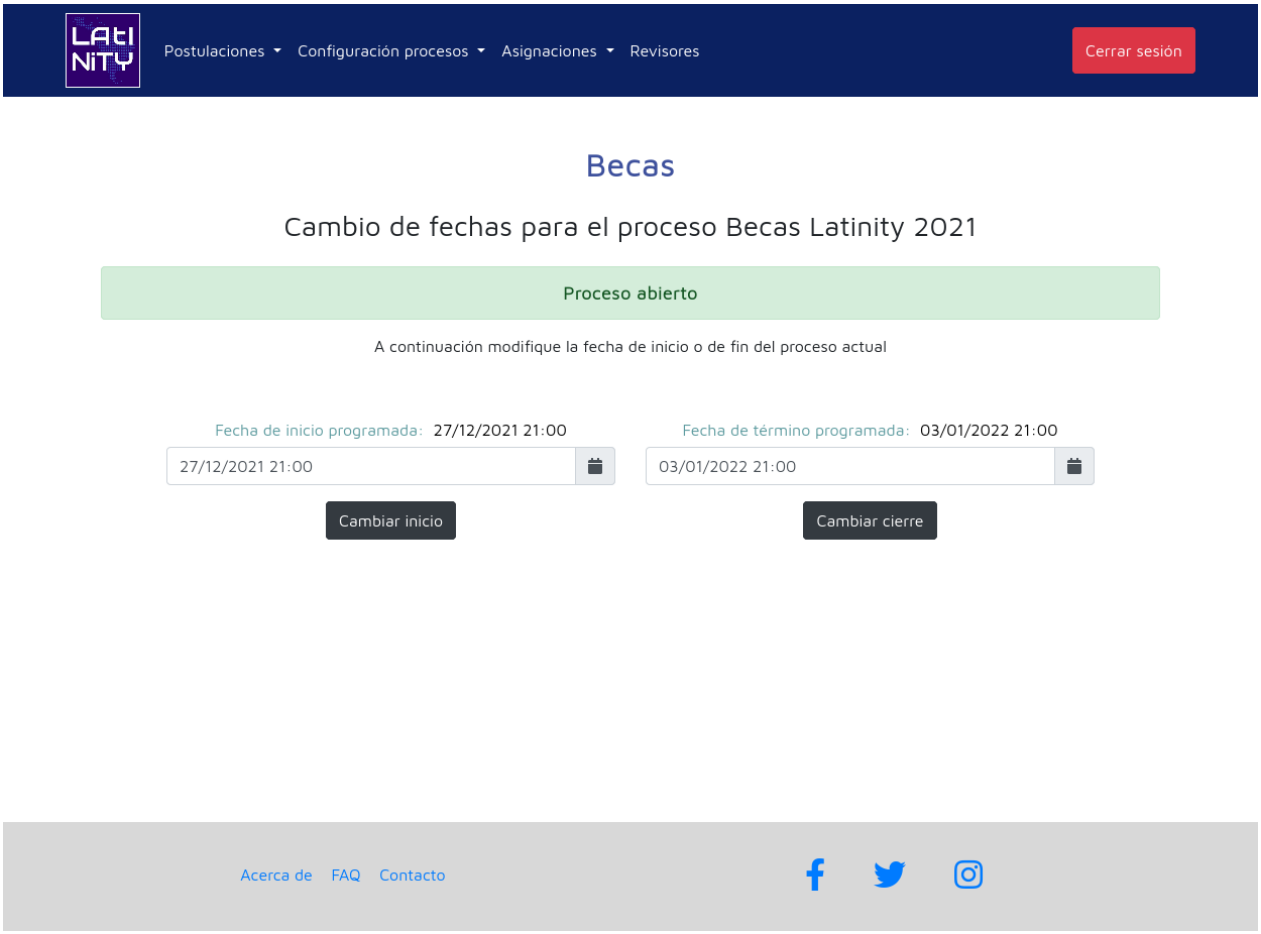

Figura 4.8: Configuración del proceso de postulación a becas

Lo realizado en esta etapa termina de abordar lo necesario para cumplir el segundo objetivo específico de *Reestructurar el módulo de becas, comenzando con el diseño de un nuevo modelo de datos.*

4.4. Revisión de becas

Lo primero a mencionar de la parte de revisión de becas son los nuevos modelos. Se crearon tres con sus respectivos campos, `ReviewerHasScholarshipApplication` para la asociación entre un revisor y una postulación a beca, `RevisionScholarshipAnswer` para almacenar la respuesta de un revisor a una determinada pregunta de revisión de becas y `ReviewProcess` para controlar los plazos de revisión, como representación del proceso de revisión de postulaciones.

En este caso no se implementaron submódulos nuevos, los modelos diseñados previamente `ReviewerHasScholarshipApplication` y `RevisionScholarshipAnswer` se definieron en el archivo `models.py` de `review_reviewers`, mientras que `ReviewProcess` se incluyó en el de `review_admin`. Sus diseños se presentaron en la Figura 3.21 del capítulo anterior.

Se comenzó esta etapa integrando el acceso a revisión de becas a la página inicial de un revisor, como una nueva tabla con las postulaciones a becas por revisar (ver la Figura 4.9). De manera que, ahora el revisor ve dos tablas, la de trabajos con las columnas Código, Nombre, Modalidad, Track, Puntaje y Estado, y la de becas con Código, Correo, Puntaje y Estado.

Figura 4.9: Página de inicio del revisor

En términos de back-end, fue pertinente extender la vista `PanelReviewer`, la que se encarga de la página inicial, y agregar la vista `ReviewScholarshipApplication` para manejar la revisión de becas. Seleccionar una postulación en la página de inicio, redirige a la página de revisión (ver la Figura 4.10), donde se muestra el detalle de la postulación y el formulario de revisión. El formulario depende del tipo de postulación, es decir, si se trata de una beca, un póster, una presentación o un círculo de mentoría.

El formulario de revisión se dejó con un funcionamiento similar a los de postulaciones, se permite el guardado parcial. Sin embargo, como mejora al diseño de la interfaz de usuario, una vez enviada la revisión se elimina el botón `Guardar`, con el fin de evitar confusiones. Si se quiere editar la revisión, basta con cambiar las respuestas y clickear `Enviar` nuevamente.

Cabe señalar que la sección que presenta el detalle de la postulación, la mitad de la izquierda en la Figura 4.10, en realidad está siendo renderizada por un template que se incluye en el que despliega el formulario de revisión (ver el Listado 4.7). Esta decisión se tomó para reutilizar el template de detalle en la sección de administración.

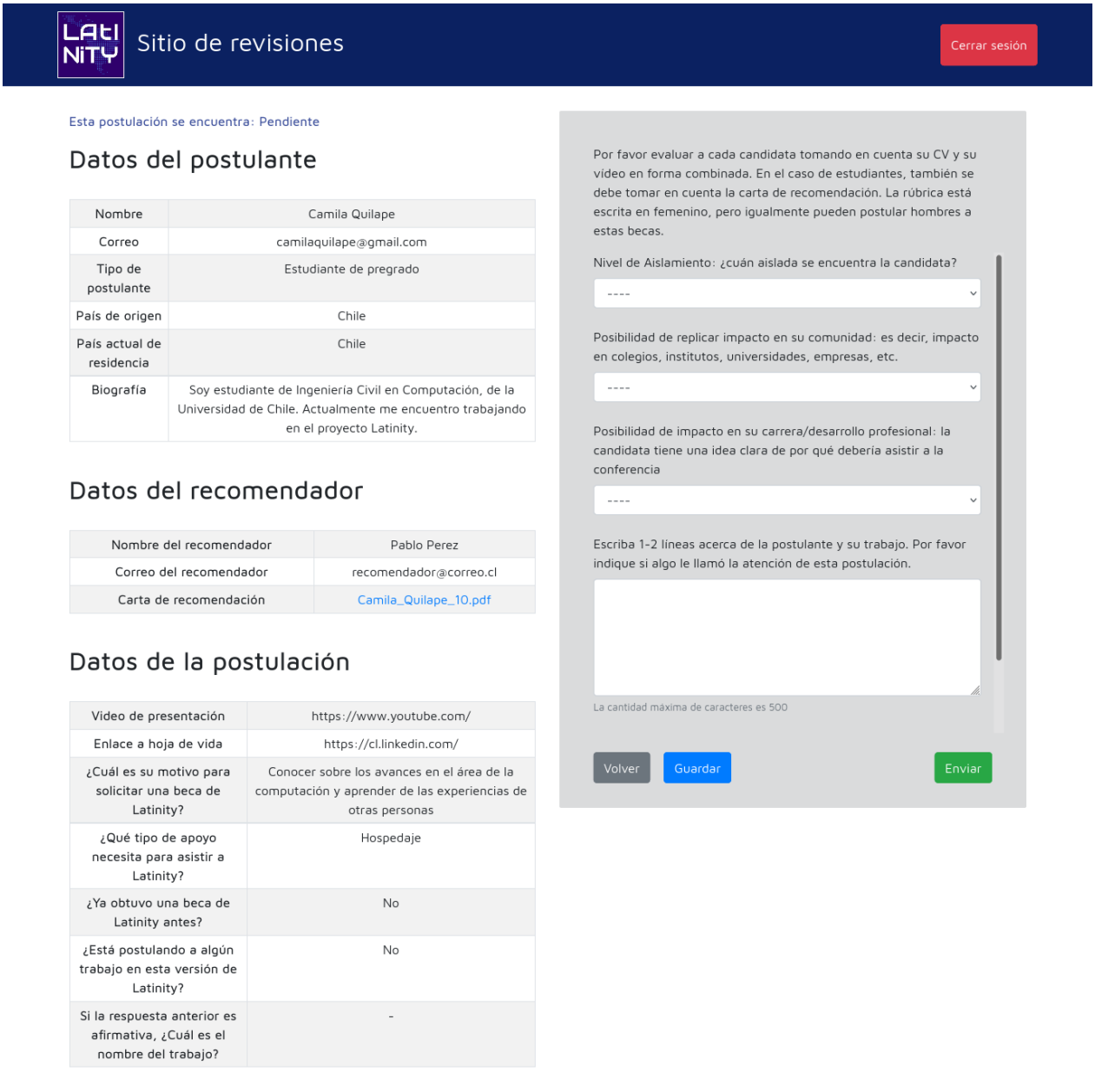

Figura 4.10: Página de revisión de una postulación a becas en particular

```
<div class="col">
    <p>Esta postulacion se encuentra: {{ state }}</p>
    {% include "application_details.html" %}
</div>
```

Listado 4.7: Extracto del template `review_revision.html`

Por otro lado, en revisión de trabajos se notó que la vista `ReviewApplication` le estaba enviando al template la información de una postulación a través de un diccionario, lo cual era un problema, ya que se recuperaban las respuestas en el template usando como llaves las preguntas del formulario de postulación. En consecuencia, si se modificaba una pregunta en el admin de Django y no se tomaba la precaución de actualizarla en el código, se caía el sitio.

Dado lo anterior, se cambió la manera de pasar los datos a una lista de tuplas, con el primer valor la pregunta y el segundo la respuesta. En tanto, la parte de revisión de becas se implementó desde el principio siguiendo la idea de las tuplas.

Como observación, también se agregaron restricciones para que un revisor no pueda evaluar una postulación si el proceso de becas o trabajos sigue abierto, pues en esos casos los postulantes todavía pueden cambiar sus respuestas. Dando así un mejor control de esta sección del sitio.

Lo recién mencionado corresponde a las páginas que ve un revisor o revisora. Desde el punto de vista del administrador se desarrolló el panel de asignaciones a postulaciones a becas (ver la Figura 4.11) y el detalle de una revisión (ver la Figura 4.13).

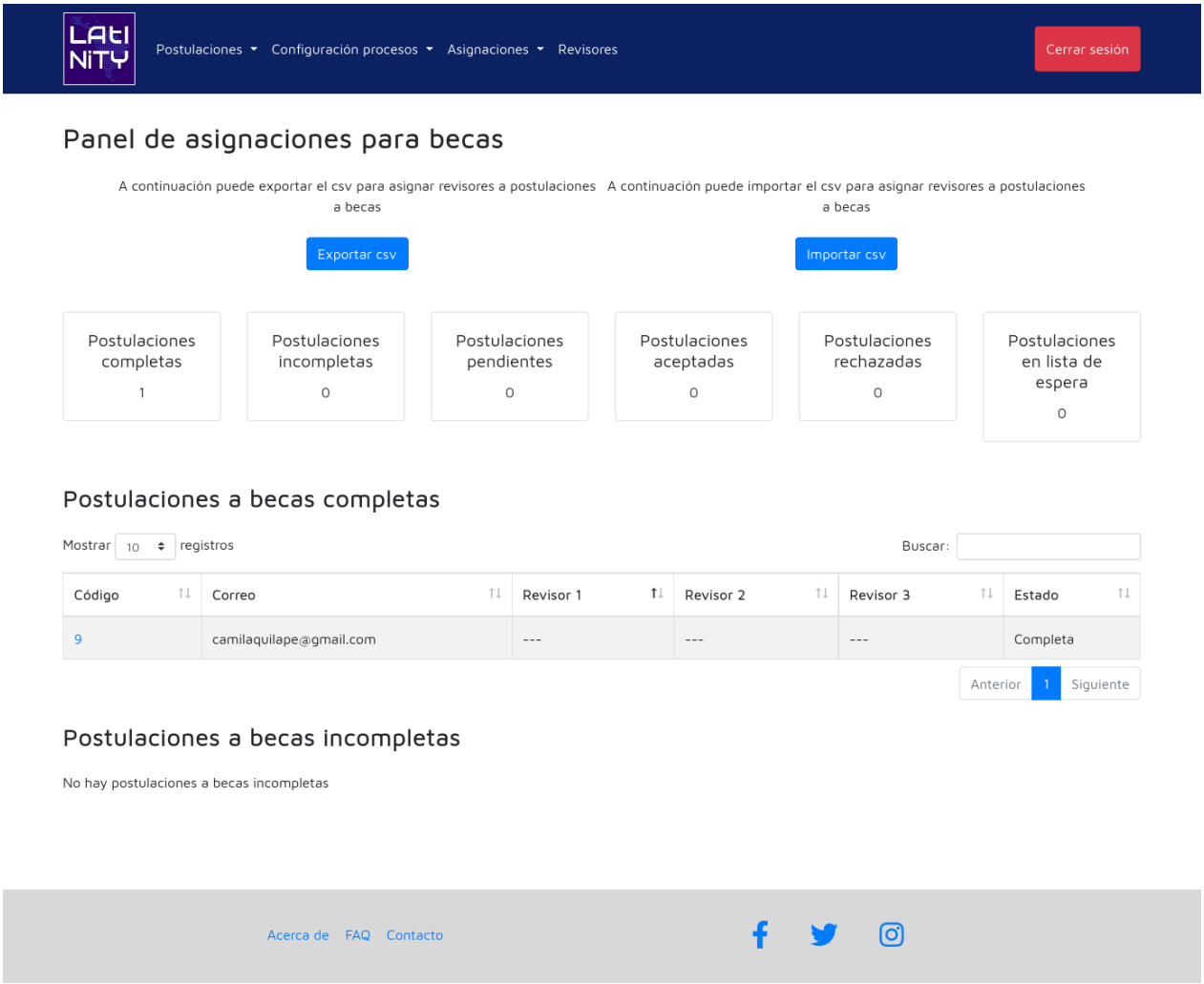

Figura 4.11: Panel de asignaciones a postulaciones a becas

El panel contiene una tabla con las postulaciones completas o con veredicto ordenadas por Estado, en la tabla se precisan los campos Código, Correo, Revisores y Estado. Aquí es relevante nombrar que la columna Revisor especifica la respuesta a la pregunta *¿Le otorgaría la beca?* y que a lo más se pueden asignar tres revisores a una determinada postulación.

El panel igualmente incluye una tabla con las postulaciones incompletas, en la que se expone Código, Correo y Tipo de postulante. Esto con el propósito de favorecer la identificación rápida de correos, en caso de que se requiera contactar a una persona en particular.

Asimismo el panel incorpora opciones de descarga y carga de archivos csv que permiten asignar revisores a postulaciones, mediante el uso de los identificadores de los revisores (ver la Figura 4.12). También presenta resúmenes de cuántas postulaciones se encuentran incompletas y completas, y de estas últimas cuántas han sido aceptadas, rechazadas, están pendientes o en lista de espera.

ID	Correo	Tipo de postulante	Revisor1	Revisor2	Revisor3
2	pablo.correa@ing.uchile.cl	Estudiante de pregrado	1		
5	camila.quilape@ing.uchile.cl	Estudiante de pregrado			

Figura 4.12: Ejemplo de archivo que permite asignar revisores a postulaciones a becas

Como se ve en la Figura 4.11, las tablas que no poseen información se reemplazan por un mensaje que hace alusión a ello. Esto con la finalidad de evitar que el usuario vea una tabla vacía y crea que se trata de un problema de la plataforma. Este comportamiento fue implementado en todas las tablas del sistema, cambiando el texto de acuerdo a la función de la tabla.

Si se selecciona una postulación completa, el sitio se redirige a la página de detalle de la revisión, donde el organizador puede ver los pormenores de la postulación, los datos de los revisores asignados y lo que han respondido a cada una de las preguntas del formulario de revisión (ver la Figura 4.13). En cambio, si la postulación está incompleta, hacer click sobre esta lleva al detalle de lo respondido por el postulante hasta el momento.

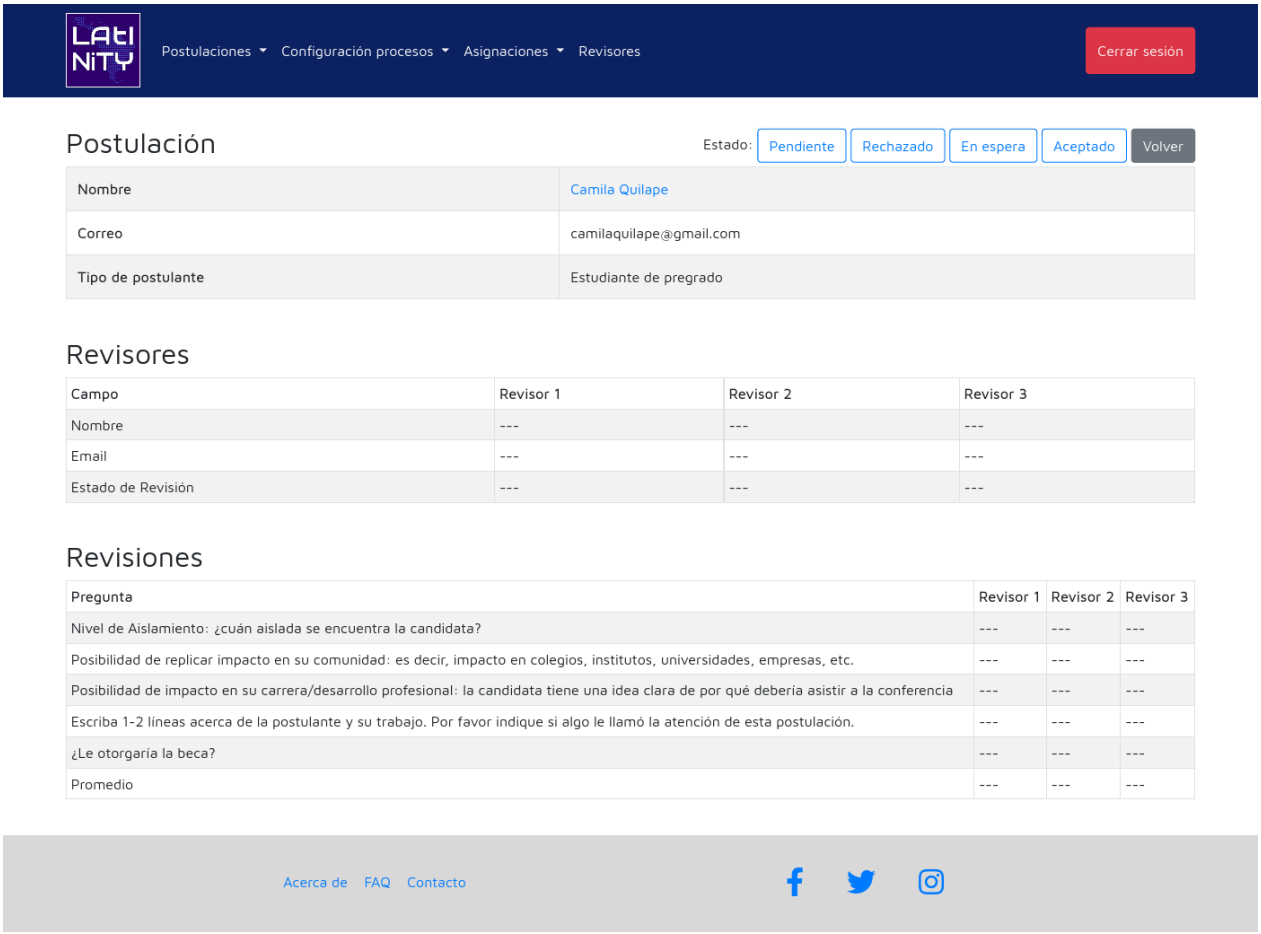

Figura 4.13: Página de veredicto a una postulación a beca en particular

Cada una de las posibles respuestas a las preguntas de revisión, excepto las de comentarios, posee un valor asociado, que va de 1 a 5 para trabajos y de 1 a 3 para becas. A partir de estos puntajes se calcula el promedio que aparece en la última fila de la tabla inferior (ver la Figura 4.13). Este promedio es considerado por el administrador o administradora, junto a la información desplegada, para tomar una decisión final.

Lo expuesto en este capítulo permite gestionar todo el proceso de revisión de becas, logrando así satisfacer el cuarto objetivo específico de *Desarrollar la revisión de becas con el propósito de completar la gestión del proceso de postulación a becas.*

4.5. Unificación de las secciones de administración

Como se mencionó antes, el sistema contaba con dos secciones de administración totalmente aisladas, la del módulo de trabajos y la del módulo de revisión de trabajos. Esto sumado a los errores y la poca optimización, hizo imprescindible un refactoring y la unificación de esas páginas con las nuevas de becas y revisión de becas

La Figura 4.14 representa a la interfaz que recibe a los usuarios de tipo administrador, en ella se encuentran resúmenes de las postulaciones y acordeones por cada tipo de postulación. Los acordeones diferencian entre postulaciones enviadas o no, y tienen la opción de descarga de un archivo csv con todas las postulaciones de esa categoría.

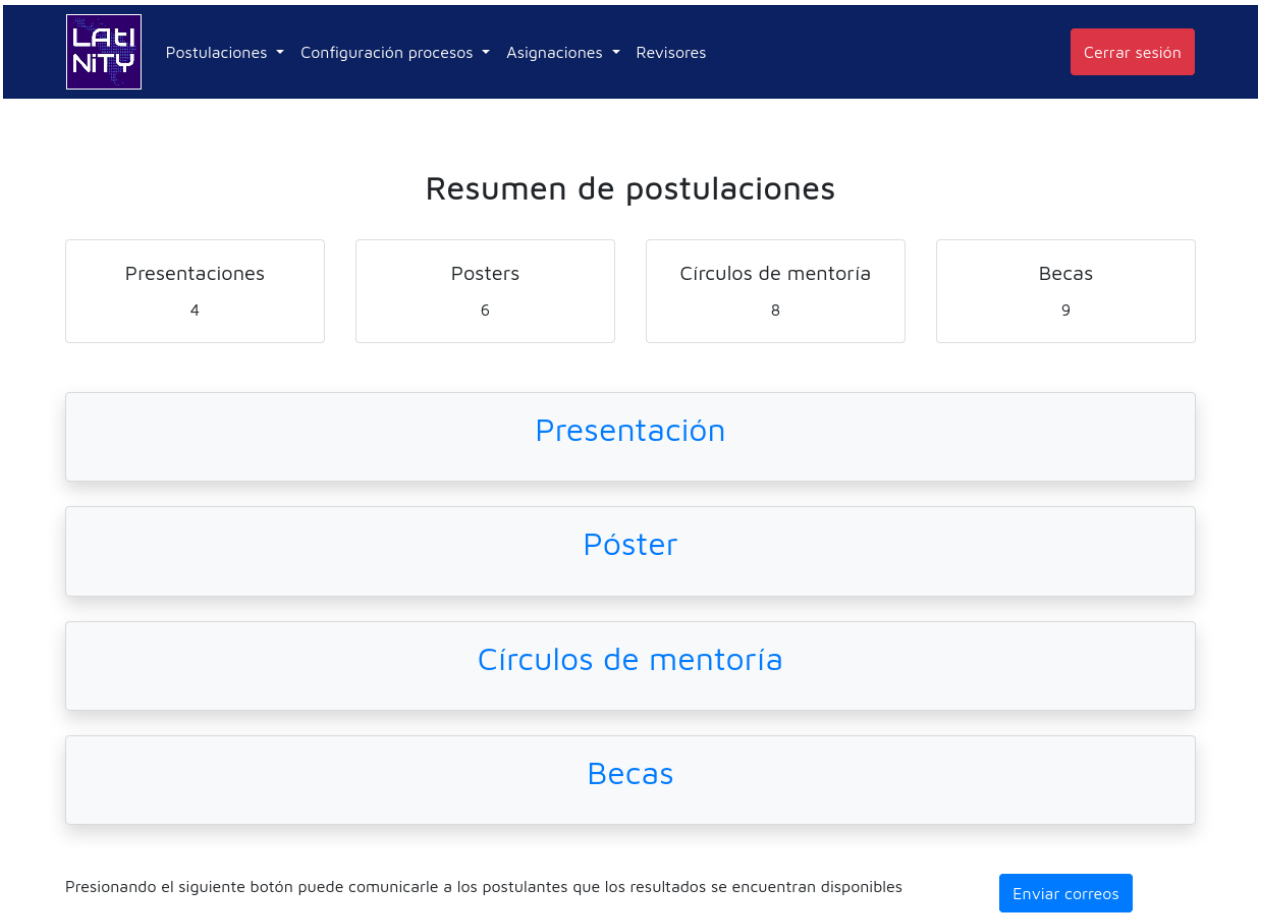

Figura 4.14: Página de inicio del administrador

En la sección de administración unificada actualmente es posible descargar archivos con las respuestas a las preguntas de postulación, tanto para becas, presentaciones, pósters y círculos de mentoría. También se pueden visualizar resúmenes con las respuestas a las principales preguntas de postulación, configurar fechas, generar asignaciones y gestionar revisores. Se accede a estas funcionalidades usando el menú superior (ver la Figura 4.14).

Entre las correcciones a lo ya implementado destaca el ajuste del archivo de asignaciones de trabajos, ahora se incluyen en el archivo solo las postulaciones que efectivamente pueden ser revisadas y no todas. Relacionado a esto, como las postulaciones no enviadas no tienen revisores, se llevaron a otra tabla de postulaciones incompletas. De modo que, esta página quedó similar al panel de asignaciones a postulaciones a becas, descrito en la sección anterior.

La diferencia fundamental es que en la tabla de las postulaciones completas a trabajos, las columnas Revisor reflejan las respuestas a la pregunta *Con respecto a la afirmación "Yo recomiendo este trabajo para Latinity", usted se encuentra en:*, con la palabra trabajo reemplazada por póster, presentación o círculo de mentoría.

Por otra parte, se corrigió el archivo de descarga de postulaciones a trabajos, el que muestra las respuestas a cada pregunta respondida por los postulantes, para que incorporara todas las postulaciones independiente de sus estados. Es importante señalar que existen tres opciones de descarga, una por cada tipo de trabajo.

El archivo descrito se puede bajar desde dos partes del sitio. De la página de inicio de la sección de administración o desde la sección de resumen por tipo de postulación, a la que se accede presionando la pestaña de Postulaciones del menú superior. Las páginas de resumen son similares a la presentada en la Figura 4.7 para becas.

Asimismo, se configuró la zona horaria del sitio para que fuera la correspondiente a la chilena y se adecuó la funcionalidad de configuración del proceso de postulación a trabajos, logrando solucionar los errores que hacían que el cambio de fechas no fuera exacto.

Por otro lado, se ajustó el código Ajax utilizado en la toma de decisiones finales, el que provocaba que no hubiera un cambio visual al seleccionar por primera vez un veredicto. Luego de estudiar la implementación, se notó que faltaba considerar el caso que se ve en el Listado 4.8, cuando una postulación aún no tiene una decisión final. El valor 2 significa que la postulación solamente ha sido enviada.

```
if (data.actual_state == 2) {
    let next_state = document.getElementById("state" + state);
    next_state.className = 'btn btn-primary';
    next_state.style.fontWeight = "bold";
}
```

Listado 4.8: Caso faltante al Ajax de veredictos

Con respecto a la seguridad, para visualizar cualquier página de esta sección de administración, es necesario estar autenticado con una cuenta de super usuario de Django. Estas cuentas se crean en el admin de Django o usando una terminal.

La unificación de las secciones de administración sumado a las correcciones y mejoras abordan parte del quinto objetivo específico.

4.6. Gestión de revisores

La siguiente etapa de implementación, en orden cronológico, fue la de gestión de revisores. Se comenzó moviendo la tabla con los revisores del sistema, desde el panel de asignaciones a postulaciones a trabajos a una nueva página, enfocada en revisores. Del mismo modo, se agregó el acceso a esta página al menú, como la pestaña Revisores.

La página de revisores desarrollada posee opciones para descargar y cargar revisores, mediante archivos csv (ver la Figura 4.15). Cuando se presiona `Exportar csv`, se origina un archivo con las columnas Código, Estado, Nombre, Apellido, Correo, Experiencia, País de origen, Track e Idioma. Este archivo, aparte de exhibir los datos de los revisores, sirve para importar nuevos, añadiendo filas con todos los campos, menos Código y Estado.

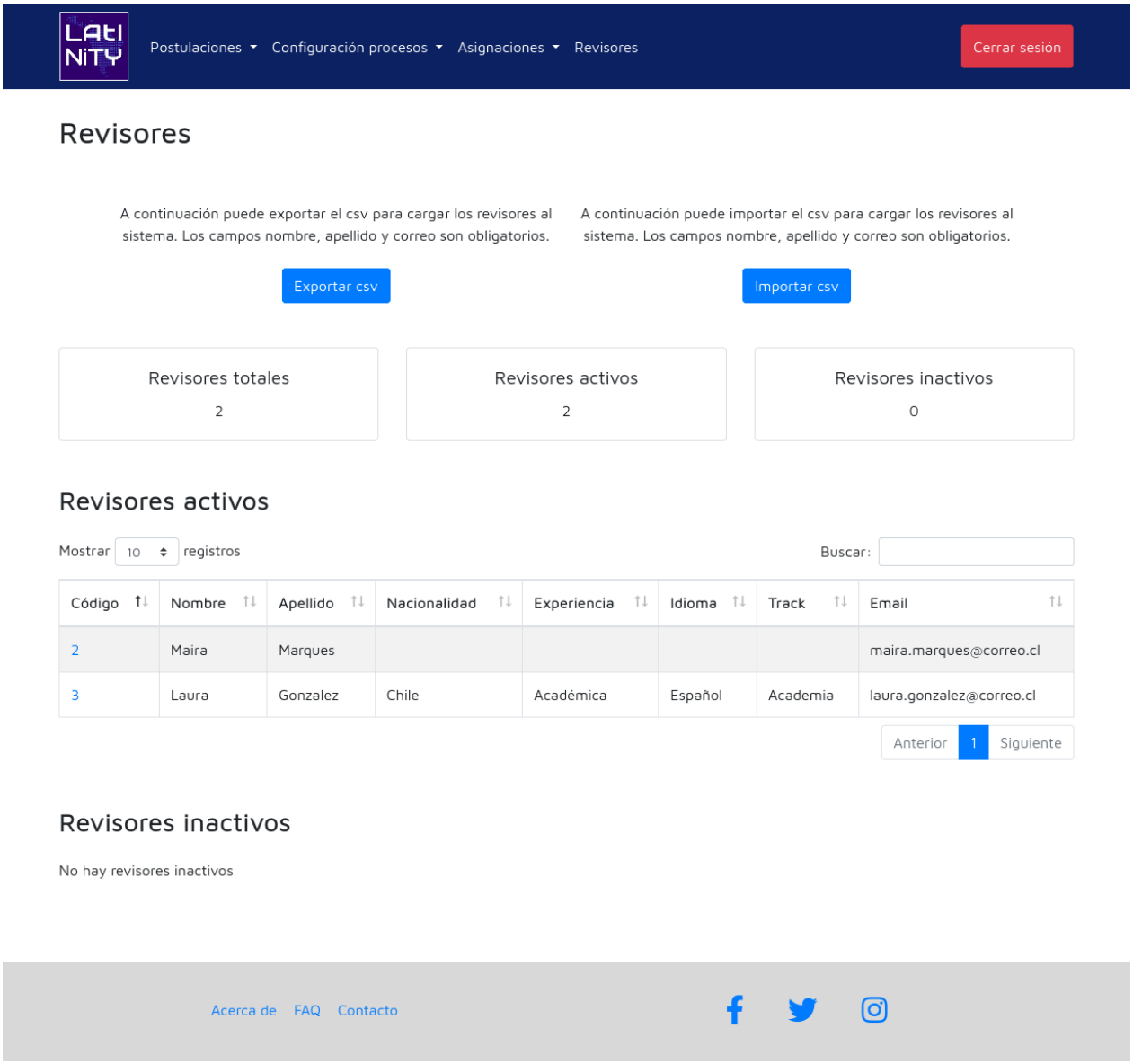

Figura 4.15: Panel de revisores

Las mismas columnas anteriores, excepto por Estado, se presentan en las tablas de revisores activos e inactivos. En dichas tablas se expone información proveniente del modelo `Reviewer`.

Es importante mencionar que los revisores inactivos corresponden al reemplazo de los eliminados. Esta decisión se tomó considerando que, eliminar revisores de la base de datos podía provocar errores al realizar ciertas consultas, como el intentar recuperar revisiones antiguas del sistema. Para hacer efectivo este cambio, se incorporó al modelo un campo booleano, que define si el revisor está activo o no.

En cuanto a desafíos, fue complejo desarrollar la funcionalidad de importar revisores, dadas las precauciones que se debían tomar en cuenta. Actualmente, la vista encargada `PanelReviewersAdmin` lee el archivo subido y comprueba que estén al menos los datos obligatorios, es decir, nombre, apellido y correo. Igualmente revisa que la información sea del

tipo y formato correcto.

La validación se hace por fila, de modo que si está todo bien, se genera de inmediato el objeto revisor. También se crea y se le asocia un token de acceso, que es enviado por correo electrónico automáticamente. En caso contrario, se guarda la fila y tipo de error en una lista, para ser mostrados al administrador, y se sigue con la fila siguiente. La validación solo se realiza para los revisores nuevos, los ya existentes se ignoran.

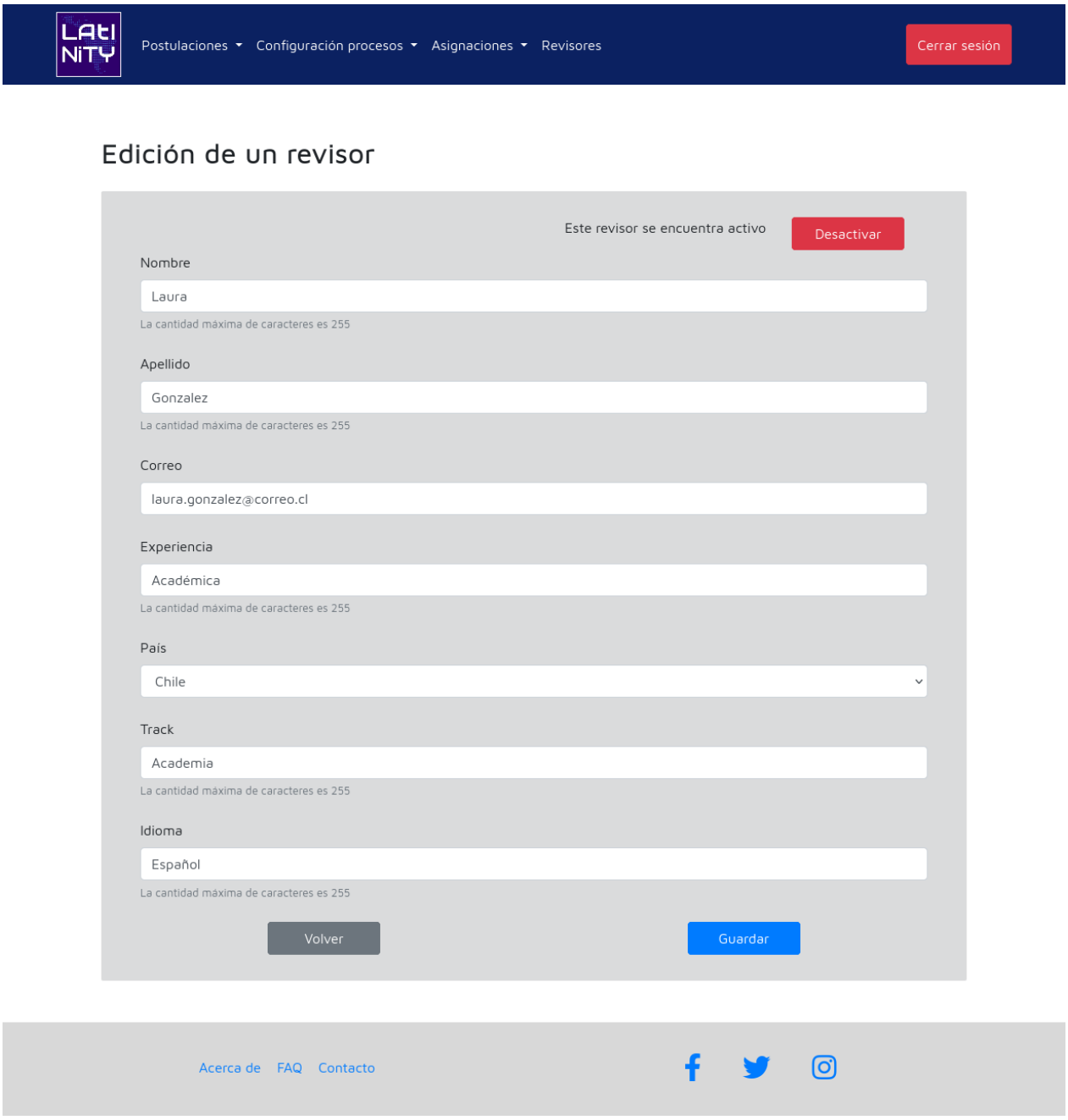

Figura 4.16: Página de edición de un revisor

Al hacer click sobre el identificador de un revisor, se redirige al usuario a una página de detalle (ver la Figura 4.16). En ella se presentan los datos del revisor en un formulario, lo que permite, además de visualizarlos, editarlos. Asimismo se añadió la posibilidad de activar o desactivar al revisor, según sea el caso.

4.7. Resúmenes e implementaciones varias

En relación a los resúmenes, se agregaron como tarjetas a la sección de administración, usando `cards` de Bootstrap. En concreto, en la página de inicio, se incluyeron con el conteo de las postulaciones a becas, a presentaciones, a pósters y a círculos de mentoría hasta el momento.

En el resumen de postulaciones a becas, las tarjetas reflejan el número de solicitudes según tipo de apoyo, a becas totales, de entrada, hospedaje o transporte. Mientras que en las páginas de asignaciones a becas y a trabajos contienen la cantidad de postulaciones incompletas y completas, desglosando estas últimas de acuerdo a si están pendientes, aceptadas, rechazadas o en lista de espera.

En cuanto a notificaciones, el sistema actualmente tiene las siguientes, las que se llevan a cabo por correo electrónico. Las primeras cuatro tenían implementaciones iniciales, que fueron completadas y/o corregidas en términos visuales y de contenido, en tanto las demás fueron desarrolladas desde cero. Se expone un ejemplo de correo en la Figura 4.17.

1. Correo de registro con el token asignado al postulante
2. Correo con un nuevo token si se solicitó recuperación
3. Correo de contacto
4. Correo de inclusión en la postulación de un grupo
5. Correo de postulación enviada
6. Correo con token para un revisor
7. Correo mencionando que los resultados ya están disponibles

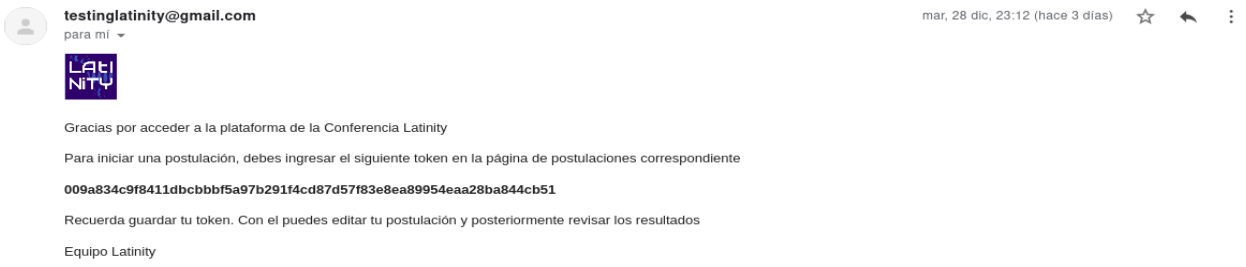

Figura 4.17: Ejemplo de correo de registro en el sistema

Referente a implementaciones varias, el sistema venía con un formulario de contacto al equipo Latinity que no funcionaba. En virtud de lo cual, se adecuó estéticamente y a nivel funcional, para que llegara efectivamente el correo con el problema ingresado, como se aprecia en la Figura A.1 del Apéndice A. Por otro lado, se ajustó el idioma del software, con el propósito de que los errores provocados por los usuarios se mostraran en español.

Un punto a destacar es que, una vez ingresados los datos personales de los integrantes en una primera postulación a trabajos, se optimizó el formulario para que en las postulaciones que siguen venga precargada la información del editor, agilizando de esta forma el paso dos de las demás postulaciones. Esta información corresponde a la de la persona que está realizando la postulación, la que se completa en el primer subformulario del paso 2 (ver la Figura A.2).

Todo este conjunto de funcionalidades complementa lo desarrollado por los equipos anteriores, logrando así que la plataforma gestione de manera adecuada parte de los procesos que comprende la conferencia. En consecuencia se cumple con el quinto objetivo específico: *Implementar funciones de gestión, tales como la unión de las secciones de administración, la carga de revisores mediante archivo y la visualización de resúmenes de apoyo. Esto es esencial para el correcto funcionamiento de los procesos que involucra la conferencia.*

4.8. Uniformización

Sobre la uniformización de interfaces, ésta se fue efectuando a medida que se trabajaba en las distintas funciones del proyecto. Por lo que no hubo un periodo de tiempo dedicado solamente a esto.

La uniformización abarcó el reordenamiento de los elementos dentro de las páginas y ajustes visuales. Por ejemplo, se cambiaron las coordenadas fijas que ubicaban los botones por el uso de grillas de Bootstrap, las que se adecuan automáticamente a los distintos tamaños de ventana. Asimismo se fijaron los footer al final de las páginas.

También se agregaron especificaciones de los límites máximos de caracteres en los campos de los formularios y alertas de Bootstrap en todo el sitio, para destacar mensajes de error, precaución o éxito.

En el Apéndice A se exhiben capturas de cómo quedó el formulario de postulación a trabajos (ver Figuras A.3 a A.4). Aquí es preciso señalar que los datos solicitados en el paso 3 del formulario dependen del tipo de trabajo al que se está postulando (ver la Figura A.4), por este motivo una vez respondida la pregunta del paso 1, no se puede editar (ver la Figura A.5).

Todas estas mejoras hicieron posible que el sitio fuera más usable. Relacionado a ello, se considera que se satisface el tercer objetivo específico: *Uniformar las interfaces, que eran diferentes en cada módulo e incluso dentro del mismo.*

4.9. Fixtures

Esta sección se dedica a comentar el manejo de la información del sitio, para lo cual se utilizaron archivos especiales llamados fixtures. Una fixture es un archivo con extensión JSON, XML o Yaml, que almacena una colección de objetos de un modelo determinado. Las fixtures permiten poblar directamente la base de datos, sin pasar por el admin de Django y asegurando consistencia. Además garantizan que se esté trabajando con los mismos datos en diferentes ambientes.

El proceso para generar estos archivos se describe a continuación. Se comienza verificando que el ambiente de donde se van a obtener los datos se encuentre limpio, en este caso el local. Luego, se van creando uno a uno los objetos en el admin de Django. Posteriormente se chequea que todo funcione bien en la plataforma y si es así se realiza la copia de la base de datos.

En el proyecto se originaron fixtures de tipo JSON, que se ubicaron en el módulo al que pertenece el modelo asociado. Por ejemplo, la fixture del modelo `Application` de becas se guardó en la carpeta del módulo de becas.

Para usar las fixtures en un ambiente dado, es necesario correr un comando por cada modelo especificado. Por lo que, se decidió optimizar esto agrupando las instrucciones en un solo archivo `load_base_data.sh`. En consecuencia, basta con correr `bash load_base_data.sh` en la terminal para cargar la información en la base de datos.

Una fixture tiene la siguiente estructura. Cada uno de sus objetos indica el modelo del que forma parte, un identificador y un campo `fields`, que engloba todos los atributos del modelo. Se aprecia lo mencionado en el Listado 4.9, para el proceso de postulación a becas.

```
{
    "model": "scholarship_admin.process",
    "pk": 1,
    "fields": {
        "name": "Becas Latinity 2021",
        "start_date": "2021-09-20T00:00:00Z",
        "end_date": "2021-10-01T00:00:00Z",
        "state": 0
    }
}
```

Listado 4.9: Ejemplo de fixture para el proceso de postulación a becas

4.10. Despliegue

Un tema importante en el que ahondar es en el de despliegue, debido a las dificultades que surgieron en el proceso. Se inició esta etapa escogiendo el proveedor a usar, Amazon Web Services (AWS). AWS [13] es una plataforma de Amazon enfocada en computación en la nube, que ofrece una colección de servicios, tales como recursos para cómputo, almacenamiento, bases de datos, análisis, redes y dispositivos móviles. Asimismo brinda herramientas para desarrolladores y de administración, y soluciones para IoT, seguridad y aplicaciones

empresariales.

En particular, se utilizó el servicio web Amazon Elastic Compute Cloud (Amazon EC2). EC2 [14] permite a las personas u organizaciones pagar por máquinas virtuales, en las que pueden ejecutar sus propias aplicaciones. Amazon EC2 proporciona capacidad informática de manera segura, ajustándose a las necesidades, de modo que el precio depende de los requerimientos y del tiempo de uso.

En Amazon EC2 se puede elegir el sistema operativo de la máquina y gestionar los permisos de acceso a la red. En este caso, el sistema operativo seleccionado fue Amazon Linux [15], una distribución de Linux provista por Amazon, similar a Fedora y CentOS. Esto dado que provee un entorno de ejecución estable y de alto desempeño. Además las instancias que corren Amazon Linux se encuentran bajo actualizaciones continuas de seguridad y mantenimiento, por parte de AWS.

Referente al servidor web, se optó por Nginx [16], debido a que es de código abierto y tiene un mejor rendimiento comparado a otros servidores web. Sobre todo sirviendo archivos estáticos, en cuanto al número de usuarios concurrentes y al consumo de recursos.

Nginx no puede comunicarse directamente con el sistema Django, por lo que necesita de una interfaz que ejecute la aplicación Latinity, que alimente las solicitudes de los clientes web y devuelva las respuestas. Quién hace este trabajo es Web Server Gateway Interface (WSGI) [17], es una interfaz que facilita la comunicación entre servidores web y aplicaciones web escritas en Python.

En específico se usó la implementación uWSGI de WSGI, configurándola para que creara un socket Unix y sirviera a Nginx mediante el protocolo uwsgi. El socket establece la conexión entre Nginx y uWSGI. Todas estas especificaciones de configuración se escribieron en un archivo de ejecución.

uWSGI se corrió en modo emperador, en este modo vigila un directorio de archivos de configuración, generando instancias por cada uno. El emperador puede crear, recargar o detener instancias de uWSGI. Así pues, si se modifica algún archivo, el emperador vuelve a cargar la aplicación automáticamente.

En la Figura 4.18 que sigue se presenta cómo se vinculan las componentes señaladas. Luego de instalarlas y establecer sus conexiones, se continuó con el código del proyecto.

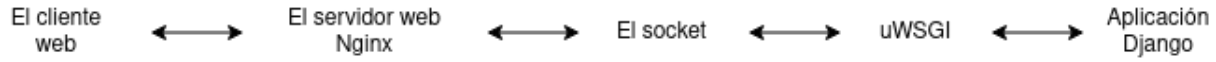

Figura 4.18: Relación entre las componentes usadas para el despliegue

Primero, se creó una nueva rama git `deploy` desde la de desarrollo `memoria` con el fin de tener una rama dedicada exclusivamente al despliegue, a la que se le incorporaron los ficheros de configuración. Luego, se clonó el proyecto en la máquina Amazon, se originaron y plasmaron las migraciones, y se cargaron las fixtures.

Entonces a medida que se iban implementando nuevas funcionalidades, el procedimiento para mantener actualizado el sitio era el siguiente. Se partía deteniendo Nginx y uWSGI. Posteriormente se llevaban los cambios de la rama `memoria` a `deploy` y se revisaban posibles conflictos. Finalmente, si se veía bien el *merge*, se volvía a levantar el sistema.

Antes de seguir con los problemas que se tuvieron durante el despligue, es relevante mencionar que la memorista no conocía ni manejaba ninguna de las herramientas descritas, por lo que se dedicaron varios días al estudio de los conceptos. El despliegue se realizó a mitad de semestre y en total tomó alrededor de una semana y media.

De partida fue un inconveniente la distribución de Linux escogida, ya que la mayoría de los tutoriales disponibles eran para Ubuntu. También fue un problema la versión de base de datos SQLite, cuyo módulo de soporte se incluye en Python, debido a que Django necesitaba una versión más reciente, ocasionando que no corrieran las migraciones.

Nginx posee el archivo de configuración `nginx.conf`, donde se establecen las ubicaciones de los archivos de registro de acceso y de errores, y se especifican tanto formatos como puertos, entre otras características. Por cada aplicación que se quiera correr en Nginx, es pertinente crear un fichero que extienda a `nginx.conf`, en este caso `latinity.conf`, como se muestra en el Listado 4.10. En `latinity.conf` se escribió la dirección del sitio, y las rutas del socket, de la carpeta de archivos estáticos y de la carpeta de archivos multimedia.

```
server {
    listen 80;
    server_name 3.15.248.200;
    access_log /var/log/nginx/access.log;
    error_log /var/log/nginx/error.log;

    location /static/ {
        alias /home/ec2-user/Latinity/latinity2020/code/assest/;
    }

    location /media/ {
        alias /home/ec2-user/Latinity/latinity2020/code/media/;
    }

    location / {
        uwsgi_pass unix:///home/ec2-user/Latinity/latinity2020/code/
   latinity.sock;
        include uwsgi_params;
    }

    error_page  404              /404.html;
    location = /40x.html {
        root   /usr/share/nginx/html;
    }

    error_page   500 502 503 504  /50x.html;
    location = /50x.html {
        root   /usr/share/nginx/html;
    }
}
```

Listado 4.10: Archivo `latinity.conf`

El archivo `latinity.conf` le provee a Nginx la información precisa para que proporcione los ficheros multimedia y estáticos directamente del sistema de archivos. Separando de este modo, esas solicitudes de las que requieren la intervención de Django. Entre las dificultades que se tuvieron con los archivos de configuración se encuentran dónde ubicarlos, cómo hacerlos visibles y qué escribir exactamente en ellos.

Se tuvieron muchos problemas con los permisos, propietarios, grupos y usuarios de los archivos, los que provocaban errores que no parecía que estuvieran relacionados. Por ejemplo, por este motivo no funcionaban los botones de agregar/quitar postulante, en el paso de integrantes de la postulación a trabajos (ver la Figura A.2). Otro error, en esta misma página, fue que no se visualizaban los botones de la parte inferior, lo cual se solucionó modificando el usuario del archivo `nginx.conf`.

Uno de los últimos inconvenientes notados fue que los archivos de registro, tanto el de acceso como el de errores, dejaban de funcionar de un día a otro, vale decir Nginx dejaba de escribir en ellos. Lo que ocurría en realidad era que `nginx.conf` tenía una línea errónea. Esto ocasionaba que los archivos, que se crean diariamente, se generaran con un propietario que hacía que Nginx se quedara sin permisos de escritura.

Las secciones exhibidas en este capítulo junto a las imágenes del Apéndice A reflejan el trabajo realizado durante la etapa de desarrollo. En este capítulo se detallaron funcionalidades interesantes, funciones que fueron difíciles de lograr y los problemas enfrentados. También se expuso la conexión entre las tareas y los objetivos, y se describió el proceso de despliegue.

Capítulo 5

Validación

Con el propósito de verificar que el sistema desarrollado resuelve el problema planteado y satisface el objetivo general, se validó el software de dos maneras. La primera se fue realizando durante el transcurso de la memoria, a través de reuniones semanales con la profesora guía. En dichas reuniones se revisaban los avances, se resolvían dudas y se planificaban los pasos siguientes.

La segunda manera se efectuó una vez finalizada la implementación, llevando a cabo pruebas de usuario, entrevistas y encuestas de usabilidad a los distintos tipos de actores. Este capítulo se enfoca en esta validación, especificando la metodología seguida, los resultados obtenidos y el análisis.

5.1. Encuesta de usabilidad

Lo primero a mencionar es la herramienta que fue utilizada para recopilar información cuantitativa, la encuesta System Usability Scale (SUS) [5] que mide la usabilidad de un determinado software. SUS se escogió por su fácil administración, rapidez y validez, es confiable a la hora de clasificar un sistema como usable [18].

SUS consiste en un cuestionario de diez declaraciones, para las cuales el encuestado debe indicar, en una escala de uno a cinco, qué tan de acuerdo está. Con uno asociado a muy en desacuerdo y 5 a muy de acuerdo. Las otras alternativas son en desacuerdo, ni de acuerdo ni en desacuerdo y de acuerdo.

La mitad de las declaraciones se relacionan a aspectos positivos de la plataforma, mientras que la otra mitad a aspectos negativos. La siguiente lista presenta las aseveraciones que contempla la encuesta.

1. Creo que me gustaría utilizar este sistema
2. Encontré el sistema innecesariamente complejo
3. Creo que el sistema era fácil de usar

4. Creo que necesitaría apoyo para poder utilizar este sistema
5. Encontré que las diversas funciones del sistema estaban bien integradas
6. Creo que había demasiada inconsistencia en el sistema
7. Creo que la mayoría de las personas aprendería a utilizar este sistema muy rápidamente
8. Encontré el sistema muy complicado de usar
9. Me sentí muy seguro/a usando el sistema
10. Necesitaba aprender muchas cosas antes de poder utilizar este sistema

El procedimiento para calcular los resultados es el siguiente. A cada una de las respuestas a las declaraciones positivas, las impares 1, 3, 5, 7 y 9, se les resta uno. En tanto, para las declaraciones negativas, las pares 2, 4, 6, 8 y 10, a 5 se le resta el valor de la respuesta.

Después, se suman estos valores y se multiplica el resultado por 2,5. Este valor corresponde al puntaje final del sistema, varia entre 0 y 100. Los sitios con más de 68 puntos en esta escala son considerados como usables, pero pueden mejorar, y sobre 80 totalmente usables.

5.2. Procedimiento

La validación se orientó a revisar la usabilidad del software, su funcionamiento y la experiencia de usuario. Con este fin, se decidió separar la evaluación de la plataforma en tres partes: la sección de postulaciones, la sección de revisiones y la sección de administración.

Para validar la sección de postulaciones se reclutaron a cinco personas, las cuales tuvieron que simular una postulación a beca y dos a trabajos. Se comenzó esta etapa comunicando y habilitando un formulario de inscripción, en el que se hacía un llamado a participar. La publicación fue difundida en la comunidad Alumnos DCC de u-cursos y en grupos de Latinity.

Como resultado de lo anterior, las personas que colaboraron en las entrevistas tenían diferentes perfiles, desde estudiantes hasta profesionales no ligados a tecnología. También cooperaron personas que ya habían asistido a Latinity y otras que tenían intenciones de concurrir.

Los participantes fueron contactados por correo electrónico. Se les enviaba un correo en el que se explicaba el mecanismo de la evaluación, se pedía disponibilidad horaria y se adjuntaban dos documentos. Uno con una breve descripción de cada tipo de postulación y otro con ejemplos de trabajos. El o la participante podía acudir a estos documentos si no sabía qué responder a un campo de los formularios.

Dada la pandemia, las pruebas se realizaron de forma online, en una sesión grabada por Zoom, donde se partía pidiendo al entrevistado que compartiera su pantalla. Aparte de la postulación a beca, los tipos de trabajos a postular iban cambiando por participante. No se consideraron los tres tipos de trabajos debido a que las postulaciones solo se diferencian en el paso 3, por lo que hubiese sido redundante.

En las sesiones se observaba cómo los usuarios iban completando los pasos. Además, se les solicitaba ir verbalizando sus pensamientos a medida que interactuaban con el sistema. La idea era incentivar a los participantes a que fueran comentando lo que encontraban extraño, poco claro o malo, con el fin de conseguir mayor información del diseño y de la experiencia, e identificar posibles errores.

Al finalizar las simulaciones se le pedía a los entrevistados que respondieran la encuesta de usabilidad. La encuesta SUS fue disponibilizada por medio de Google Forms, una herramienta que permite crear y contestar formularios en línea. En el formulario también se incluyó una caja de comentarios, para que los encuestados pudieran ingresar observaciones.

Para revisar la sección de revisiones se estableció el contacto con una revisora de Latinity, a la que previamente se le preguntó su disponibilidad para participar en la evaluación del sitio. Dado que en este caso los formularios a contestar por el revisor son más cortos, de 3 y 6 preguntas, la validación fue más bien subjetiva.

A medida que se le presentaba esta sección, ella iba comentando sus apreciaciones, de acuerdo a su experiencia. La revisora señaló aspectos importantes a destacar al interior de las interfaces y buscó errores. Asimismo respondió los dos formularios, el de revisión de becas y el de trabajos para postulaciones de prueba, y contestó la encuesta de usabilidad.

Para evaluar la sección de administración, se contactó a una organizadora de la conferencia a través de correo electrónico y se coordinó un horario de reunión. En dicha videollamada se efectuó una demostración completa del sistema, exponiéndole a la organizadora el funcionamiento de la plataforma para los distintos perfiles de usuario. Igualmente se fueron resolviendo dudas y se recibieron comentarios iniciales.

Esta reunión tuvo una duración de una hora y se llevó a cabo con el objetivo de dar un contexto del sitio. Posteriormente, se le pidió a la organizadora que revisara por su cuenta el software, enfocándose en la sección de administración, con el propósito de buscar errores y oportunidades de mejora. Los resultados de lo anterior se comentaron en una segunda videollamada.

Cabe mencionar que todas estas pruebas se ejecutaron con el sistema corriendo en la máquina de Amazon. Al comenzar la sesión de Zoom se le enviaba un mensaje al entrevistado, en el que iba el link de la dirección del sitio.

5.3. Resultados cuantitativos

A continuación se detallan los puntajes obtenidos al tomar la encuesta SUS a los participantes de la sección de postulaciones (ver la Tabla 5.1), y a la revisora (ver la Tabla 5.2). Los resultados están desglosados por los valores asociados a las respuestas a cada declaración, de D1 a D10. Las últimas columnas corresponden al puntaje SUS.

	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	Puntaje
Participante 1	5	1	5	1	5	1	5	1	5	1	100
Participante 2	5	1	4	1	5	1	5	1	5	1	97.5
Participante 3	5	2	5	2	5	2	5	2	5	2	87.5
Participante 4	5	1	4	1	5	1	5	1	5	1	97.5
Participante 5	5	2	4	1	5	1	5	1	4	3	87.5

Tabla 5.1: Valores seleccionados por los postulantes y puntajes SUS

	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	Puntaje
Participante 1	5	1	5	1	5	1	5	1	5	1	100

Tabla 5.2: Valores seleccionados por la revisora y puntaje SUS

Los puntajes SUS conseguidos se consideran bastante buenos, ya que son valores sobre los 80 puntos y cercanos o iguales al máximo posible. También se nota que todas las respuestas a las declaraciones positivas varían entre 4 y 5, números vinculados a *de acuerdo* y *muy de acuerdo*, y que la mayoría de las respuestas a las negativas son 1 o 2, es decir, *muy en desacuerdo* o *en desacuerdo*. Estos resultados reflejan una buena percepción del sistema, por parte de los usuarios.

La excepción a lo descrito corresponde a la declaración 10: "Necesitaba aprender muchas cosas antes de poder utilizar este sistema". Aquí uno de los encuestados respondió 3, lo que significa *ni de acuerdo ni en desacuerdo*. Esto puede deberse a que se trata de una plataforma orientada a una conferencia científica, por lo que algunos términos son muy específicos a ella. Por consiguiente, se podría mejorar lo anterior incorporando más definiciones o instrucciones a las preguntas, por ejemplo con tooltips.

Asimismo destaca la valoración a las aseveraciones 1: "Creo que me gustaría utilizar este sistema", 5: "Encontré que las diversas funciones del sistema estaban bien integradas" y 7: "Creo que la mayoría de las personas aprendería a utilizar este sistema muy rápidamente", debido a que todos los entrevistados respondieron *muy de acuerdo*. Se aprecia que las tres declaraciones se relacionan directamente con los motivos que hicieron indispensable construir un sistema para Latinity.

5.4. Resultados cualitativos

Respecto a la retroalimentación recopilada durante las pruebas de usuario y entrevistas, se obtuvieron una cantidad considerable de comentarios, los que se especifican en el Apéndice B. En esta sección solo se van a mostrar y analizar los más relevantes e interesantes.

5.4.1. Postulaciones

Lo primero a mencionar es que todos los participantes de la sección de postulaciones lograron ingresar postulaciones con éxito. Relacionado a esto, igualmente se advirtió que los usuarios comprendieron rápidamente el flujo de los formularios.

Uno de los comentarios que más se repitió fue que los links a la edición de postulaciones eran inconsistentes entre becas y trabajos. Estos links se encuentran en las tablas de las páginas de inicio, en becas el enlace está en el identificador, mientras que en trabajos en el nombre del trabajo. Se piensa corregir esto haciendo toda la fila clickeable o dejando el link vinculado a un botón editar.

El otro comentario más frecuente fue que en el formulario de becas falta decir que el vídeo de presentación pedido se trata de un enlace. En general, se notó el bajo número de instrucciones y la necesidad de campos más descriptivos en cuanto a lo que se espera. Esto toma aún más importancia para usuarios no familiarizados con la conferencia, por lo que son aspectos que deben ser mejorados.

Referente a los botones, es pertinente señalar qué significa el botón `Guardar` en los formularios, ya que se confunde un poco con el botón `Paso siguiente`. A los entrevistados les preocupaba presionar `Paso siguiente` y que se perdiera la información ingresada. En realidad los botones `Guardar` y `Paso siguiente` funcionan parecido, con la diferencia de que luego de almacenar los datos, el primero recarga la página y el segundo lleva al paso que sigue.

Asimismo, en el paso 2 de trabajos, es necesario añadir el texto de que se pueden agregar hasta ocho postulantes y mencionar que para eliminar un postulante se debe presionar `Eliminar` y luego `Guardar`, si no, no es efectivo el cambio.

En este mismo paso, también falta especificar que el primer formulario, de los ocho posibles, se debe completar con los datos de quién está realizando la postulación, vale decir, los de la persona que se registró en el sistema. Lo anterior se debe a que es la información de ese formulario la que viene precargada, si se postula a otro trabajo.

Por otra parte, hubieron varios comentarios enfocados a aspectos estéticos, entre ellos: "faltan los tildes en las opciones de países", "podrías dejar los links que están en los resúmenes de becas y trabajos como clickeables", "revisa los colores y ubicación de los botones, pues algunos cambian de posición de un paso a otro" y "podrías eliminar el botón eliminar postulante cuando haya solo uno". Estos comentarios implican cambios pequeños, pero que le darían un mayor valor a la solución, de modo que se van a considerar todos.

Los puntos más graves hallados durante las pruebas fueron que el sitio se cae al intentar registrarse con correos del estilo `aaaa@aaa` y que se puede acceder a otras postulaciones a través de la url, cambiando el identificador. Estos problemas se van a abordar lo antes posible, incorporando el caso que falta en el primero y añadiendo el token a los filtros que recuperan las postulaciones para el segundo.

Por otro lado, los participantes sugirieron los siguientes cambios. En trabajos, mover la pregunta nombre del trabajo al paso 1, con la finalidad de que se creen objetos postulaciones solo cuando se complete el nombre y tipo de trabajo, y así no tener postulaciones sin nombre hasta llegar al paso 3.

En el paso 2 del mismo módulo, donde se ingresan los integrantes de un grupo de trabajo, implementar el eliminar un miembro en particular y no solo el último, como está ahora. También se podría agregar un modal de confirmación, dado que es una operación delicada.

En el paso 3 de becas y trabajos, reemplazar el botón `Enviar` por uno que diga `Paso siguiente` o `Ir a resumen`, debido a que no se envía realmente la postulación ahí, si no en la página de resumen, al presionar `Confirmar envío`.

Relacionado a lo previo, una vez enviada la postulación quitar el botón `Confirmar envío`, para que cuando se edite el formulario no lleguen correos innecesarios. Tal vez igualmente sacar los botones `Guardar` en este mismo caso.

Todas estas posibles modificaciones se van a analizar, para decidir si llevarlas a cabo o no. Es importante mencionar que lo más difícil sería el eliminar un integrante específico, pues conlleva repensar y rehacer por completo la vista encargada del paso 2 de postulación a trabajos, un cambio significativo.

Como oportunidad de mejora se tiene el asignar un id único a postulaciones a becas y trabajos, y diferente al de la base de datos. La clienta lo considera útil para que los postulantes puedan realizar sus consultas señalando ese identificador, y así encontrar las postulaciones rápidamente. Además permitiría ocultar la posición de una determinada postulación en la base de datos.

Otra oportunidad de mejora, que surgió de lo observado, es cambiar la lógica de generación de postulaciones a trabajos. Se podría originar el objeto postulación al guardar lo ingresado en el paso 1 y no al presionar `Nueva postulación`. Esto en vista de que, si solo se hace click en `Nueva postulación` queda una fila casi en blanco en la página de inicio, es decir, se crea un objeto sin información relevante.

Referente a los comentarios positivos, se recibieron estos: “plataforma de uso simple”, “bastante rápido postular”, “siento que el sistema ayudaría mucho a la organización de latinity” y “muy buen sistema”. Los que dejan en evidencia una buena acogida y se relacionan al propósito del software.

Excepto por los errores encontrados, se considera que se lograron buenos resultados, ya que un gran porcentaje de los comentarios implican cambios pequeños. Está contemplado trabajar en estos cambios para mejorar esta sección del sitio.

5.4.2. Revisiones

La evaluación de esta sección fue algo más informal, como se indicó en la sección Procedimiento, y tuvo una duración de aproximadamente media hora.

Entre los comentarios recibidos destaca que el texto CV del formulario de revisión de becas y el campo enlace a hoja de vida del formulario de postulación a becas son inconsistentes. Se debería usar el mismo término, pues CV y enlace a hoja de vida hacen alusión a lo mismo.

También se mencionó que falta incorporar la definición de aislamiento en el formulario de revisión de becas, dado que es un concepto muy importante para la evaluación adecuada de postulaciones a becas.

Otro comentario fue que sería beneficioso añadir la columna aislada a la tabla de postulaciones a becas, que se presenta en la página de inicio de la sección de revisión, por el mismo motivo del párrafo anterior. Todos estos comentarios se orientan a implementaciones pequeñas, que mejoran lo ya desarrollado, por lo que, se considera valioso llevarlas a cabo.

Los cambios más significativos mencionados por la revisora fueron el de solicitar una página personal Github o página del proyecto en los formularios de postulaciones, y el de adicionar una caja de comentarios extra en el formulario de revisión de trabajos, para otorgar feedback al postulante. Estas modificaciones van a quedar como trabajo futuro por el tiempo disponible.

Tomando en cuenta el comentario final de la revisora: "el sistema es bastante intuitivo y sencillo de usar", los descritos antes y que no se encontraron errores, se concluye que esta sección cumple con ser funcional y completa.

5.4.3. Administración

Al igual que en la sección anterior, en ésta no hubieron muchos comentarios. Algunos de ellos fueron revisar la redacción del acordeón de la página de inicio, y ajustar textos varios, como los de los títulos de las páginas y los nombres de las columnas en las tablas. Asimismo la organizadora notó la falta de instrucciones en las páginas de asignaciones, que precisen cómo se utilizan los archivos csv.

Por lo tanto, se pueden agrupar los comentarios en problemas de redacción y falta de instrucciones. Se espera que estos puntos sean tomados en cuenta e implementados en la versión final del sistema.

De lo expuesto en este capítulo, considerando los puntajes SUS y los resultados cualitativos, se puede decir que el software es usable, funcional y completo, y que además resuelve el problema planteado inicialmente. No obstante, todavía existen varios aspectos a mejorar, con el fin de conseguir un sistema más intuitivo.

El trabajo descrito en este capítulo logra cumplir con el sexto objetivo específico, el cual dice: *Validar la usabilidad y experiencia de usuario del sistema implementado, mediante pruebas a posibles postulantes, a revisores y a administradores. Recopilando de estas validaciones niveles de satisfacción y oportunidades de mejora.*

Capítulo 6

Conclusiones

En el presente informe se profundizó en el proceso seguido para resolver el problema de extender lo desarrollado. A continuación se exhibe cómo se conectan las tareas realizadas con el cumplimiento de cada uno de los objetivos específicos.

El primer objetivo específico: *Analizar el sistema, a nivel de código, con el fin de entender cómo estaba diseñado e implementado. También para determinar si reestructurar el módulo de becas modificándolo o reimplementándolo, dado que el recibido no satisfacía las necesidades de la conferencia* fue abordado en el curso Introducción al Trabajo de Título y los hallazgos se precisaron en el capítulo de Análisis y Diseño. A partir de lo anterior se tomó la decisión de reimplementar el módulo de becas, debido a los cambios significativos que se querían realizar.

Una consecuencia implícita del primer objetivo fue la de corregir errores, ya que no se podía comenzar a trabajar en nuevas funcionalidades si las que habían no funcionaban bien. Por ello, se empezó con la resolución de problemas, lo que contempló la corrección de los errores de navegación y de validación en formularios, la limpieza del código y el ajuste de las condiciones y filtros.

Luego se continuó con el segundo objetivo específico: *Reestructurar el módulo de becas, comenzando con el diseño de un nuevo modelo de datos.* Dicho modelo de datos fue diseñado el primer semestre del 2021. Esta etapa partió con la creación de nuevos submódulos y abarcó el plasmar los modelos en el código, la construcción de las vistas y el ajuste de las interfaces de usuario.

Posteriormente se abordó el cuarto objetivo específico: *Desarrollar la revisión de becas con el propósito de completar la gestión del proceso de postulación a becas.* Para cumplir con él, se determinó extender el módulo de revisión de trabajos, agregando nuevos modelos y ampliando o generando vistas. Esta parte engloba a grandes rasgos la asignación de revisiones, la revisión de postulaciones a becas y la toma de decisiones finales.

Para satisfacer el quinto objetivo específico: *Implementar funciones de gestión, tales como la unión de las secciones de administración, la carga de revisores mediante archivo y la visualización de resúmenes de apoyo. Esto es esencial para el correcto funcionamiento de los procesos que involucra la conferencia* se dividió el trabajo en varias etapas.

La primera fue la de la unión de las secciones de administración, la que comprendió la reorganización de elementos y la resolución de inconvenientes de gestión. La siguiente etapa se enfocó en la gestión de revisores, incluyó la descarga y carga de revisores a través de archivos, su edición y desactivación.

La tercera etapa fue la de añadir resúmenes de apoyo, que visibilizaran cifras relevantes para el administrador, tales como cantidad de revisores, de postulaciones completas/incompletas, de revisiones, etc. Finalmente, la última etapa se orientó a implementaciones varias, como la incorporación de notificaciones y el ajuste del idioma.

El tercer objetivo específico: *Uniformar las interfaces, que eran diferentes en cada módulo e incluso dentro del mismo* se abordó a medida que se iba avanzando en las distintas funciones, de modo que no hubo un periodo de tiempo dedicado exclusivamente a esto.

Con respecto al sexto objetivo específico: *Validar la usabilidad y experiencia de usuario del sistema implementado, mediante pruebas a posibles postulantes, a revisores y a administradores. Recopilando de estas validaciones niveles de satisfacción y oportunidades de mejora*, efectuarlo arrojó buenos resultados.

En general, la mayoría de los comentarios recopilados apuntaron a ajustes visuales y a la interpretación de las funcionalidades brindadas por el sitio. Esto se considera positivo, ya que se asocian a cambios que no son complicados de llevar a cabo.

Las excepciones a lo anterior fueron el problema del correo y el de la accesibilidad a otras postulaciones por medio de la url. Corresponden a problemas graves, por lo que necesitan ser solucionados lo antes posible.

Los altos puntajes SUS y los comentarios positivos obtenidos reflejan una buena acogida del sistema, permitiendo asegurar su usabilidad. Aún así, se reconoce que existen varias oportunidades de mejora en diferentes partes de la plataforma, que pueden aportar una mayor robustez y calidad al sitio.

A partir de lo mencionado, se medita que las validaciones fueron las correctas. Sin embargo quedó pendiente llevar a cabo pruebas de carga y estrés, con el propósito de determinar los recursos computacionales adecuados (memoria, disco, cpu, etc.) para ajustar la máquina de Amazon.

Es relevante señalar que la configuración de recursos debe tolerar un alto número de postulaciones en un periodo corto de tiempo, debido a que la mayoría de las personas ingresan postulaciones a último momento.

Dado que se abordaron todos los objetivos específicos, se concluye que se cumplió con el objetivo general de *reestructurar y completar el sistema de gestión de Latinity, con el propósito de lograr por primera vez un software usable, funcional y completo, que facilitara la administración de todas las tareas relacionadas a las postulaciones a becas y trabajos. Esto contemplaba la integración de funcionalidades presentes con otras corregidas y nuevas.*

Se considera que el software desarrollado es un gran aporte a la organización de la conferencia, ya que simplifica las tareas de gestión de las postulaciones a becas y trabajos, reduciendo sus tiempos de ejecución y las probabilidades de cometer errores. Actualmente la plataforma corresponde a la primera versión totalmente unificada.

Como se comentó en el capítulo de Implementación, el sistema se encuentra corriendo en una máquina de Amazon y se puede acceder a él a través de una dirección IP, por lo que ya puede ser utilizado desde cualquier navegador web.

Lecciones aprendidas

Referente a las lecciones adquiridas, a nivel de programación se aprendió más del desarrollo de software con el framework Django, sobre todo con respecto al manejo de archivos y de errores. Asimismo, se obtuvieron conocimientos sobre formsets de Django y su empleo para controlar el dinamismo en formularios.

Por otro lado, a pesar de los problemas que se tuvieron durante su configuración, se valora el uso de Amazon Web Services, junto al aprendizaje logrado en el despliegue, debido a que es un servicio ampliamente utilizado hoy en día.

A nivel de organización, se medita que el uso de la técnica pomodoro fue eficaz, puesto que aparte de facilitar la contabilización del tiempo de trabajo, permitió establecer descansos controlados. También se aprecian las reuniones de seguimiento semanales, pues favorecieron la retroalimentación oportuna y la comunicación entre la profesora guía y la memorista.

En cuanto a las dificultades enfrentadas, destaca la implementación de vistas cuya funcionalidad es compleja, por los casos a tener en cuenta, por ejemplo las encargadas de los formularios, y el enmendar errores. Como consecuencia del tamaño del proyecto, a veces era complicado hallar el trozo de código que fallaba y dimensionar los efectos que podía tener.

Del trabajo efectuado, se considera valioso el aprendizaje asociado a tomar tres prototipos académicos distintos y convertirlos en un producto real, con todo lo que eso conlleva y pese a lo difícil que es entender un código en el que han participado tantas personas. Se cree que fue una experiencia enriquecedora.

Relacionado a lo anterior, se reflexiona que se habría ganado tiempo si se hubieran establecido algunos lineamientos al principio, antes de comenzar a crear el software. Como seguir un determinado estilo de interfaces, construir vistas de cierta manera, y generar commits descriptivos y constantes. El sistema recibido tenía un único commit para el módulo de becas y el de trabajos, de modo que no fue de ayuda en el análisis. Estos aspectos habrían sido útiles para reducir el tiempo que fue dedicado a estudiar el sistema, a resolver problemas y a uniformizar interfaces.

Como opinión personal, se piensa que el proyecto fue provechoso para aprender sobre ingeniería de software, testing y a trabajar con ambientes reales. Por otro lado, el desarrollo ayudó a consolidar conocimientos sobre el manejo de objetos, herencia, bases de datos, entre otros.

Trabajo futuro

Como recomendación para un trabajo futuro, se sugiere seguir los siguientes pasos. Empezar con la comprensión de la estructura del proyecto, carpetas y archivos, y de los modelos de datos. Posteriormente, continuar con el estudio de lo implementado a nivel de código, ahondando en el flujo de la información y en la relación modelo - vista - template.

Los cambios vinculados a los comentarios, las oportunidades de mejora y las sugerencias vistas en el capítulo anterior quedaron pendientes, principalmente debido a las limitaciones de tiempo. Es importante tener en cuenta que algunas mejoras implican modificaciones significativas en el back-end, que pueden afectar a otras funcionalidades.

Una funcionalidad que se podría incorporar en el futuro es la opción de eliminar postulaciones a trabajos por parte del postulante, con el propósito de que al cierre de un proceso no queden postulaciones incompletas porque se dejaron "botadas". Una postulación puede quedar botada si el usuario se equivoca al seleccionar el tipo de trabajo.

Otra función interesante podría ser extender la sección de administración para añadir estadísticas, que ayuden a los organizadores a tomar mejores decisiones finales. Estas estadísticas podrían englobar diferentes tipos de gráficos.

En esta misma parte, un aspecto que le daría mayor valor al sistema es poder seleccionar qué información ver, de cuál año o proceso. Esto dado que actualmente solo se muestran los datos relacionados a los procesos más recientes de becas, trabajos y revisiones.

Bibliografía

[1] B. Castro, S. Cifuentes, K. Cossio, D. Gálvez, F. Lizama, J. Robert, and M. Sepúlveda, "Equipo Latinity Primavera 2019."

[2] M. Araneda, M. Becerra, J. Díaz, S. Donoso, I. Peña, and V. Pinto, "Equipo Latinity Otoño 2020."

[3] R. Aguilera, V. Espina, D. Ortego, C. Quilape, F. Sanhueza, J. Toloza, and S. Vega, "Equipo Latinity Primavera 2020."

[4] "Técnica Pomodoro," https://blog.trello.com/es/tecnica-pomodoro, recuperado el 02 de marzo de 2022.

[5] "System Usability Scale," https://en.wikipedia.org/wiki/System_usability_scale, recuperado el 01 de mayo de 2021.

[6] "EasyChair," https://easychair.org/conference, recuperado el 20 de julio de 2021.

[7] "HotCRP," https://hotcrp.com/, recuperado el 20 de julio de 2021.

[8] "PaperCall," https://www.papercall.io/, recuperado el 20 de julio de 2021.

[9] "Software de Anita Borg para Grace Hopper Celebration," fuente profesora Jocelyn Simmonds.

[10] J. Simmonds, "Repositorio Latinity 2020," https://github.com/jsimmond/latinity2020.

[11] "Lorem Ipsum," https://loremipsum.io/es/, recuperado el 02 de marzo de 2022.

[12] "DataTables," https://datatables.net/, recuperado el 27 de diciembre de 2021.

[13] "Amazon Web Services," https://aws.amazon.com/, recuperado el 27 de diciembre de 2021.

[14] "Amazon Elastic Compute Cloud," https://aws.amazon.com/es/ec2/, recuperado el 27 de diciembre de 2021.

[15] "Amazon Linux," https://aws.amazon.com/es/amazon-linux-ami/, recuperado el 27 de diciembre de 2021.

[16] "Nginx," https://www.nginx.com/, recuperado el 27 de diciembre de 2021.

[17] “Web Server Gateway Interface,” https://www.fullstackpython.com/wsgi-servers.html, recuperado el 27 de diciembre de 2021.

[18] “Validez Encuesta SUS,” https://www.usability.gov/how-to-and-tools/methods/system-usability-scale.html, recuperado el 02 de marzo de 2022.

ANEXOS

Anexo A

Capturas de páginas del sistema

A continuación se presentan capturas de las sección de trabajos uniformizada.

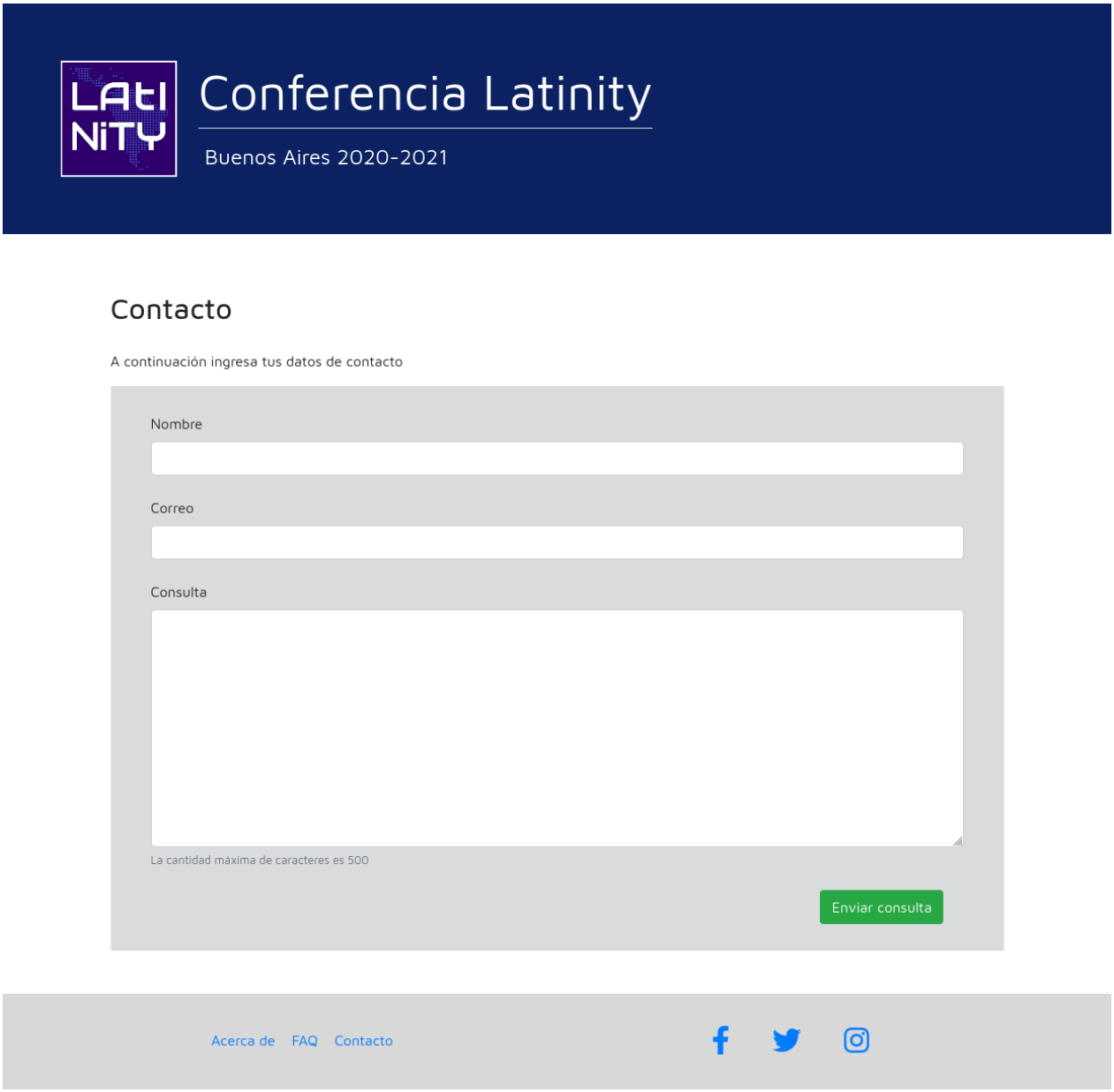

Figura A.1: Página de contacto del sistema

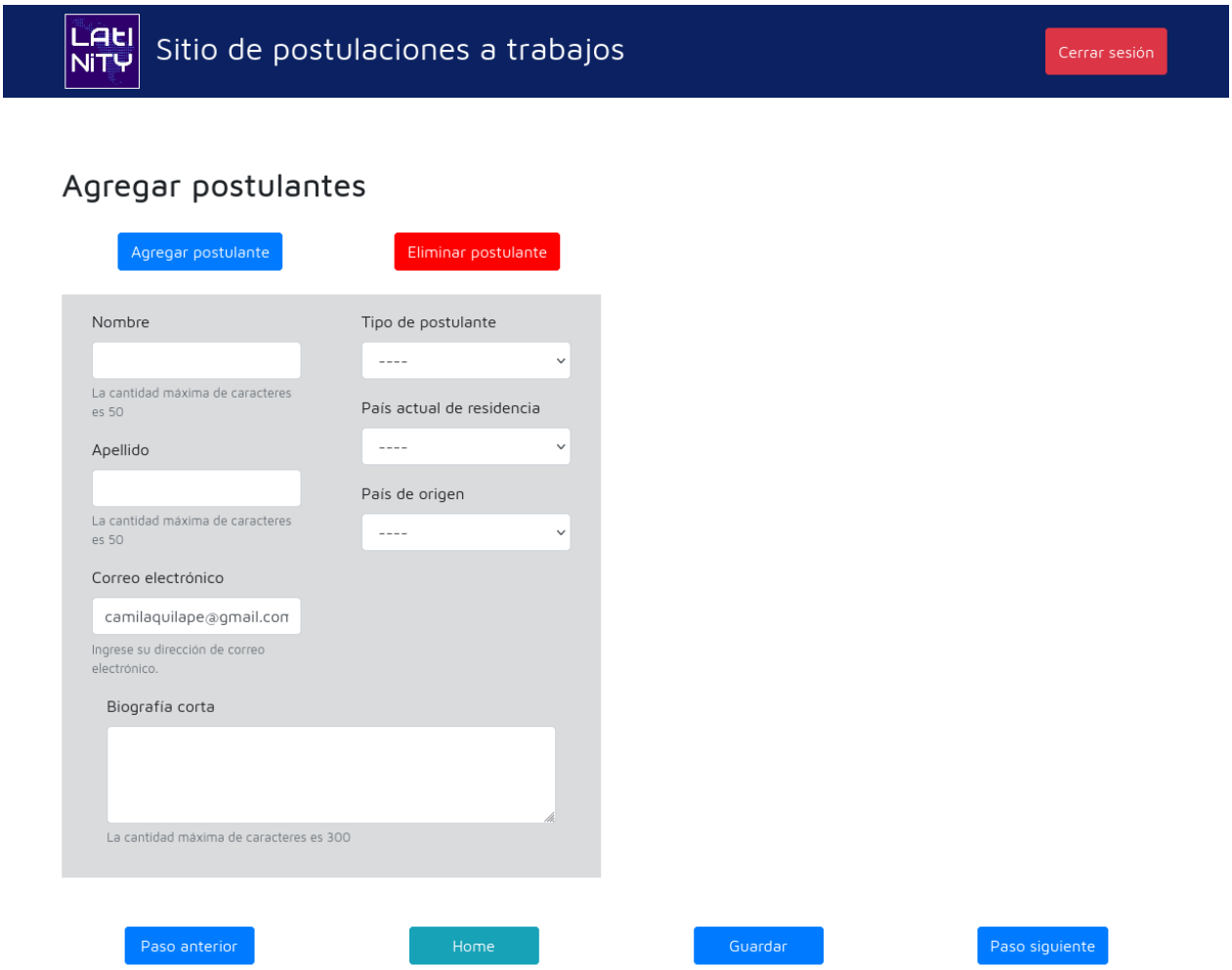

Figura A.2: Paso 2 de postulación a trabajos

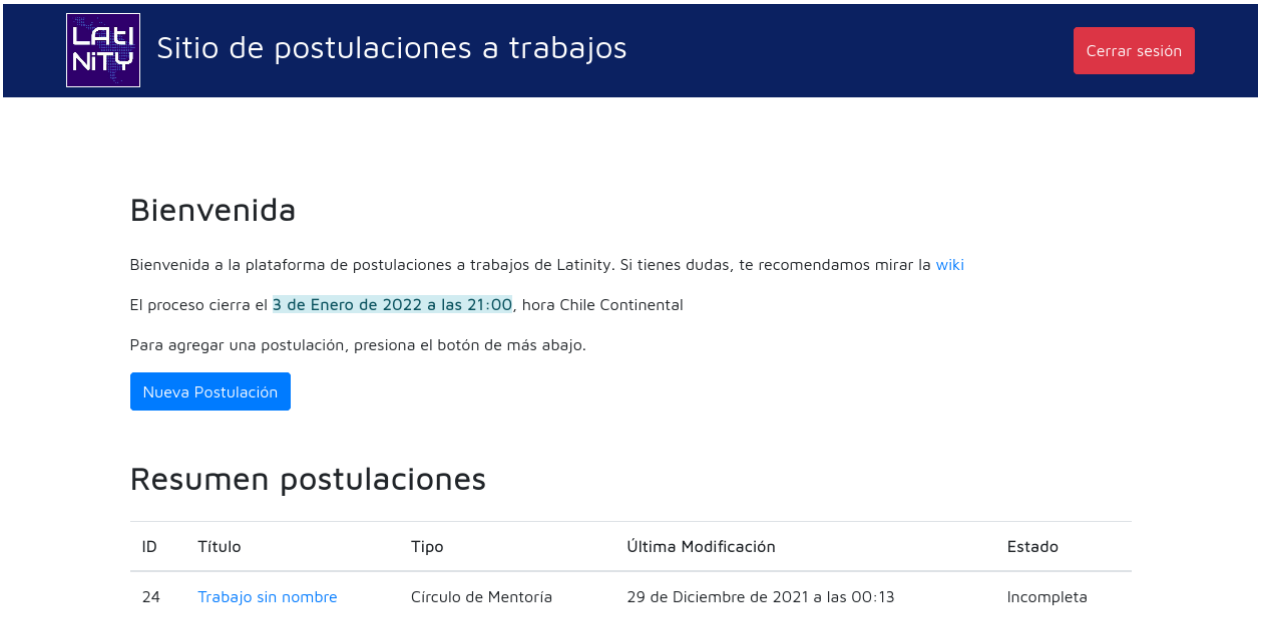

Figura A.3: Página de inicio de postulación a trabajos

Figura A.4: Paso 3 de postulación a trabajos

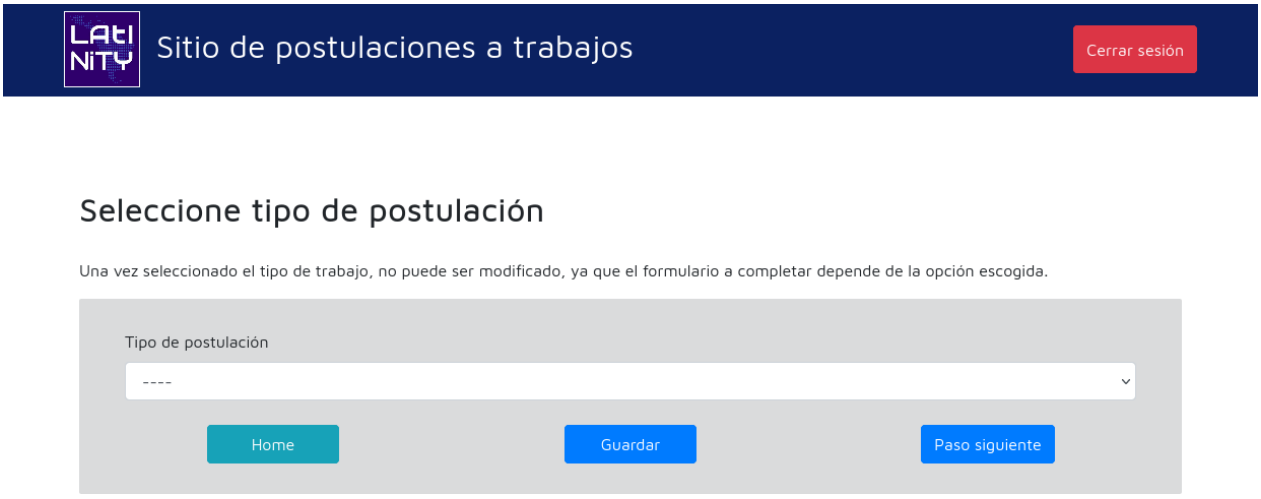

Figura A.5: Paso 1 de postulación a trabajos

Anexo B

Comentarios de la validación

B.1. Comentarios de la parte de postulaciones

1. Links a edición inconsistentes entre becas y trabajos, tal vez que sea toda la fila clickeable o el link en un botón editar. En becas el enlace está en el id y en trabajos en el nombre del trabajo. En el identificador es poco visible. Aquí mismo, sacar el identificador de la base de datos, ya que no aporta información al usuario.
2. En el formulario, decir que el video de presentación se trata de un link y mencionar que deben asegurarse que los enlaces sean visibles. En general, campos más descriptivos, en cuanto a lo que se pide. También se podrían implementar faltas de ortografías en las cajas de textos.
3. Mencionar qué significa el botón guardar (confunde un poco con el botón paso siguiente) y que para eliminar un postulante en trabajos, se debe presionar eliminar y luego guardar para hacer efectivo el cambio. Quizás agregar un modal de confirmación, ya que es una operación delicada. Asimismo agregar el texto de que se pueden agregar hasta ocho postulantes y eliminar botón eliminar postulante cuando solo hay uno.
4. Se podría repetir la información del faq en la wiki de trabajos.
5. En la parte de integrantes de un grupo de trabajo, se podría implementar el eliminar un integrante en particular y no solo el último, como está ahora.
6. El símbolo @ está raro en la parte del correo del recomendador.
7. Se ven igual login de becas y trabajos, falta un título que indique la parte dónde se encuentra el usuario.
8. Poner si el proceso está abierto o cerrado en home del postulante. Aquí mismo, tal vez agregar la hora en formato estándar (UTC)
9. Acordar qué títulos usar, hay partes con bienvenida, revisore, etc.
10. Abrir carta en otra pestaña, al momento de seleccionarla. Poner los links clickeables en resumen de becas y trabajos.

11. Revisar colores de los botones paso anterior, home, guardar y paso siguiente, junto a su ubicación. Paso anterior y paso siguiente podrían ser del mismo color y diferentes al resto. Cambiar texto del botón home por inicio.
12. En lugar de botón enviar, poner paso siguiente o ir a resumen en el paso 3 de postulaciones a becas y trabajos, ya que no se envía la postulación realmente ahí, si no en resumen, al confirmar envío.
13. Quitar botón confirmar envío al momento de editar el formulario, para que no lleguen correos innecesarios, esto luego de haber enviado la postulación. Tal vez también quitar los botones guardar en este mismo caso.
14. Falta información en general de qué es Latinity, cómo partió, versiones anteriores, a qué se puede postular, misión, etc.
15. Se cayó con correo aaaa@aaa en el registro de postulaciones.
16. Tal vez poner el nombre del trabajo y tipo de postulación en el paso 1 de trabajos, de modo de crear postulaciones sólo cuando se tenga el nombre y tipo de trabajo y así no tener trabajos sin nombre hasta que se llegue al paso 3.
17. Dejar los campos de trabajos con 500 caracteres y el título del trabajo con 300. Tal vez ir contando los caracteres x/500 por ejemplo.
18. Campos obligatorios con *
19. Faltan los tildes en países
20. Ver los números de los pasos en becas cuando no hay paso 2.
21. Fijarse en cómo queda home con nombres de trabajo largos.
22. Revisar el autocompletado en trabajos, tal vez mencionar que en el primer formulario de integrantes deben estar los datos de quién está realizando la postulación, del editor.
23. En círculo de mentoría, tal vez poner experiencia en la información personal.
24. Cambiando la url se pueden revisar las otras postulaciones. Falta agregar el token en los filtros, ahora está solo el id.
25. Diferenciar el correo de registro, diciendo que se trata del token de trabajos o de becas.
26. Falta el track desarrollo profesional en la wiki.
27. Decir que home lleva a la página de inicio sin guardar lo ingresado.

B.2. Comentarios de la parte de revisiones

1. Pedir link a repositorio Github, a página personal Github o página del proyecto como un campo extra para el postulante.
2. Hacer más visible el track, dejándolo arriba o destacándolo.

3. En trabajos, se podría agregar una caja de comentarios extra para otorgar feedback al postulante.
4. Cambiar nombre de la columna puntaje en home por la pregunta a la que hace referencia.
5. Que el enlace a hoja de vida y video sean clickeables.
6. Que la carta de recomendación se abra en otra ventana, ya que así es más usable.
7. Texto CV del formulario y campo enlace a hoja de vida son inconsistentes en revisión de becas, usar el mismo término.
8. Agregar definición de aislamiento, debido a que es muy importante para la correcta revisión de postulaciones.
9. Poner que el campo comentario es para la comisión organizadora.
10. Agregar columna aislada en la tabla de becas del home del revisor.

B.3. Comentarios de la parte de administración

1. Evaluar el botón volver junto a los posibles veredictos.
2. Se podría crear una ventana de confirmación antes de efectivamente cambiar el estado de una postulación, es decir, antes de asignar un veredicto.
3. Revisar la redacción del acordeón de la página de inicio.
4. Similar a la sección de postulaciones, ajustar los títulos de las páginas y los nombres de las columnas en las tablas.
5. Agregar instrucciones en la sección de asignaciones para que quede claro cómo se realizan usando los archivos csv.