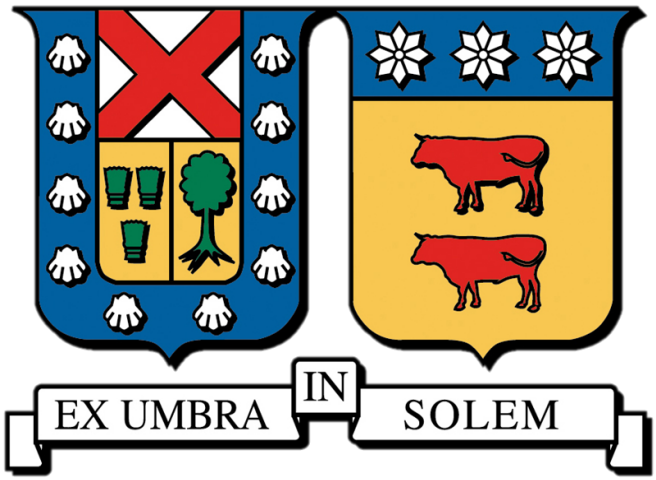
UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE INFORMÁTICA
VALPARAÍSO - CHILE



Actualización de Diseño, Desarrollo e Implementación de Sistema de Postulación a Ayudantías (SPA) del Departamento de Informática de la UTFSM

INTI GAËL PONTT LE GALL

MEMORIA DE TITULACIÓN PARA OPTAR AL TÍTULO DE INGENIERO DE EJECUCIÓN EN INFORMÁTICA

PROFESOR GUÍA: CECILIA REYES

PROFESOR CORREFERENTE: JOSÉ LUIS MARTÍ

NOVIEMBRE 2019

RESUMEN

El ***S**istema de **P**ostulación a **A**yudantías* (SPA) usado por el ***D**epartamento de **I**nformática* (DI) de la ***U**niversidad **T**écnica **F**ederico **S**anta **M**aría* (UTFSM) ha estado en funcionamiento desde el año 2004 para el campus Casa Central. Este sistema permite a alumnos y profesores administrar sus ayudantías sin recurrir a formularios en papel. La última actualización importante que recibió el sistema fue en el 2012 en la cual se integró el *Departamento de Ingeniería Metalúrgica y de Materiales* (DIMM) y los campus de Santiago (San Joaquín y Vitacura).

El presente informe tiene como finalidad documentar el proceso de desarrollo de una nueva versión del SPA que busca corregir problemas que presenta la versión actual del sistema, así como incorporar requerimientos nuevos, junto con mejorar y modernizar el sitio web del sistema. Los principales cambios realizados al SPA en esta nueva versión son:

- Actualización de la interfaz y experiencia del usuario.
- Mejoras en las funcionalidades de los distintos perfiles de usuario del sistema.
- Uso de herramientas modernas y buenas prácticas para facilitar el desarrollo y, posteriormente, la mantención del sistema.
- Implementación de un sistema robusto de roles y permisos para los distintos perfiles de usuario.
- Incorporación de métricas de apoyo a la gestión y auditoría de datos.

ABSTRACT

The SPA system (Spanish: ***S**istema de **P**ostulación a **A**yudantías*) is an Assistantship system used by the *Informatics Department* (Spanish: ***D**epartamento de **I**nformática*) (DI) of the *Federico Santa María Technical University* (Spanish: ***U**niversidad **T**écnica **F**ederico **S**anta **M**aría*) (UTFSM). It has been online since 2004 for the Casa Central Campus. This system helps students and teachers manage their assistantships without resorting to paper applications. SPA's last important update was in 2012 where both the *Metallurgical and Material Engineering Department* (Spanish: *Departamento de Ingeniería Metalúrgica y de Materiales*) (DIMM) and the campi located in Santiago (San Joaquín Campus and Vitacura Campus) for all included departments were added into the system.

The following document's purpose is to document the development process of a new version for the SPA system aiming to fix problems present in the system's current version, as well as including new requirements, upgrading and modernizing the system's website. The main changes to be made in this new version are the following:

- Upgrade on the UX and the UI.
- Improvements on features for all user types.
- Usage of modern tools and good practices to facilitate development and maintaining of the system.
- Implementation a robust roles and permissions system for all user types.
- Incorporation of metrics for management and data audit.

GLOSARIO

- ➢ *BPM*: Modelado de procesos de negocio (Inglés: **B**usiness **P**rocess **M**odelling).
- ➢ *DBMS*: Sistema de Gestión de Bases de Datos (Inglés: **D**ata**b**ase **M**anagement **S**ystem).
- ➢ *DEA*: **D**irección de **E**nseñanza y **A**prendizaje.
- ➢ *DI*: **D**epartamento de **I**nformática.
- ➢ *DIMM*: **D**epartamento de **I**ngeniería **M**etalúrgica y de **M**ateriales.
- ➢ *DTI*: **D**irección de **T**ecnologías de la **I**nformación.
- ➢ *EAA*: **E**scuela de **A**sistentes de **A**prendizaje.
- ➢ *Laravel*: Framework MVC de PHP creado por Taylor Otwell.
- ➢ *CI/CD*: Integración Continua (Inglés: **C**ontinuous **I**ntegration/**C**ontinuous **D**elivery).
- ➢ *Jenkins*: Servidor para CI/CD.
- ➢ *REPL*: **R**ead-**E**val-**P**rint-**L**oop.
- ➢ *PHP*: **P**HP: **H**ypertext **P**reprocessor (acrónimo recursivo).
- ➢ *PSR*: Estándares recomendados de PHP (Inglés: **P**HP **S**tandards **R**ecommendations).
- ➢ *RDBMS*: Sistema de Gestión de Bases de Datos Relacionales (Inglés: **R**elational **DBMS**).
- ➢ *UTFSM*: **U**niversidad **T**écnica **F**ederico **S**anta **M**aría.

ÍNDICE

INTRODUCCIÓN

La presente memoria muestra el desarrollo de la nueva versión del SPA, la cual surge por la necesidad de actualizar el sistema actual para proveer una mayor escalabilidad y mejorar la usabilidad de sus interfaces.

Esta nueva versión sigue integrando a los departamentos ya adheridos al sistema (DI y DIMM, en los campus Casa Central, San Joaquín y Vitacura), y plantea la posibilidad de integrar nuevos departamentos posteriormente, sin hacer alteraciones mayores.

Se considera la funcionalidad ya existente del sistema así como requerimientos nuevos, dejando de lado funcionalidades que han caído en desuso y/o no se ajustan a la realidad actual. Se desarrolla la documentación y el análisis para cada uno de los requerimientos principales, comparando con los requerimientos de la versión anterior y posteriormente, se plantean posibles mejoras y proyecciones a ser consideradas para el desarrollo futuro y la evolución del sistema.

En el primer capítulo, se realiza una descripción del problema y de la situación actual en lo que respecta a la segunda versión del SPA. Se enumeran los distintos procesos que intervienen en la postulación y selección de ayudantes docentes a nivel de departamento (usando Informática como ejemplo), se explican los problemas que posee el modelo de datos actual y se listan los distintos objetivos que se busca cumplir en el desarrollo de esta nueva versión.

En el segundo capítulo, se analizan las distintas facetas del sistema a construir y se enuncian las prácticas, conceptos claves que serán utilizados en el diseño de éste y la integración que se tendrá con sistemas ya existentes del DI. Se comienza definiendo los distintos actores del sistema y se realiza un análisis de los requerimientos cubiertos por la versión actual, así como la necesidad de conservar su funcionalidad. Luego, se explica los requerimientos nuevos para los distintos perfiles de usuario a ser integrados en el sistema, junto con las estrategias de autorización.

En el tercer capítulo, se explica en más detalle el modelo de datos, la estructura que tiene el *framework* a usar y algunas de las más importantes librerías o dependencias a utilizar en el desarrollo de éste.

Finalmente en el cuarto capítulo se enumeran las distintas conclusiones a las cuales se llega después de desarrollar un sistema tan grande, las principales dificultades y oportunidades que tendrá en nuevo sistema para crecer y evolucionar en el futuro.

Capítulo 1

DESCRIPCIÓN DEL PROBLEMA

1.1 Antecedentes Previos

La versión actual del Sistema de Postulación a Ayudantías docentes del Departamento de Informática nació como una mejora al sistema, que partió el año 2004, con el fin de gestionar el proceso de ayudantías docentes de pregrado del DI. Esta mejora buscó agregar más funcionalidades a un sistema excesivamente simple e integrar a los campus de Santiago al mismo tiempo, en donde se seguía manejando las ayudantías con formularios en papel. Esta mejora les permitió a los alumnos de la Casa Central realizar ayudantías en Santiago y vice-versa a través del sistema. (Urso, A. 2012, p2).

El proceso de ayudantías docente del DI es llevado a cabo cada semestre por la Subdirección de Pregrado, entidad compuesta por los puestos de Subdirector/a de Pregrado y los jefes de carrera de los campus Casa Central y Santiago (San Joaquín/Vitacura). En la figura 1.1.1, se pueden apreciar los distintos subprocesos que intervienen en las ayudantías docentes.

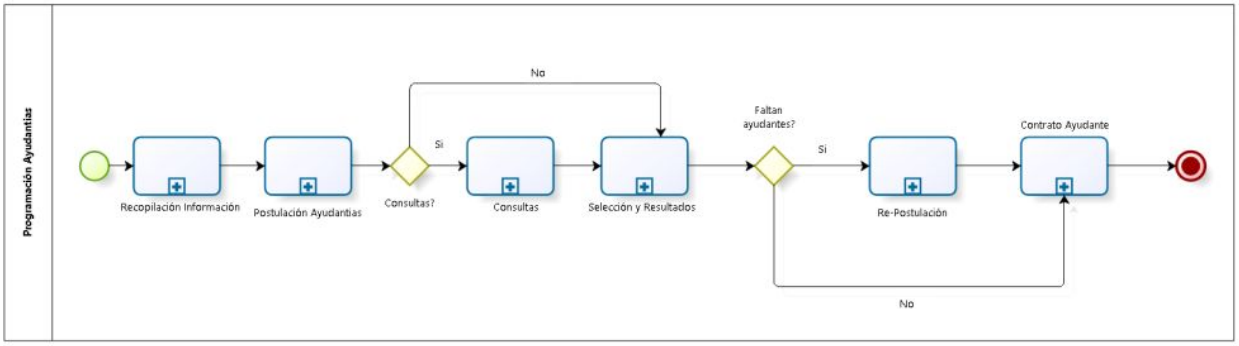


Figura 1.1.1: Modelado BPM del proceso de ayudantías docentes. Este proceso está dividido en varios subprocesos. Fuente: Muñoz, M (2016, p51)

En las figuras 1.1.2 y 1.1.3 se puede observar que los subprocesos de recopilación de información son ligeramente distintos para los campus de Valparaíso y Santiago, siendo esta diferencia justificada porque los cargos en Santiago tienen mayor concentración de tareas al existir menos personal de apoyo.

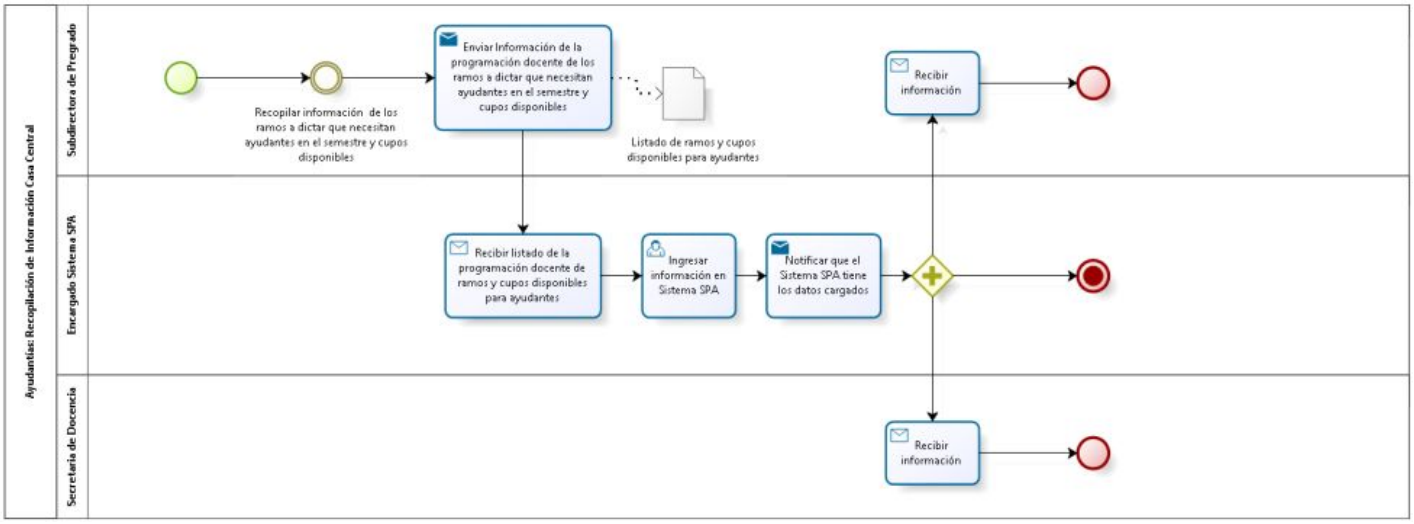


Figura 1.1.2: Modelado BPM de los subprocesos de recopilación de información para campus Casa Central.
Fuente: Muñoz, M (2016, p52)

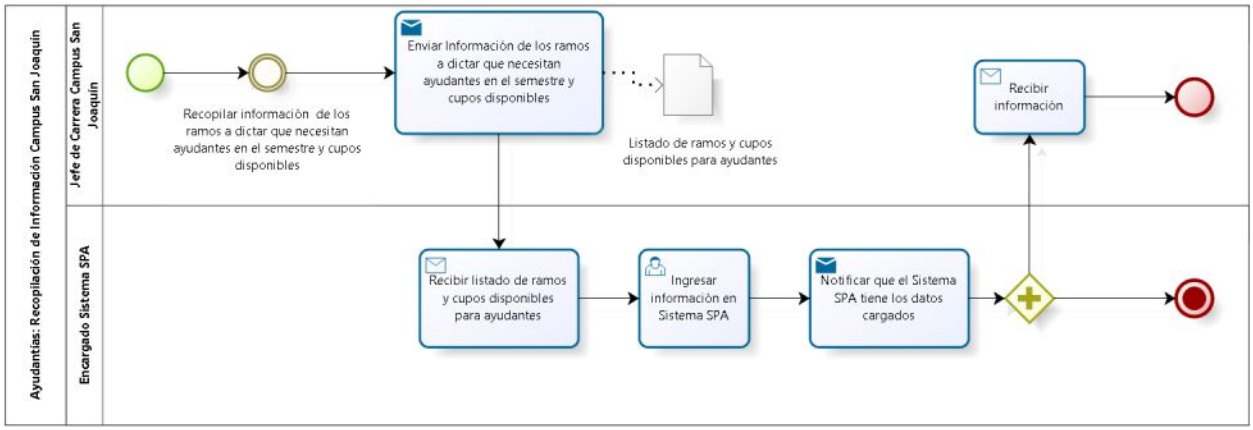


Figura 1.1.3: Modelado BPM de los subprocesos de recopilación de información para campus San Joaquín.
Fuente: Muñoz, M (2016, p53)

Gracias al SPA, quienes quieran postular a ayudantías docentes pueden hacerlo interactuando únicamente con el sistema, tal como describe la figura 1.1.4.

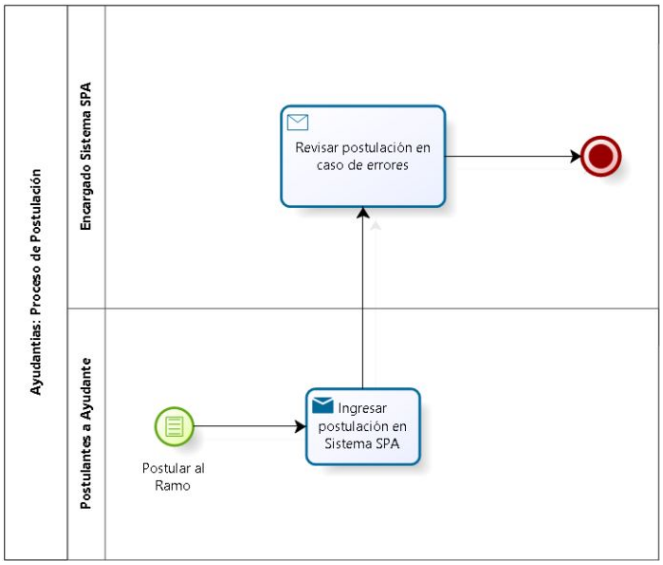


Figura 1.1.4: Modelado BPM del subproceso de postulación. Fuente: Muñoz, M (2016, p53)

La figura 1.1.5 pone en evidencia la importancia que tienen los administradores del SPA actualmente, pues son ellos los que deben resolver cada caso especial, así como todas las solicitudes que el sistema no soporta actualmente.

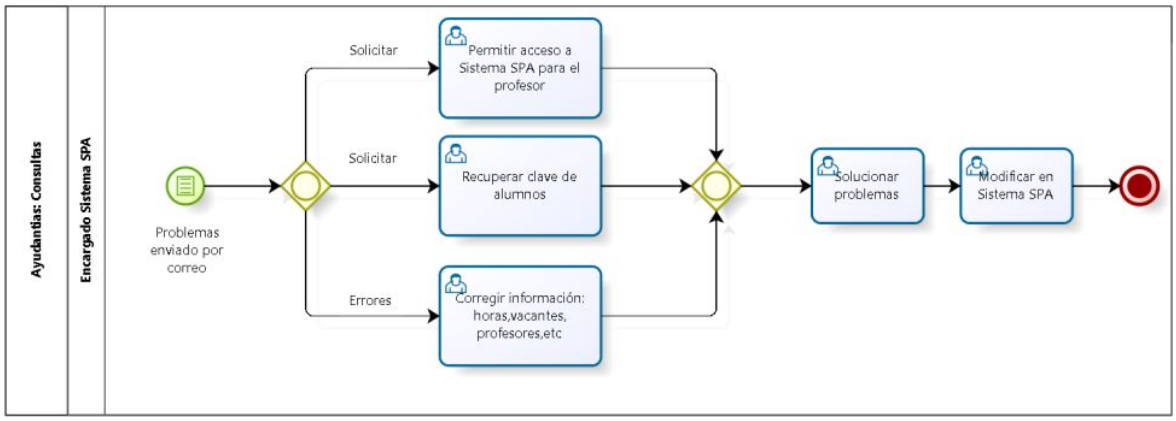


Figura 1.1.5: Modelado BPM del subproceso de consultas al administrador. Fuente: Muñoz, M (2016, p54)

El subproceso de selección (y posteriormente reselección) de ayudantes está descrito por la figura 1.1.6. Este subproceso actualmente posee una falla crítica pues los profesores o encargados de las asignaturas que seleccionan a sus ayudantes a partir de la lista de postulantes disponibles, no pueden deshacer o cambiar su selección en caso de error, debiendo recurrir a los administradores del SPA para que rectifiquen la situación directamente en la base de datos.

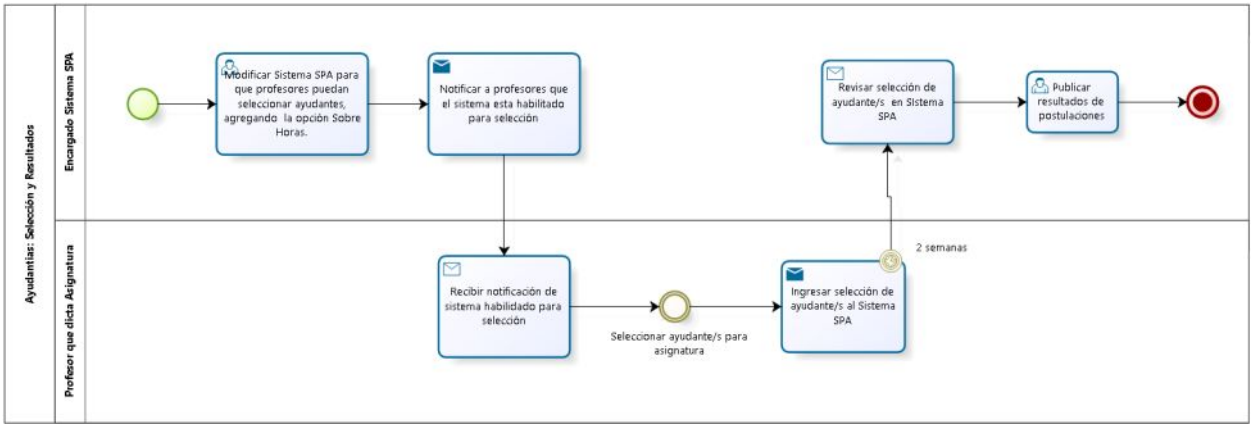


Figura 1.1.6: Modelado BPM del subproceso de selección de ayudantes. Fuente: Muñoz, M (2016, p55)

Cuando se da el caso donde quedan asignaturas sin ayudantes, se debe hacer un llamado a re-postulación (descrito en la figura 1.1.7), donde solo están disponibles aquellas asignaturas que aún poseen vacantes.

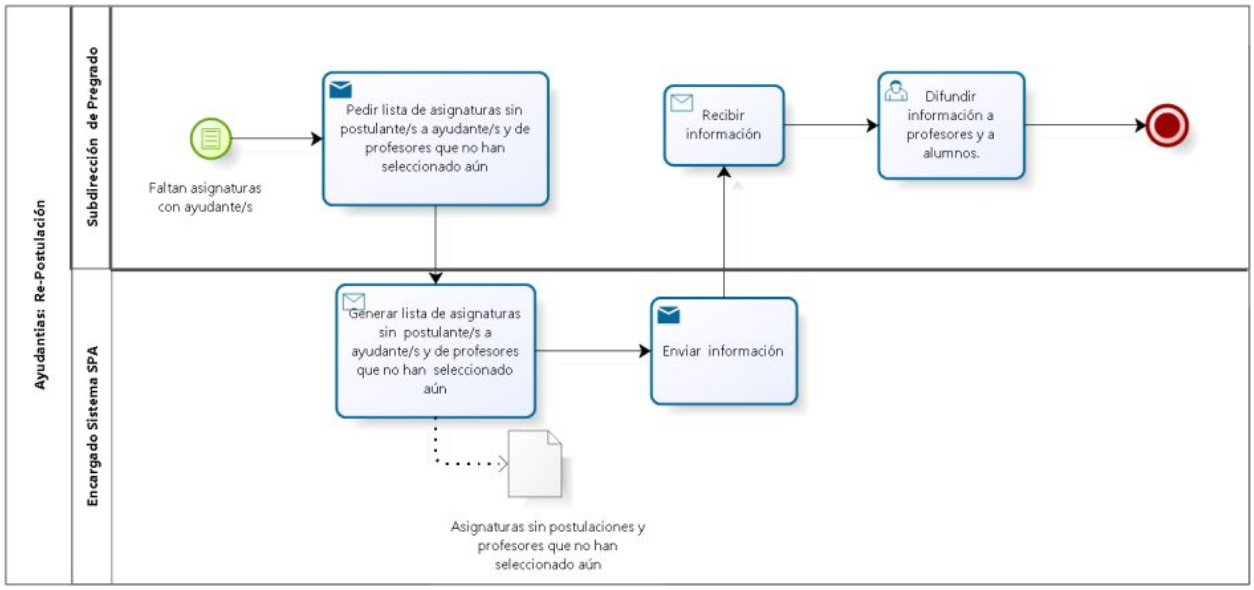


Figura 1.1.7: Modelado BPM del subproceso de re-postulación. Fuente: Muñoz, M (2016, p55)

El subproceso de contratación modelado en la figura 1.1.8, no será tomado en cuenta para el desarrollo de esta memoria pues no es responsabilidad del SPA, sino de otro sistema de la universidad.[1]

[1] http://www.vrea.usm.cl/unidades/

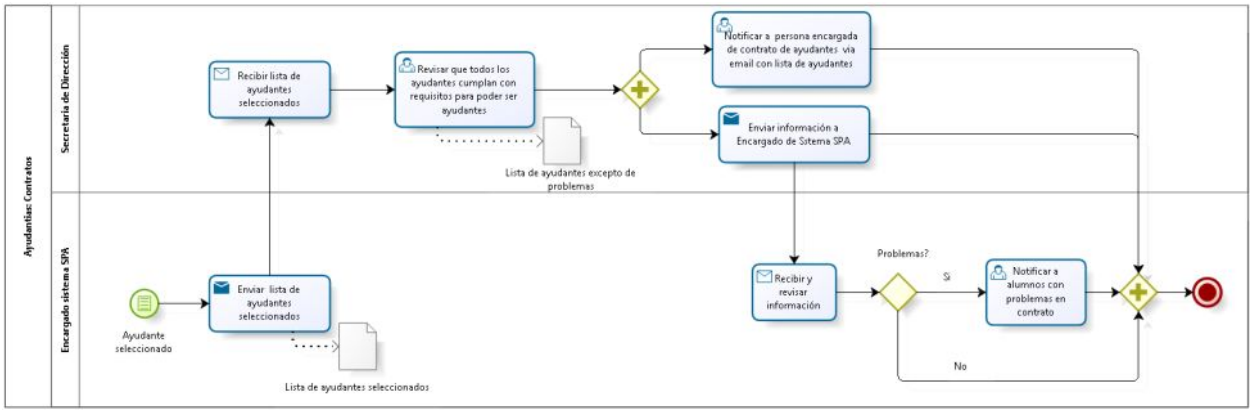


Figura 1.1.8: Modelado BPM del subproceso de contratación para ayudantes. Fuente: Muñoz, M (2016, p56)

1.2 Identificación de Problemas

Actualmente, el SPA tiene varios problemas que deben ser corregidos, tales como procesos redundantes o no usados, un código base difícil de mantener, casos de uso que no se ajustan a la realidad actual, vulnerabilidades de seguridad. A continuación se describen algunos de estos problemas.

El modelo de datos usado por el SPA, presente en la figura 1.2.1, contiene 49 tablas. Su complejidad es muy elevada, lo cual dificulta el desarrollo y la mantención del sistema. Existen relaciones y entidades que tratan de abarcar responsabilidades ajenas a las de un modelo de datos. Algunas tablas también han caído en desuso o simplemente no cumplen con su razón de ser, debido a que fueron ignoradas en las mantenciones que se han hecho al sistema a lo largo de los años, por lo cual solo contaminan el modelo de datos. Se puede observar esto en la tabla 1.2.1:

TABLA(S)	**DESCRIPCIÓN DEL PROBLEMA**
version	Guarda la versión actual, así como su fecha de lanzamiento y una descripción de los cambios importantes que tiene respecto a la versión anterior, responsabilidad que debiera ser del repositorio del proyecto. Actualmente no cumple su propósito y ha caído en desuso.
bugs, **estado_bug**	La tabla **bugs** guarda descripciones de problemas encontrados con el sitio web, así como una referencia a su estado, cuya glosa está guardada en la tabla **estado_bug**. Esta responsabilidad debería ser de un *issue tracker* o en su defecto, del repositorio del proyecto. Actualmente no cumplen su propósito y han caído en desuso.
log_asignacion_yudante, **log_ayudante,** **log_modificacion_horas**	La función de estas tablas es la de guardar información acerca de los cambios que ocurren en el sistema en ciertos momentos del proceso con tal de poder responder a las preguntas *¿Quién hizo qué y cuándo lo hizo?* Sin embargo, la información guardada en ellas es insuficiente y requiere consultar otras tablas a la vez para obtener más información. Actualmente no cumplen su propósito y han caído en desuso.
semestre	La función de esta tabla es la de guardar la fecha de inicio y término de cada semestre, con tal de ser referenciada por otras tablas. Sin embargo, no es referenciada correctamente.
asig_campus_historico, **criterio_asignatura_historicos,** **criterio_historico,** **postulacion_historica,** **solicitudes_historicas**	Estas tablas guardan información acerca de cada proceso después que éste se termina. Sin embargo, el administrador debe ejecutar unos scripts manualmente al término de cada proceso para que estas tablas sean pobladas y además es complejo manejar las múltiples asociaciones para recopilar los datos históricos, fallando en su propósito.

menu, **menu_permiso**	Estas tablas guardan información para completar enlaces del sitio dependiendo del permiso (rol) que tenga el usuario (Urso, A., 2012, p22). Ésta información no debiera existir en el modelo de datos sino en la lógica del sitio web.
contratos, **contratos_alumnos**	En la segunda iteración del SPA, se consideró que la administración de contrato por parte de los jefes de carrera y de los administradores del sistema iba a ser una labor fundamental (Urso, A., 2012, p24). Sin embargo, el manejo de contratos, y por consiguiente de las remuneraciones, no es una responsabilidad que debería tener el sistema a cargo de las ayudantías docentes y no son funciones que se estén ocupando actualmente.
valores	Esta tabla guarda el avalúo por hora que se remunera a cada ayudante dependiendo del tipo de hora (corrección, contacto, laboratorio) y de su nivel (A, B, o C, el cual está dado por su experiencia previa). Tiene que ver con remuneraciones lo cual se escapa de las responsabilidades que debería tener este sistema.
solicitud_sobrehoras, **solicitud_sobrehoras_asignaturas**	Estas tablas se llenan solo usando un trigger SQL que el administrador activa a lo largo del proceso cuando este pase a la etapa 'Selección Pendientes' según las especificaciones descritas en (Urso A., 2012, p125-126). Actualmente no se usan, por lo tanto las tablas permanecen vacías.
registro_rechazo_ayudante	Esta tabla no ha sido usada desde el 2014 y no está referenciada en la documentación.

Tabla 1.2.1: Problemas del modelo de datos de la versión actual del SPA.

Sin tomar en cuenta las vistas SQL, las 20 tablas mencionadas anteriormente componen aproximadamente un 40% del modelo de datos (49 tablas), tal como se puede apreciar en la figura 1.2.2

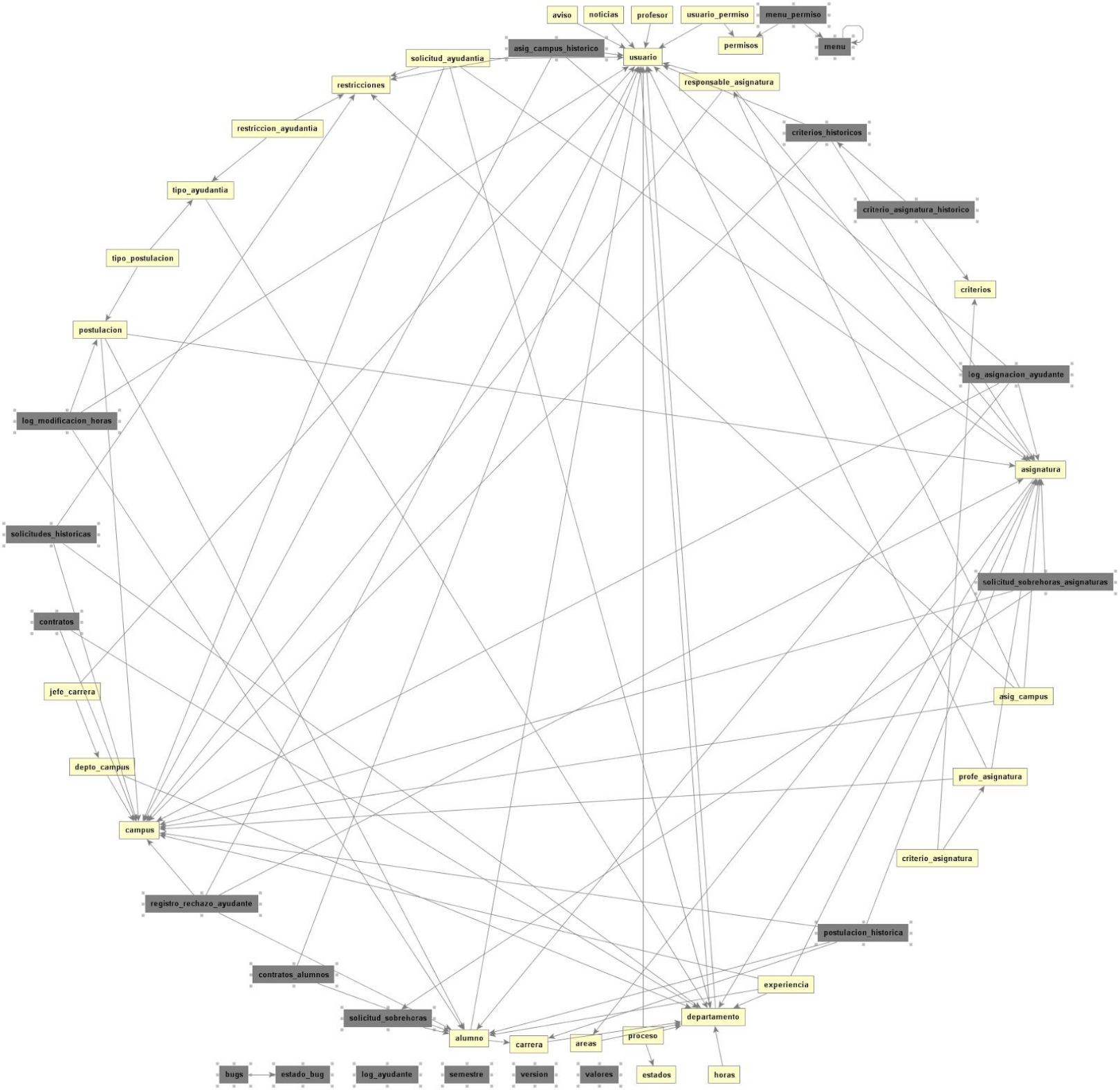


Figura 1.2.2: Representación circular del modelo de datos actual del SPA.
Las tablas oscurecidas contaminan el modelo de datos. Fuente: Elaboración propia con DbVisualizer.

Los alumnos pueden ingresar sus propios datos académicos, lo cuales pueden no ser verídicos pues no son verificados (ver figura 1.2.3). También pueden omitir sus postulaciones a ayudantías actuales fuera del DI y DIMM, así como su estado actual, lo cual hace el proceso de selección de ayudantes más difícil de lo que debe ser por el límite de horas semanales que un alumno puede trabajar en ayudantías pagadas según el reglamento de la UTFSM. No respetar este límite ocasiona problemas a futuro en los contratos.

Figura 1.2.3: Interfaz para la modificación de datos personales, accesible por usuario Alumno. Fuente: Urso, A. (2012, p32)

Cuando profesores seleccionan un estudiante, no pueden deshacer la selección y deben acudir a los administradores para que reviertan la operación directamente en la base de datos. Navegar por el sitio también es complicado ya que los listados de postulantes no pueden ser reordenados o filtrados y la información que dan puede ser crítica para un usuario nuevo.

Los administradores, no conformes con el diseño del sitio, usan un sistema aparte (SPA-Manager, figura 1.2.4) para administrar el SPA. Actualmente las acciones que deben realizar a lo largo de cada semestre sufren de cierta redundancia y existe información que debe ser cargada en forma manual, cuando podría obtenerse directamente de la ***D**irección de **T**ecnologías de la **I**nformación* (DTI).

Figura 1.2.4: Interfaz de SPA Manager, sitio que usan los administradores del SPA para manejar el proceso. Este sitio no es accesible externamente, y solo está disponible para la red interna del DI cuando es necesario.

Hay algunas hojas de cálculo que pueden ser exportadas del sistema por ciertos perfiles administrativos (como se muestra en la figura 1.2.5) pero los datos dentro pueden estar algo desordenados, contener caracteres basura y en algunos casos, no tener valor alguno. Existe un campo opcional que pueden llenar los postulantes para agregar una razón o motivación por la cual quieren postular a una ayudantía, pero no es usado adecuadamente y termina en una de estas hojas de cálculo.

							Listado
Paralelo	Prioridad	Motivo	Nivel	Pa	Pp	Curso	Asignatura
-1	1	Me gusto mucho el ramo cuando lo curse, por	C	4234,12	61	2015	Estructuras Discretas
-1	2	Siento tener las aptitudes suficientes para po	C	4234,12	61	2015	ProgramaciÃ³n
-1	1	porque me gustó el ramo de program	C	3644	59	2019	ProgramaciÃ³n
1	2	.	B	4000	60	2016	AnÃ¡lisis y DiseÃ±o de Softwar
1	1	Me gustarÃ-a continuar con la ayudantÃ-a.	A	5593,86	64	2015	IngenierÃ-a, InformÃ¡tica y Soc
-1	2	Por insistencia.	C	4477	69	2015	InformÃ¡tica TeÃ³rica
1	1	Agregada por Coordinador SPA	A	10096	83	2012	Bases Tecn. para la Inteligencia
1	1	:D	B	4717	61	2017	Laboratorio de ComputaciÃ³n
1	1		C	5368	70	2016	Laboratorio de ComputaciÃ³n
1	1	:)	B	4594	60	2014	Laboratorio de ComputaciÃ³n
1	2	why not?	A	8386,69	79	2015	Lenguajes de ProgramaciÃ³n
1	1	Motivos personales	C	7146,52	65	2018	Laboratorio de ComputaciÃ³n
1	1	.	A	7971	76	2014	InvestigaciÃ³n de Operaciones
1	1	Considero que puedo ser un aporte para que los	A	8108	83	2015	ComputaciÃ³n CientÃ-fica

Figura 1.2.5: Extracto de hoja de cálculo del listado de ayudantes. Puede apreciarse que se entregan caracteres basura y datos que no son de utilidad para el personal administrativo. Datos sensibles (nombre de usuario, login) fueron ocultados. Fuente: SPA

El sistema en sí no provee auditoría de datos tampoco, lo cual dificulta asegurar la integridad de los datos así como quien es responsable por los cambios que ocurren dentro de la base de datos y algunos roles ya no son usados. Como se explicó en 1.2, las tablas que fueron pensadas para ese fin no están siendo usadas correctamente.

1.3 Objetivos de una Solución

Según las problemáticas planteadas, se analizaron nuevamente los requerimientos del sistema para comenzar el desarrollo de una nueva iteración del SPA, modernizando el diseño y las herramientas de desarrollo para facilitar así su mantención a futuro. Se reconocen como objetivos general:

- Implementar una actualización al sistema existente con un nuevo diseño sencillo y responsivo, considerando nuevos requerimientos aparte de las funcionalidades que ya provee el sistema actual.

En cuanto a objetivos más particulares, se reconocen los siguientes:

- Usar un *framework* moderno para el desarrollo del sistema
- Desarrollar un sistema sencillo de mantener protegido por roles y permisos fáciles de comprender e implementar.
- Documentar las funcionalidades y funciones internas del sistema usando pruebas automatizadas de bajo y alto nivel.
- Diseñar una interfaz sencilla y flexible, que permita a los usuarios desempeñar sus labores fácilmente.
- Implementar el sistema en una máquina virtual del DI para su futura mantención.
- Utilizar patrones de diseño conocidos y buenas prácticas de programación en el desarrollo del sistema.
- Permitir la semi-automatización (carga masiva) de las vacantes de ayudantías en cada semestre, disminuyendo el costo de mantención en el proceso.

Capítulo 2

ANÁLISIS Y DISEÑO DE LA SOLUCIÓN

2.1 Actores del Sistema

Teniendo en consideración el estado actual del SPA, así como sus funciones y los usuarios que interactúan con el sistema, se identifican los siguientes actores:

Administrador de Sistema: (en inglés *SysAdmin*) es el encargado de mantener el sistema funcionando. Sus labores son completamente administrativas y si bien tiene acceso a todas las funcionalidades del sistema, no requiere interactuar mayormente con éste. Es el único actor que puede interactuar directamente con la base de datos, así como ver los registros de actividad.

Administrador de Departamento: este actor es similar al Administrador del sistema, pero sesgado a un solo departamento. Corresponde al encargado de gestionar los procesos. Su principal función es la de cargar masivamente información acerca de las ayudantías docentes que son ofrecidas por el departamento así como administrar las fechas del proceso de postulación cada semestre.

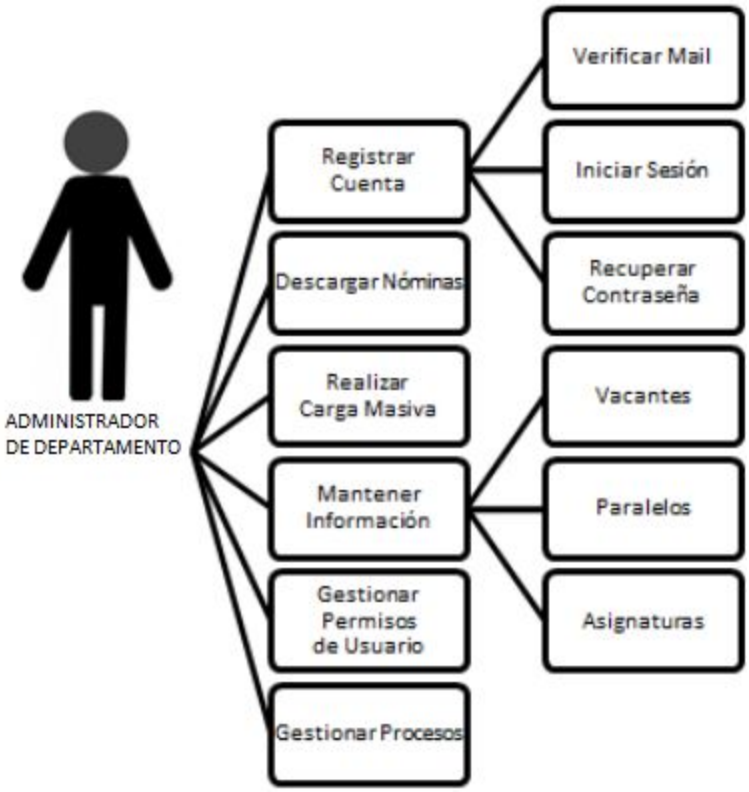


Figura 2.1.1: Estructura de funciones para actor Administrador de Departamento. Fuente: Elaboración propia.

Subdirección de Pregrado: este actor corresponde tanto a los Jefes de Carrera así como al(a la) Subdirector(a) de Pregrado de cada departamento incluido en el SPA. Tienen acceso a toda la información perteneciente a su departamento (sesgado además por campus para los jefes de carrera). Si bien puede realizar la gestión del proceso, su principal función es la de observar el desarrollo de los procesos de ayudantía y, similar a Secretaría, poder exportar datos cuando necesite.

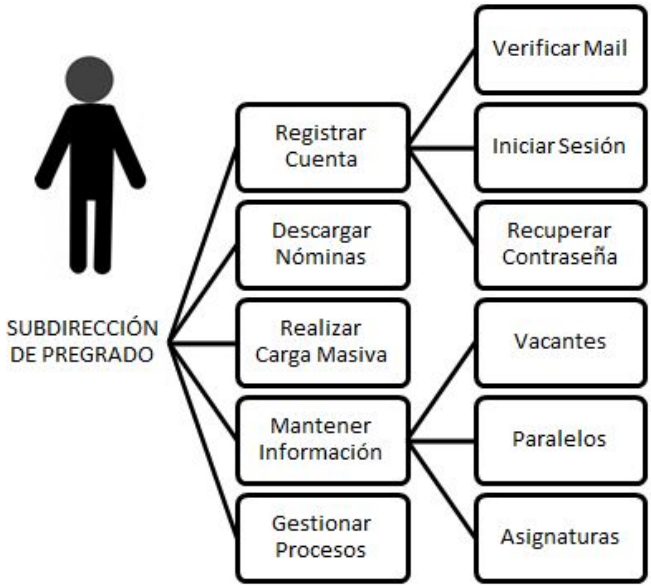


Figura 2.1.2: Estructura de funciones para actor Subdirección de Pregrado. Fuente: Elaboración propia.

Profesor: este actor corresponde a los profesores encargados de los distintos paralelos de asignaturas que ofrecen vacantes a ayudantías. No necesariamente son quienes dictan dicho paralelo. Pueden ver información acerca de las asignaturas y paralelos de los cuales están encargados. Sus principales funciones son las de seleccionar ayudantes entre las postulaciones disponibles en el sistema y calificarlos al final del semestre.

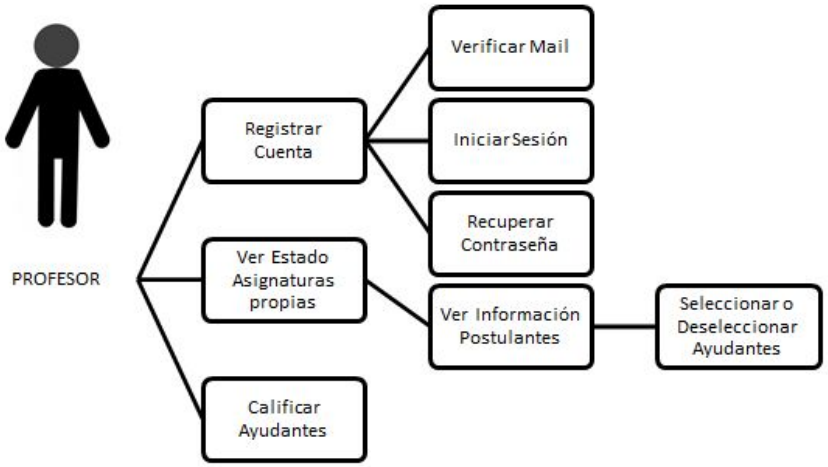


Figura 2.1.3: Estructura de funciones para actor Profesor. Fuente: Elaboración propia.

Ayudante: este actor corresponde a aquellos alumnos de la USM que deseen postular a ayudantías docentes en el sistema. Sus principales funciones son las de actualizar sus datos académicos y postular a las ayudantías que ofrece el sistema. Pueden ver información acerca de todas las asignaturas que tengan vacantes disponibles.

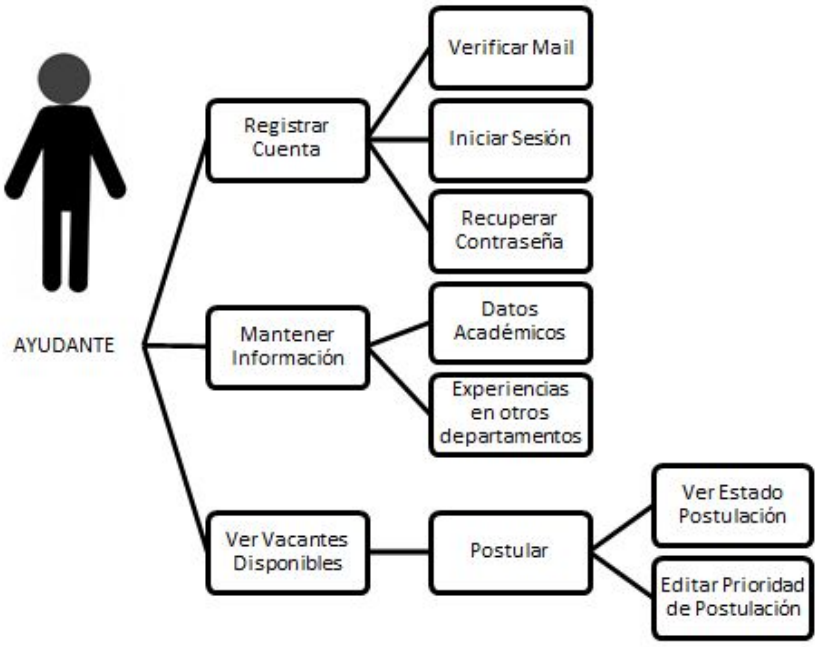


Figura 2.1.4: Estructura de funciones para actor Ayudante. Fuente: Elaboración propia.

Secretaría de Pregrado: Este actor corresponde a la Secretaría de Pregrado de cada departamento. Su principal función es la de exportar información del sistema en forma de gráficos y planillas de datos con formatos dados. En el Campus San Joaquín, esta tarea la realiza el Jefe de Carrera.

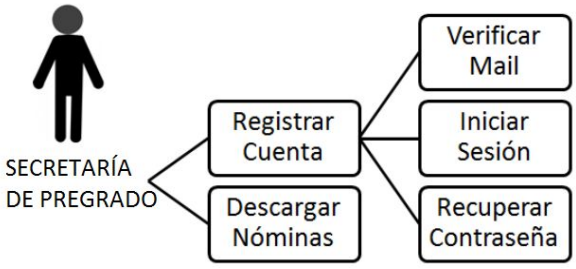


Figura 2.1.5: Estructura de funciones para actor Secretaría de Pregrado. Fuente: Elaboración propia.

A grandes rasgos, una aproximación de la estructura jerárquica de los distintos actores se puede apreciar en la figura 2.1.6.

Figura 2.1.6: Representación jerárquica de los roles del sistema. Fuente: Elaboración propia.

2.2 Análisis de Requerimientos Previos

La documentación del sistema actual (Urso, A., 2012) especifica los requerimientos básicos que debería cumplir cada tipo de usuario, así como el sistema en sí, siendo la base para la nueva iteración. Para cada requerimiento, se marcará con una cruz (X) si no se mantendrá o con una tic (✓) de lo contrario, dando una explicación para cada caso.

2.2.1 Requerimientos del Sistema

- ➢ Debe permitir el ingreso de los usuarios a través de los datos obtenidos desde la cuenta del Departamento de Informática y registrarlo automáticamente en caso que no tenga una cuenta previamente asociada.
  - X Si bien es factible utilizar las cuentas del DI para iniciar sesión, esto no va acorde a un sistema que no va a ser utilizado exclusivamente por el Departamento de Informática. Utilizar las cuentas institucionales (Pasaporte USM) sería más correcto.
- ➢ Debe permitir el registro de alumnos externos al departamento.
  - ✓ Registrar usuarios nuevos que provengan o no de los departamentos adheridos al SPA es una funcionalidad que seguirá en pie en esta nueva iteración del sistema.
- ➢ Debe permitir la modificación de la información personal propia del usuario conectado en ese momento.
  - X Cada usuario podrá editar su imagen de perfil y opcionalmente agregar información adicional. Sin embargo, el sistema no debería requerir que se llene información personal.
- ➢ Debe registrar las experiencias de los ayudantes seleccionados de manera automática.
  - ✓ Esta funcionalidad seguirá en pie en esta nueva iteración del sistema.

2.2.2 Requerimientos del Alumno

Este tipo de usuario corresponde al actor definido en [2.1] como **Ayudante.**

- ➢ Debe poder postular en asignaturas del DI.
  - ✓ Esta funcionalidad seguirá en pie en esta nueva iteración del sistema, aunque no está limitada solamente al DI. Cualquier alumno es libre de postular a cualquier vacante ofrecida, sea esta del departamento de su carrera o no. La selección por otro lado, queda a criterio de los profesores a cargo.
- ➢ Debe poder ingresar y modificar sus datos académicos.
  - ✓ A pesar de que uno de los problemas del SPA actual es la cuestionable veracidad de los datos académicos ingresados por sus usuarios, esta funcionalidad seguirá en pie en esta nueva iteración del sistema. En un principio, se planteaba como una posible solución obtener los datos académicos directamente de la base de datos de la Universidad, pero debido a restricciones que conciernen la política de privacidad de datos de los alumnos, no es factible por ahora.
- ➢ Debe poder registrar sus experiencias de ayudante en departamentos no adheridos al sistema.
  - ✓ Esta funcionalidad seguirá en pie en esta nueva iteración del sistema.

2.2.3 Requerimientos del Profesor

- ➢ Debe poder seleccionar ayudantes de asignaturas que tenga a su cargo.
  - ✓ Esta funcionalidad seguirá en pie en esta nueva iteración del sistema. Además, se deberá poder **des-seleccionar** ayudantes mientras siga el periodo de selección.
- ➢ Debe poder registrar criterios de selección.
  - X Esta funcionalidad no se reconoce como algo necesario en esta nueva iteración del sistema. En cambio, el sistema será quien provea de la información requerida por los profesores para realizar la selección de ayudantes de manera oportuna.

2.2.4 Requerimientos del Jefe de Carrera

Este tipo de usuario forma parte de los actores definidos como **Administrador Departamento** en [2.1]

- ➢ Debe poder obtener la nómina de los ayudantes que estén participando en un proceso del sistema.
  - ✓ Esta funcionalidad seguirá en pie en esta nueva iteración del sistema.
- ➢ Debe poder ingresar o editar coordinadores docentes para cada una de las asignaturas, cargando la nómina de estos para cada asignatura, permitiendo filtrar y ordenar según criterios de selección.
  - X Esta funcionalidad no se mantendrá en la nueva iteración del sistema debido a que no se reconoce el coordinador docente como un actor, y no se podrán registrar explícitamente criterios de selección
- ➢ Debe poder validar o rechazar contratos de los alumnos seleccionados, permitiendo ver la información obtenida a través del sistema.
  - X Esta funcionalidad no se mantendrá en la nueva iteración del sistema debido a que, como se explica en [1.1] usando la figura [1.8.1], la contratación no es responsabilidad del sistema.
- ➢ Debe poder ver las asignaturas que no tengan postulantes, o que le falten ayudantes seleccionados.
  - ✓ Esta funcionalidad seguirá en pie en esta nueva iteración del sistema.

2.2.5 Requerimientos del Administrador

- ➢ Debe poder obtener la nómina de los ayudantes que estén participando en un proceso del sistema.
  - ✓ Esta funcionalidad seguirá en pie en esta nueva iteración del sistema.
- ➢ Debe poder modificar los estados del proceso de su departamento.
  - ✓ Esta funcionalidad seguirá en pie en esta nueva iteración del sistema.
- ➢ Debe poder obtener la nómina de los usuarios registrados y modificar cualquier característica de sus perfiles.
  - ✓ Esta funcionalidad seguirá en pie en esta nueva iteración del sistema.
- ➢ Debe poder crear/editar noticias o avisos del sistema.
  - ✓ Esta funcionalidad seguirá en pie en esta nueva iteración del sistema.
- ➢ Debe poder obtener la nómina de asignaturas que no tengan postulantes, o que le falten ayudantes seleccionados.
  - ✓ Esta funcionalidad seguirá en pie en esta nueva iteración del sistema.
- ➢ Debe poder realizar la carga masiva de asignaturas que se dictarán en el semestre en curso.
  - ✓ Esta funcionalidad seguirá en pie en esta nueva iteración del sistema.
- ➢ Debe poder entregar la posibilidad de agregar nuevos departamentos o campus al sistema y mantener los existentes.
  - X Este requerimiento tiene más que ver con el diseño del sistema.
- ➢ Debe poder modificar el listado de asignaturas perteneciente a cada departamento.
  - ✓ Esta funcionalidad seguirá en pie en esta nueva iteración del sistema.
- ➢ Debe poder mantener los contratos y las fechas iniciales y de término de éstos.
  - X Esta funcionalidad no se mantendrá en la nueva iteración del sistema debido a que, como se explica en [1.1] usando la figura [1.8.1], la contratación no es responsabilidad del sistema.

2.2.6 Otros Usuarios

Se mencionan más tipos de usuario en (Urso, A. 2012), pero no son reconocidos como actores en esta nueva iteración del sistema. Estos usuarios son el Coordinador Docente, y el Director de Departamento.

2.3 Atributos de Calidad del Sistema

Los atributos de calidad (también conocidos como requerimientos no funcionales), son criterios que miden qué tan bien funciona el sistema en general. La documentación original del SPA menciona algunos de estos criterios y su validación (Urso, A., 2012, p10-12). En esta iteración, se buscará que el sistema tenga una alta usabilidad, sea *robusto*, *seguro*, *mantenible*, *escalable* y *testeable*.

La **Usabilidad** se refiere a qué tan sencillo y agradable es su uso cotidiano. Tiene que ver con el diseño de la interfaz del usuario, los tiempos de respuesta, si se puede usar bien en dispositivos móviles, entre otros.

Que el sistema sea **Robusto** se refiere a que pueda lidiar con casos de errores sin mayor problema. Si bien también se busca que el sistema no tenga errores cotidianamente, es importante que un error no tenga repercusiones graves.

Un sistema **Seguro** resguarda y protege la información contenida dentro de sí mismo, previniendo que se filtre o sea manipulada indebidamente. Esto no se limita simplemente a ocultar el acceso a partes del sistema que usuarios administrativos o de alto nivel pueden utilizar, sino a controlar y limitarlo. Un sistema de autorización, roles y/o permisos es instrumental a la hora de asegurar que un sistema es seguro. Utilizar versiones actualizadas de los lenguajes de programación que intervengan en el sistema también es una necesidad, pues versiones obsoletas pueden contar con varias fallas de seguridad.

La **Mantenibilidad**, o qué tan mantenible es el sistema, es un atributo de suma importancia. Si el sistema es difícil de mantener (ya sea por una falta de documentación, malas prácticas, tecnologías en desuso u otro factor), su mantenimiento se vuelve cada vez más difícil y costoso hasta que simplemente deja de ser posible.

Un sistema sea **Testeable** implica que se pueden correr tests que verifiquen empíricamente que los distintos componentes del sistema funcionan bien. Estos tests pueden ser abstractos e imitar casos de uso completos o ser más concretos y de bajo nivel, verificando que ciertos métodos o funciones tengan el comportamiento esperado.

2.4 Control de Acceso

Es necesaria alguna manera de controlar las acciones de los usuarios del sistema para evitar que entren a páginas que no deberían acceder, o realicen operaciones para las cuales no están autorizados. Existen diferentes maneras de implementar esta idea, cada una con sus pros y sus contras.

2.4.1 Autorización Basada en Roles

Este enfoque, ejemplificado en la figura [2.3.1.1], consiste en que un usuario puede tener uno o más roles, los cuales serán examinados cada vez que el usuario quiera realizar una acción o acceder a una parte del sistema.

- ✓ Pro:
  - ➢ Muy simple de implementar.
  - ➢ Un usuario puede tener múltiples roles.
- × Contras:
  - ➢ Todos los permisos son implícitos.
  - ➢ Cada vez que se cree un rol nuevo hay que incluirlo en la lógica del sistema en todos los puntos donde se examine. (No es una solución escalable).

Figura 2.3.1.1: Modelo de Datos para una autorización básica basada en roles. Fuente: Elaboración propia.

2.4.2 Autorización Basada en Permisos

Este enfoque, ejemplificado en la figura [2.3.2.1], consiste en que un usuario puede tener muchos permisos, los cuales serán examinados cada vez que el usuario quiera realizar una acción o acceder a una parte del sistema. Cada permiso debería ser pertinente a una acción específica.

- ✓ Pro:
  - ➢ Simple de implementar.
  - ➢ Permite flexibilidad absoluta. Cada usuario puede tener permisos distintos según lo que necesite.
  - ➢ Todos los permisos son explícitos.
  - ➢ No hay que cambiar la lógica del sistema cada vez que se cree un permiso nuevo.
- × Contras:
  - ➢ Es difícil diferenciar entre un usuario y otro.
  - ➢ La cantidad de asociaciones puede ser muy elevada. Cada vez que se cree un usuario nuevo hay que asociarlo a posiblemente cientos de permisos distintos (no es escalable).

Figura 2.3.2.1: Modelo de Datos para una autorización básica basada en permisos. Fuente: Elaboración propia.

2.4.3 Autorización Basada en Roles y Permisos

Este enfoque, ejemplificado en la figura [2.3.3.1], consiste en combinan los 2 enfoques anteriores. Un usuario puede tener uno o más roles, los cuales a su vez tienen varios permisos asociados, los cuales serán examinados cada vez que el usuario quiera realizar una acción o acceder a una parte del sistema.

- ✓ Pro:
  - ➢ Un usuario puede tener múltiples roles.
  - ➢ Todos los permisos son explícitos.
  - ➢ No hay que cambiar la lógica del sistema cada vez que se cree un rol nuevo (escalable).
  - ➢ No hay que asociar a un usuario a cientos de permisos distintos (escalable).
- ✕ Contras:
  - ➢ No permite flexibilidad absoluta.

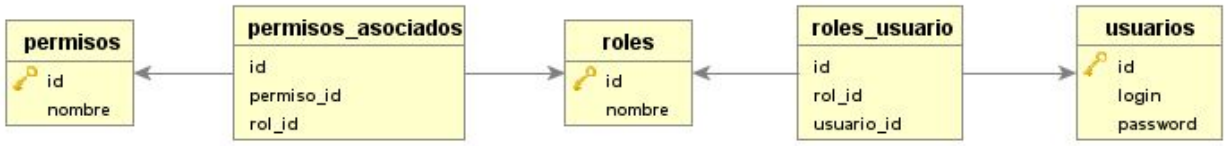


Figura 2.3.3.1: Modelo de Datos para una autorización básica basada en roles y permisos. Fuente: Elaboración propia.

2.4.4 Roles y Permisos usando Relaciones Polimórficas

Este enfoque, ejemplificado en la figura [2.3.4.1], consiste en que un usuario puede tener uno o más roles, los cuales a su vez tienen varios permisos asociados. También puede tener permisos directamente asociados, lo cual permite casos especiales.

- ✓ Pro:
  - ➢ Un usuario puede tener múltiples roles.
  - ➢ Todos los permisos son explícitos.
  - ➢ No hay que cambiar la lógica del sistema cada vez que se cree un rol nuevo (escalable).
  - ➢ No hay que asociar a un usuario a cientos de permisos distintos (escalable).
  - ➢ Permite flexibilidad absoluta.
- ✕ Contras:
  - ➢ Complejo de implementar.

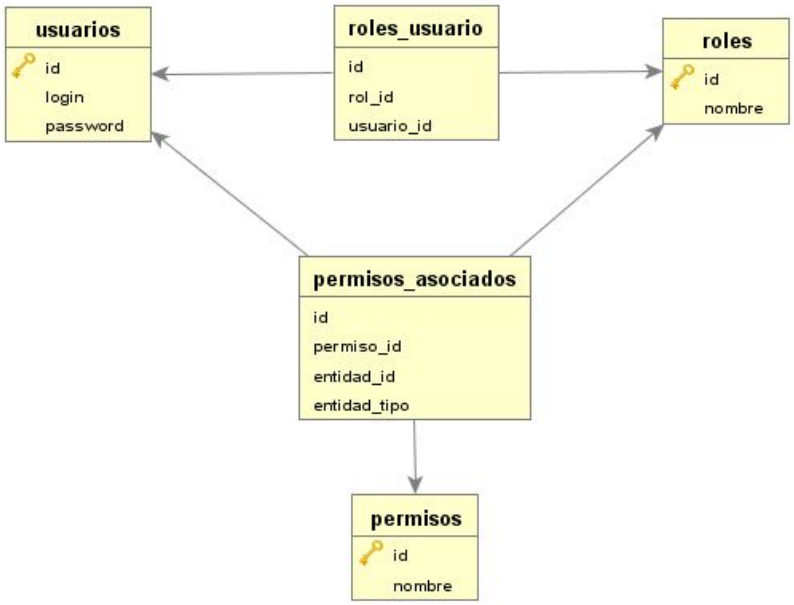


Figura 2.3.4.1: Modelo de Datos para una autorización básica basada en roles y permisos usando relaciones polimórficas.
Fuente: Elaboración propia.

La opción que se implementará en el nuevo sistema será la autorización basada en roles y permisos usando relaciones polimórficas. Si bien es una opción compleja de implementar, la versatilidad que provee será clave para facilitar el uso del nuevo sistema y posiblemente extenderlo a más departamentos a futuro.

2.5 *Framework*

Habiendo considerado diversos frameworks MVC tales como Ruby on Rails, Laravel y Django, se elige Laravel para desarrollar la nueva versión del SPA.

Laravel es un framework moderno de PHP que nació en el 2011 como una alternativa a CodeIgniter, otro framework del mismo lenguaje. Desde entonces, Laravel ha gozado de una creciente popularidad y una comunidad que ha impulsado su rápido desarrollo a lo largo de los años.[2]

Laravel implementa diversos patrones de diseño tales como Active Record y Modelo-Vista-Controlador, fomentando el desarrollo de aplicaciones web modernas y dinámicas, así como buenas prácticas de PHP, estándares y principios de programación orientada a objetos, entre otros.

2.5.1 Modelo-Vista-Controlador

Modelo-Vista-Controlador, más conocido como MVC es un patrón de diseño (Deacon, J. 1995). En el ámbito del desarrollo web, su principal ventaja es desacoplar las Vistas (lo que ve el usuario) de los Controladores y

[2] https://laravel.com/

los Modelos (la lógica de negocios del sistema), permitiendo así diseñar interfaces de usuario de manera ágil y rápida.

Otra de sus ventajas es la posibilidad de crear Vistas compuestas, desacoplando más aún los distintos componentes de cada interfaz.

La interfaz con la cual el Usuario interactúa es la Vista, la cual por detrás hace llamadas a algún Controlador, que se encarga de realizar búsquedas o cambios dentro de la base de datos a través de los Modelos para luego volver a actualizar la Vista, tal como lo muestra la figura 3.1.1.1.

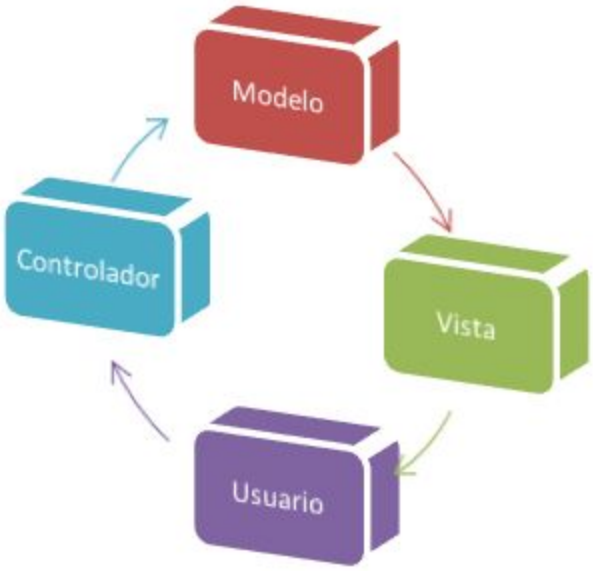


Figura 2.5.1.1: Interacción de un Usuario con un sistema que implementa el patrón MVC..
Fuente: Elaboración propia.

2.5.2 Active Record

En el patrón de diseño Active Record, a cada tabla de la base de datos le corresponde un Modelo. El modelo es una clase que se usa para interactuar con la base de datos, en operaciones tanto de lectura como de escritura. Otorga una capa adicional de protección para la base de datos y más flexibilidad a la hora de manipular los datos y sus relaciones.

Laravel implementa este patrón con Eloquent, su ORM[3]. Este ORM permite representar los datos de una manera intuitiva y relacional sin usar consultas Join explícitamente. Por ejemplo, en la figura [3.1.2.1] se muestra una asociación bastante simple (una persona tiene muchas mascotas).

Usando Join, se puede notar una duplicidad de datos mientras que usando Eloquent, la representación de estos no duplica información.

[3] https://laravel.com/docs/6.x/eloquent

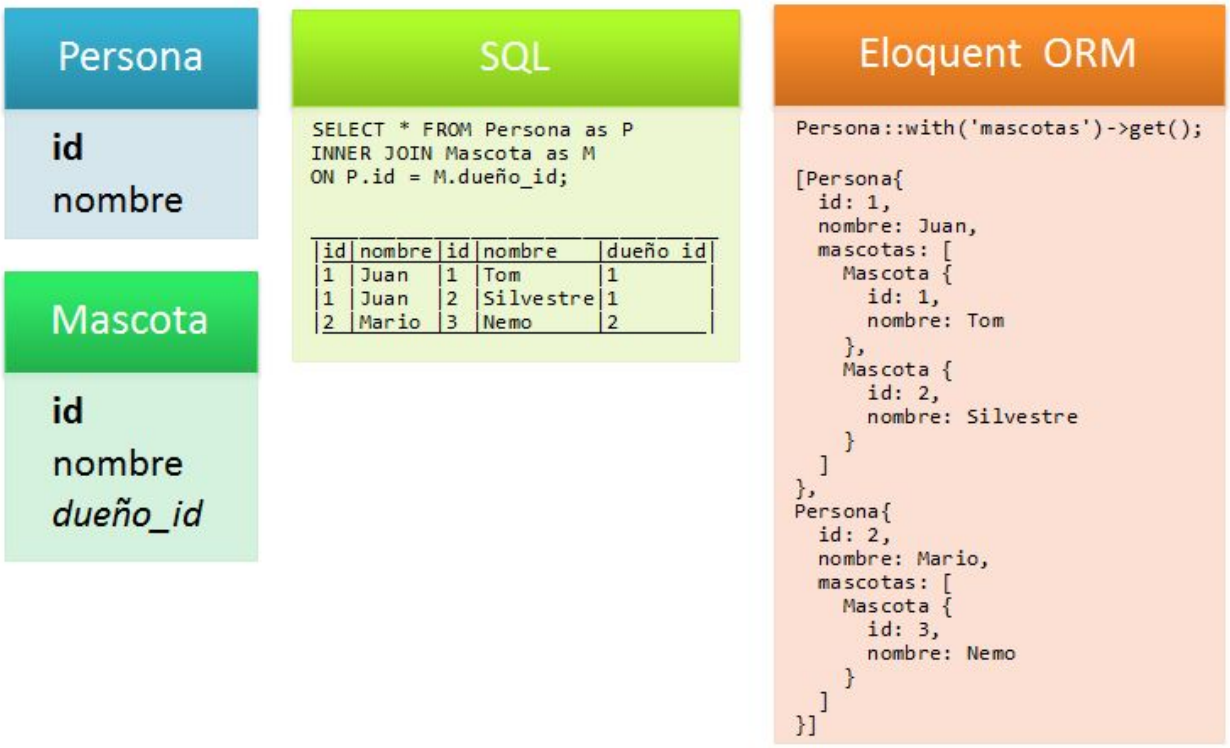


Figura 2.5.2.1: Comparación entre resultados de consulta SQL vs Eloquent ORM
Fuente: Elaboración propia.

2.5.3 Observer

El patrón de diseño Observer consiste en ciertas clases, llamadas observables, que están constantemente avisándoles a otras clases llamadas observadores acerca de sus cambios de estado. Es un patrón muy útil cuando un sistema tiene varios eventos importantes que deben manejarse.

Laravel permite la implementación de este patrón de 2 maneras distintas (mutuamente compatibles). La primera consiste en definir eventos manualmente, usando la API de desarrollo provista por el framework, y la otra es usar eventos ya definidos por Eloquent, su ORM.

2.5.4 PSR

PSR, del acrónimo **P**HP **S**tandards **R**ecommendations (Recomendaciones para estándares de PHP), son distintos estándares adoptados por la comunidad de desarrolladores web que utilizan PHP de manera cotidiana[4]. Para este nuevo sistema, se tomará en cuenta PSR/12, un estándar de estilo para redactar y formatear código.[5]

[4] https://www.php-fig.org/psr/
[5] https://www.php-fig.org/psr/psr-12/

2.5.5 SOLID

SOLID es un acrónimo que compacta 5 principios de diseño de programación orientada a objetos (OOP) que sirven de guía para lograr el desarrollo de un software mantenible, extensible y robusto. Laravel emplea estos principios en diversas partes de su arquitectura. A continuación, se describe cada uno de estos principios.

2.5.5.1 *Single Responsability Principle*

El principio de la responsabilidad única (SRP), dicta que una clase sólo debería tener una responsabilidad, definida como "un motivo para cambiar". Se refiere a que cada clase debería encargarse de una parte específica del sistema.

2.5.5.2. *Open-Closed Principle*

Este principio es muy sencillo pero a la vez muy importante. Se trata de pensar que los distintos módulos de un proyecto deberían ser diseñados pensando en que se puedan extender sin tener que modificarse ("abierto a extensión, cerrado a modificación") (Martin, R., 2000). Este principio puede sonar contradictorio, pero usando herencia y abstracciones tiene bastante lógica en un contexto orientado a objetos.

2.5.5.3 *Liskov Substitution Principle*

El principio de sustitución de Liskov simplemente postula que si una clase A es un subtipo de una clase B, entonces para cualquier función que utilice una clase B, debería poder reemplazarse por una clase A sin romper la funcionalidad (Martin, R., 2000). Es decir, cada subclase puede sustituir a su clase base.

2.5.5.4 *Interface Segregation Principle*

Este principio dice que ningún cliente debería ser forzado a depender de funcionalidades que no necesita. En términos concretos, llama a separar las funcionalidades de un servicio en varias interfaces pequeñas que podrán o no ser implementadas dependiendo de si la situación lo requiere como lo muestran las figuras [3.1.5.1] y [3.1.5.2].

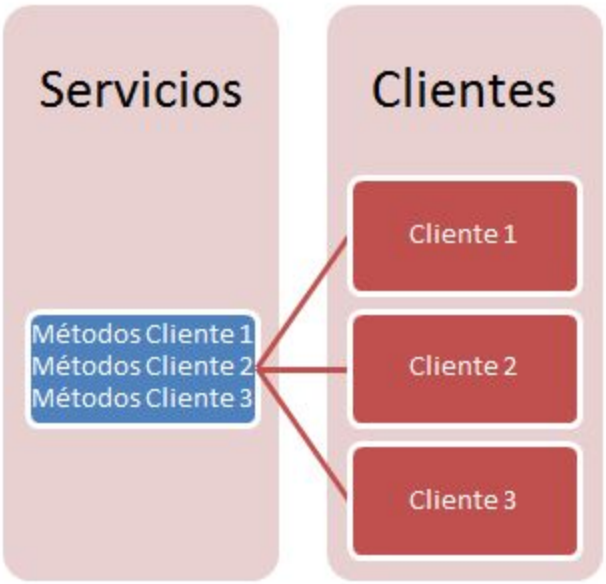


Figura 2.5.5.4.1: Cada cliente utiliza un servicio que implementa métodos de sobra que no van a utilizar. Fuente: Elaboración propia.

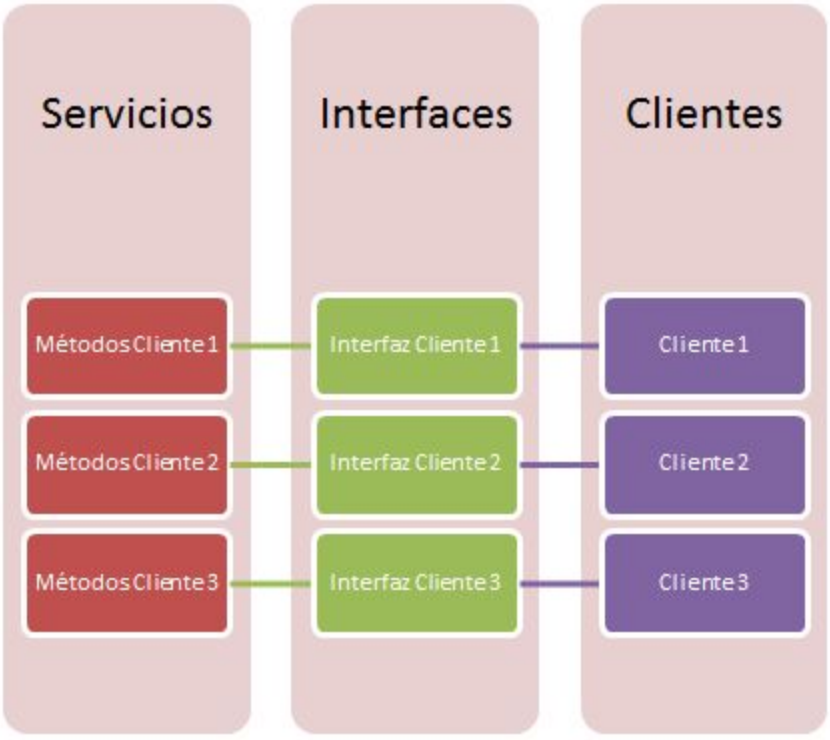


Figura 2.5.5.4.2: Aplicando el principio de segregación por interfaz, cada cliente utiliza un servicio más liviano que implementa solo lo que necesita. Fuente: Elaboración propia.

2.5.5.5 *Dependency Inversion Principle*

Este principio no es una regla inviolable, sino un ideal a seguir. En términos concisos: Se busca que cada dependencia en el diseño del sistema sea hacia una interfaz o clase abstracta, en vez de algo más concreto pues lo abstracto tiende a cambiar menos.

2.5.6 *Testing*

Realizar el testing del sistema, permite validar que cumpla los requerimientos funcionales y no funcionales, así como verificar que sea en efecto lo que el usuario necesita. Permite tener una métrica objetiva sobre qué tanto se ajusta a las especificaciones y requerimientos que debe cumplir.

También sirve para encontrar defectos y errores, pero no es esa su única función. Tener una buena cantidad de tests que aseguren una cobertura de la mayoría del código ejecutado, contribuye a mejorar la calidad del producto final.

2.5.6.1 PHPUnit

PHPUnit es un *framework* de *testing* para PHP desarrollado por Sebastian Bergmann[6]. Provee una API extensa con la que se pueden realizar diversos tipos de tests, tanto simples como complejos. Laravel extiende esta API, mejorando la sintaxis y legibilidad de estos.

src/Email.php

```php
<?php
declare(strict_types=1);

final class Email
{
    private $email;

    private function __construct(string $email)
    {
        $this->ensureIsValidEmail($email);

        $this->email = $email;
    }

    public static function fromString(string $email): self
    {
        return new self($email);
    }

    public function __toString(): string
    {
        return $this->email;
    }

    private function ensureIsValidEmail(string $email): void
    {
        if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
            throw new InvalidArgumentException(
                sprintf(
                    '"%s" is not a valid email address',
                    $email
                )
            );
        }
    }
}
```

tests/EmailTest.php

```php
<?php
declare(strict_types=1);

use PHPUnit\Framework\TestCase;

final class EmailTest extends TestCase
{
    public function testCanBeCreatedFromValidEmailAddress(): void
    {
        $this->assertInstanceOf(
            Email::class,
            Email::fromString('user@example.com')
        );
    }

    public function testCannotBeCreatedFromInvalidEmailAddress(): void
    {
        $this->expectException(InvalidArgumentException::class);

        Email::fromString('invalid');
    }

    public function testCanBeUsedAsString(): void
    {
        $this->assertEquals(
            'user@example.com',
            Email::fromString('user@example.com')
        );
    }
}
```

Figura 2.5.6.1.1:Ejemplo de clase PHP y su respectivo test usando PHPUnit. Fuente: [25]

[6] https://phpunit.de/index.html

2.5.6.2 Tipos de *Tests*

En el contexto de desarrollo web, se puede diferenciar entre 3 tipos de tests:

- ***Test* Unitario**: tiene por objetivo validar que los métodos de una clase tengan el comportamiento esperado. Son de muy bajo nivel así que no sirven para validar el cumplimiento de los requerimientos funcionales del sistema.
- ***Test* de Endpoint:** tiene por objetivo validar que cada endpoint o ruta declarada del sistema tenga el funcionamiento esperado. Particularmente útil para testear una API y el correcto manejo e interacción con la base de datos del sistema. Si bien se pueden usar para validar el cumplimiento de los requerimientos funcionales, la interacción que tienen estos tests con el front-end o con componentes dinámicos que utilicen Javascript es prácticamente nula.
- ***Test* de Browser**: consisten en simular un explorador y validar que el sistema se ajuste a cada caso de uso definido en sus requerimientos. Son perfectos para validar el diseño del sistema y que funcione exactamente como se estipula, pero contrario a los tests de Endpoint, son muy frágiles. Al depender del front-end, un ligero cambio de diseño puede significar que se deban reescribir varios tests. Laravel tiene una extensión oficial llamada Dusk que extiende PHPUnit y provee una API sencilla para programar esta clase de tests. Tienen el inconveniente de ser lentos de ejecutar, lo cual desincentiva su uso.

2.6 Integración en el DI

Si bien el sistema no es exclusivo al Departamento de Informática, éste si estará hosteado en un servidor del DI y será integrado en otros servicios del mismo departamento con tal de facilitar la respuesta rápida de un equipo de desarrollo.

2.6.1 Mattermost

Mattermost[7] es una plataforma flexible de mensajería open-source diseñada para permitir la colaboración de equipos de desarrollo de software[8]. También permite automatizar mensajes gracias a su API y la capacidad de configurar webhooks de entrada y salida.

2.6.2 Jenkins

Jenkins es un servidor *open-source* CI/CD que permite automatizar todo tipo de tareas relacionadas con testear y desplegar proyectos[9]. Puede ser instalado a través de paquetes nativos a un sistema UNIX, Docker o bien ser usado desde cualquier máquina con un entorno que tenga el Java Runtime Environment (JRE) instalado.

[7] https://mattermost.inf.utfsm.cl/
[8] https://mattermost.com/
[9] https://jenkins.io/

El DI tiene un servidor Jenkins disponible para uso interno que se está utilizando para automatizar el despliegue del SPA.

2.6.3 Stash

Stash, ahora conocido como Bitbucket Server, es una plataforma colaborativa diseñada para equipos modernos de desarrollo de software[10]. Permite mantener un control de versiones sobre varios de los proyectos del DI, entre los cuales está la intranet del departamento y esta nueva iteración del SPA[11].

2.6.4 Confluence

Otro producto de Atlassian, Confluence es una plataforma que permite planear y documentar proyectos en un espacio virtual colaborativo[12]. Al igual que Stash, se está usando en varios proyectos del DI[13], incluyendo el sistema SPA actual, la intranet del departamento y esta nueva iteración del sistema.

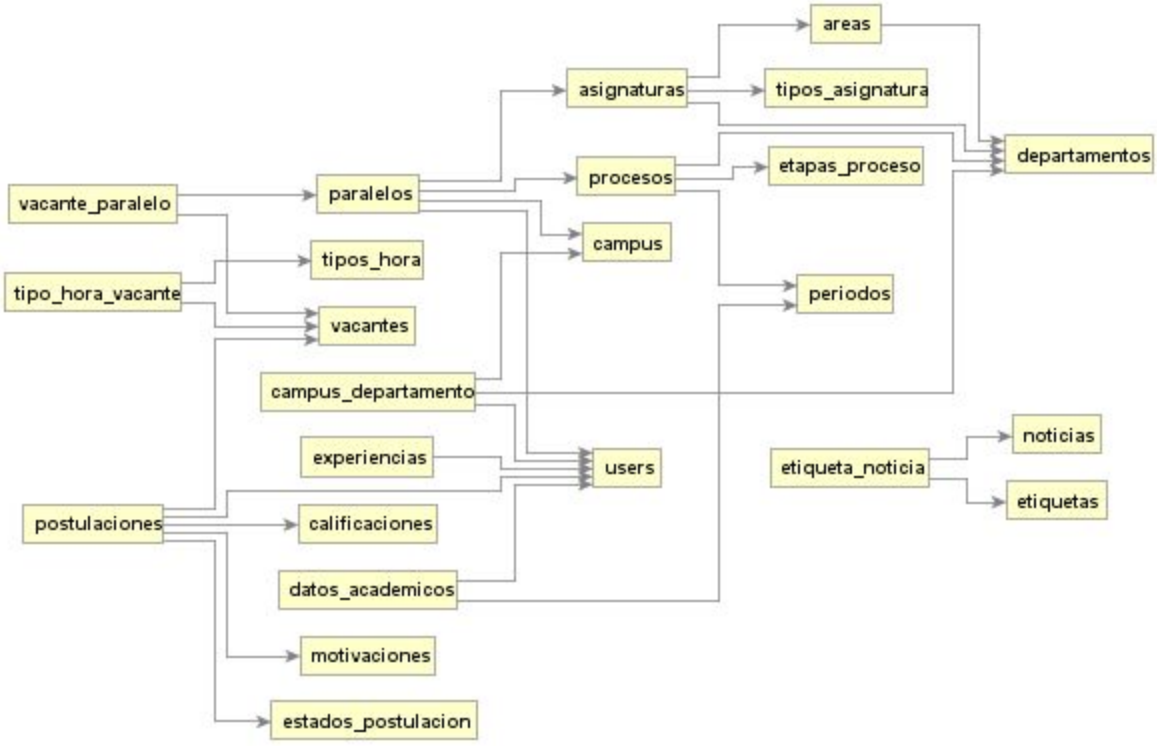
[10] https://www.atlassian.com/software/bitbucket/enterprise/data-center
[11] https://stash.inf.utfsm.cl/
[12] https://www.atlassian.com/software/confluence
[13] https://confluence.inf.utfsm.cl/

Capítulo 3

IMPLEMENTACIÓN

3.1 Modelo de Datos

A continuación, se puede ver el modelo de datos completo del sistema nuevo en la figura 3.1.1



:

Figura 3.1.1: Modelo de datos propuesto. Fuente: Elaboración propia usando DbVisualizer

3.2 Estructura de Proyecto

A continuación, se puede ver la estructura del directorio raíz del proyecto en la figura 3.2.1, y una descripción más detallada acerca de la función de cada carpeta en la tabla 3.2.2.

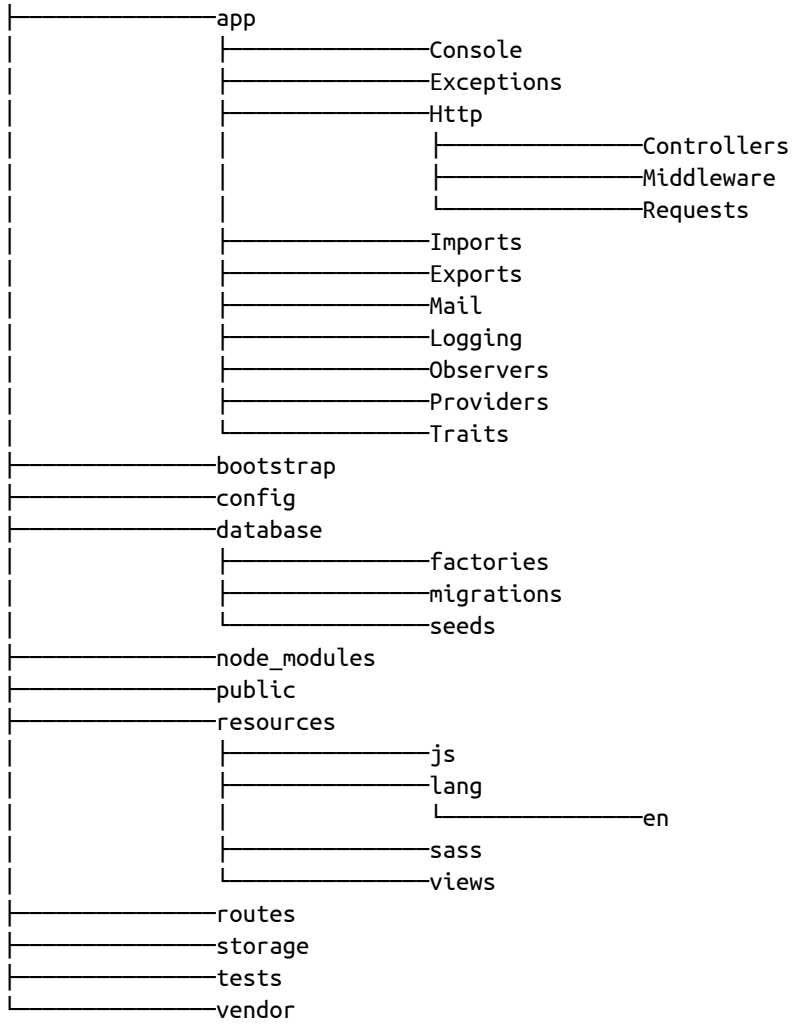


Figura 3.2.1: Estructura del directorio raíz del proyecto. Fuente: Elaboración propia.

DIRECTORIO	**FUNCIÓN**
`app/`	Concentra todo lo referente al backend y la lógica del sistema.
`app/Console/`	Contiene comandos personalizados de Artisan
`app/Exceptions/`	Contiene la clase que se preocupa de manejar los errores creados por Exceptions, así como Exceptions personalizadas creadas solo para el sistema.

DIRECTORIO	FUNCIÓN
app/Http/Controllers/	Contiene todos los Controladores del sistema.
app/Http/Middleware/	Contiene varios middlewares que se usan en el sistema, tales como los de autenticación.
app/Http/Requests/	Contiene todas las clases de tipo FormRequest, las cuales se encargan de validar toda petición antes de ejecutar código del Controlador.
app/Imports/	Contiene todas las clases que manejan la importación de datos según las especificaciones definidas en la documentación del paquete Laravel-Excel.
app/Exports/	Contiene todas las clases que manejan la exportación de datos según las especificaciones definidas en la documentación del paquete Laravel-Excel.
app/Mail/	Contiene clases que se dedican únicamente a construir mensajes a partir de vistas y enviarlos por email.
app/Logging/	Contiene clases relevantes para loguear información. En este caso, una implementación que permite enviar notificaciones a Mattermost en caso de error.
app/Observers/	Contiene clases observadoras a la espera de que un evento gatille su funcionamiento.
app/Providers/	Contiene clases que proveen servicios con respecto a diversas componentes del sistema (Rutas, Autorización, Roles y Permisos, etc).
app/Traits/	Contiene algunas funciones reusables.
bootstrap/	Contiene scripts que se corren cada vez que el sistema se inicia.
config/	Contiene distintos archivos de configuración para distintas herramientas. Son todos archivos públicos por lo cual no se deberían guardar claves en ellos. Si se necesitan claves, se pueden obtener de los archivos de entorno sin riesgo.
database/	Contiene todo lo referente a la base de datos: Migraciones, clases para poblar la base de datos con datos de prueba (Seeders) y Factories.
node_modules/	Contiene todas las dependencias de Javascript instaladas a través de Npm o Yarn.
public/	La carpeta pública del proyecto. Aquí van los archivos de Javascript, estilos e imágenes después de ser minificados.
resources/	Contiene todas las plantillas Blade, archivos de Javascript y estilos sin minificar, imágenes y archivos de traducción.
routes/	Contiene los archivos que definen los 4 tipos de endpoints del sistema. Endpoints para acceso por web, para acceso por API, endpoints de broadcasting, y endpoints de consola.
storage/	Contiene archivos de caché, sesión, compilado de vistas Blade y otros.
tests/	Contiene todos los archivos y clases referentes a testing.
vendor/	Contiene todas las dependencias de PHP instaladas a través de Composer.

Tabla 3.2.2: Descripción de carpetas de interés en proyecto Laravel. Fuente: Elaboración propia.

3.2.1 Archivos de Entorno

A continuación, se describen algunos de los archivos de entorno de mayor importancia para el nuevo sistema. Están ubicados en el directorio raíz del proyecto.

- **composer.json**: Es un archivo en formato JSON que dicta las dependencias de *PHP* primarias que tiene el proyecto. Es usado por **Composer** para gestionar las dependencias de *PHP* secundarias del proyecto.
- **composer.lock**: Es un archivo en formato JSON generado por **Composer** a partir del archivo anterior. Dicta que dependencias de *PHP* deben ser instaladas, así como sus versiones.
- **.env**: Es un archivo de texto plano que se usa para guardar variables de entorno. Como su contenido tiene contraseñas, es un archivo que no debe ser agregado al repositorio.
- **.env.dusk.testing**: Similar a .env, guarda variables de entorno. Sin embargo, se usa para el entorno de desarrollo de Laravel Dusk, por lo cual no debería haber datos sensibles o contraseñas de valor aquí.
- **.env.example**: Este archivo, tal como lo dice su nombre, es simplemente un .env con valores por defecto. Se debe agregar al repositorio para que otros desarrolladores conozcan la lista de variables que deben declarar para usar el sistema sin problemas.
- **package.json**: Es un archivo en formato JSON que dicta las dependencias de *javascript* primarias que tiene el proyecto. Es usado por **NPM** para gestionar las dependencias de *javascript* secundarias del proyecto.
- **package.lock**: Es un archivo en formato JSON generado por **NPM** a partir del archivo anterior. Dicta que dependencias de *javascript* deben ser instaladas, así como sus versiones.
- **phpunit.xml**: Este archivo se ocupa a la hora de correr tests con PHPUnit. Similar a los .env, guarda variables de entorno, pero en formato XML en vez de texto plano.
- **phpunit.dusk.xml**: Este archivo es lo mismo que phpunit.xml, pero aplicado a Laravel Dusk. Debido a las restricciones de correr pruebas de integración, es mejor tener estos dos archivos por separado.
- **Jenkinsfile**: Este archivo contiene instrucciones para el servidor de Jenkins.

3.3 Dependencias

3.3.1 Composer

Composer es un administrador de dependencias para PHP[14] [15]. Permite declarar cuáles librerías/paquetes se van a usar en el proyecto en un archivo (composer.json). Composer instala y actualiza las dependencias localmente para cada proyecto, evitando problemas de compatibilidad entre distintas versiones instaladas en distintos proyectos. A continuación se listan algunas de las dependencias más usadas en el proyecto.

3.3.1.1 Artisan

Artisan es la interfaz de comandos de Laravel (CLI). Para listar todos los comandos que posee, hay que ejecutar `php artisan list`. Todos los comandos tienen una ayuda. Por ejemplo, para saber cómo usar el comando `migrate`, se puede ejecutar `php artisan help migrate`.

[14] https://getcomposer.org/
[15] https://packagist.org/

3.3.1.2 PHPunit

Ya se explicó en [2.5.6.1] lo que es PHPUnit. Una vez incorporado al proyecto de Laravel como dependencia a través de Composer, se pueden ejecutar los tests con el comando `./vendor/bin/phpunit`.

3.3.1.3 Tinker

Tinker[16] es el REPL[17] de Laravel. Permite probar código directamente en la consola facilitando la exploración y el debugueo, lo cual facilita un aprendizaje rápido de las distintas funciones del framework.

Para usar Tinker, se debe ejecutar el comando `php artisan tinker`.

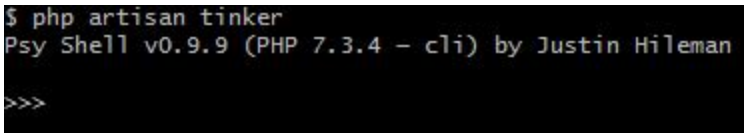


Figura 3.3.1.3.1. REPL Tinker.

3.3.1.4 Laravel-Excel

Laravel-Excel[18] es una biblioteca basada en PHP-Spreadsheets[19]; otra biblioteca que permite la creación y manipulación de hojas de cálculo. Laravel-Excel define una arquitectura basada en clases de exportación e importación, simplificando la lectura y escritura de un código mantenible y extensible. Será usado en este sistema principalmente para exportar nóminas de ayudantes y realizar la carga masiva semestral.

[16] https://github.com/laravel/tinker
[17] Read-Eval-Print-Loop
[18] https://laravel-excel.com/
[19] https://phpspreadsheet.readthedocs.io/en/latest/

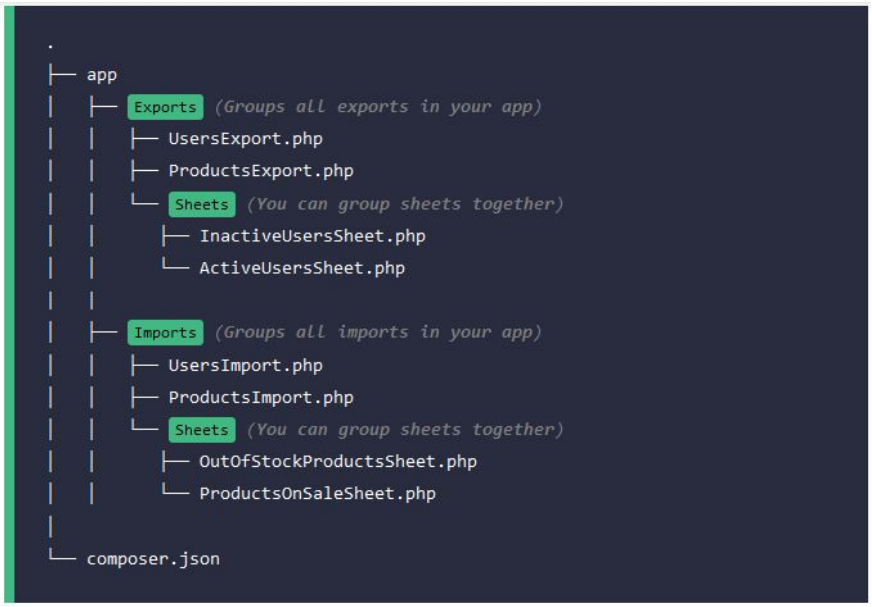


Figura 3.3.1.4.1: Estructura de directorio para objetos de exportación e importación. Fuente:[20]

3.3.1.5 Predis

Predis es un cliente flexible y completo de Redis para PHP. No es una dependencia con la que se vaya a interactuar mucho, pero es un requisito para poder utilizar caché con Redis.

3.3.1.6 Bouncer

Bouncer[21] es una biblioteca para manejar roles y permisos usando Eloquent. Como se expuso en la sección [2.4], el mejor enfoque para que el sistema sea versátil en cuanto a control de acceso es usar roles y permisos con relaciones polimórficas.

Bouncer utiliza este enfoque con las 4 tablas que se muestran en la figura 3.3.1.6.1. Además, agrega otra capa de versatilidad agregando la posibilidad de prohibir explícitamente un permiso. En el anexo B se detalla la sintaxis de Bouncer.

[20] https://docs.laravel-excel.com/3.1/architecture/objects.html
[21] https://github.com/JosephSilber/bouncer

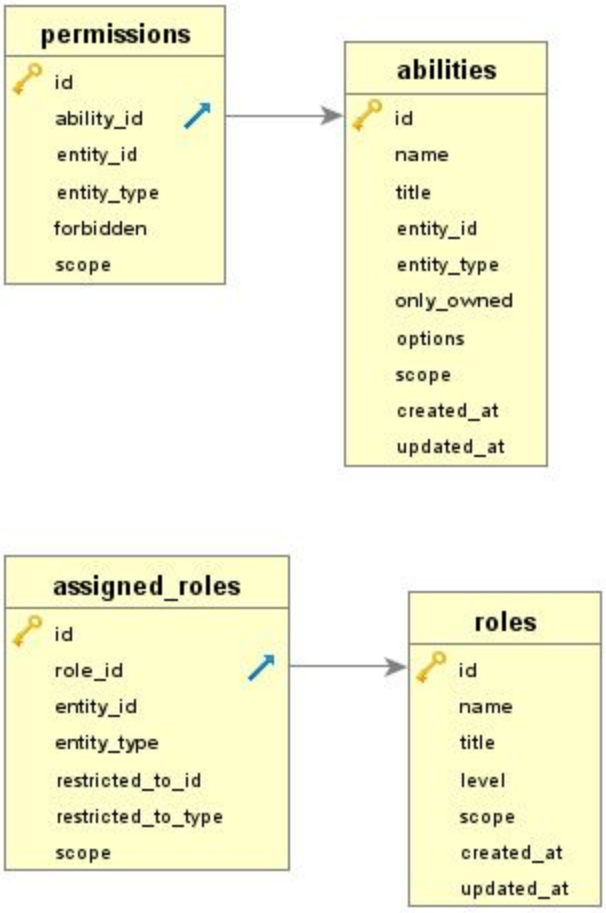


Figura 3.3.1.6.1: Tablas adicionales creadas por Bouncer. Fuente: Elaboración propia usando DbVisualizer

3.3.1.7 Mattermost PHP

Mattermost PHP[22] es un driver que permite utilizar los webhooks de mattermost para enviar mensajes usando la API de Mattermost. Se usa en el sistema para loguear errores directamente al chat the Mattermost.

3.3.1.8 Laravel Debugbar

Laravel Debugbar[23] es una dependencia que solo debe estar presente en un entorno de desarrollo (logrado usando la opción --require-dev de Composer) que entrega información que ayuda en el debug del sistema.

[22] https://github.com/ThibaudDauce/mattermost-php
[23] https://github.com/barryvdh/laravel-debugbar

3.3.1.9 Laravel Dusk

Dusk[24] es una dependencia que solo debe estar presente en un entorno de desarrollo. Provee una sintaxis sencilla y expresiva para automatizar testing the integración, simulando las acciones de un usuario en un explorador web. Los tests de Dusk se ejecutan con el comando `php artisan dusk`. Se puede especificar parámetros adicionales como si fuera PHPUnit.

3.3.1.10 Faker

Faker[25] es una dependencia para generar datos falsos, inspirada en otras librerías con el mismo nombre para PERL[26] y Ruby[27]. Se usa principalmente para hacer el testing del sistema y para poblar inicialmente la base de datos. Laravel usa Faker en sus generadores de modelos (Model Factories) como se puede apreciar en la figura 3.3.1.10.1.

```php
<?php

use Illuminate\Support\Str;
use Faker\Generator as Faker;

$factory->define(App\User::class, function (Faker $faker) {
    $faker = \Faker\Factory::create('es_AR');
    $genero = $faker->randomElement(['M', 'F']);
    $nombre = $genero == 'M' ? $faker->firstNameMale : $faker->firstNameFemale;
    return [
        'nombre' => $nombre,
        'apellido_paterno' => $faker->lastName,
        'apellido_materno' => $faker->lastName,
        'genero' => $genero,
        'telefono' => '+56 9 ' . $faker->numberBetween(7,9) . $faker->randomNumber(3, true) . ' ' . $faker->randomNumber(4, true),
        'email' => $faker->unique()->safeEmail,
        'email_verified_at' => now(),
        'password' => '$2y$10$TKh8H1.PfQx37YgCzwiKb.KjNyWgaHb9cbcoQgdIVFlYg7B77UdFm', // secret
        'remember_token' => Str::random(10),
        'api_token' => Str::random(60),
        'bio' => $faker->sentences(5, true),
        'rut' => $faker->unique()->numberBetween(18000000, 22000000) . '-' . $faker->randomElement(['1','2','3','4','5','6','7','8','9','0','K'])
    ];
});

$factory->afterCreatingState(App\User::class, 'ayudante', function ($user, $faker) {
    $user->assign('ayudante');
});

```

Figura 3.3.1.10.1: Ejemplo de Model Factory definida usando Faker. Fuente: Elaboración Propia.

3.3.2 NPM

Similar a Composer, NPM[28] [29] es un manejador de dependencias para Javascript. El sistema no será desarrollado usando un framework de Javascript así que las dependencias usadas no son muchas.

[24] https://laravel.com/docs/6.x/dusk
[25] https://github.com/fzaninotto/Faker
[26] https://metacpan.org/release/JASONK/Data-Faker-0.07
[27] https://rubygems.org/gems/faker
[28] https://www.npmjs.com/
[29] Node Package Manager

3.3.2.1 Laravel Mix

Laravel Mix[30] provee una abstracción sobre Webpack, permitiendo compilar y minificar archivos de estilos y Javascript, para luego moverlos desde la carpeta resource hacia la carpeta public.

webpack.mix.js

```
const mix = require('laravel-mix');

/*
 |--------------------------------------------------------------------------
 | Mix Asset Management
 |--------------------------------------------------------------------------
 |
 | Mix provides a clean, fluent API for defining some Webpack build steps
 | for your Laravel application. By default, we are compiling the Sass
 | file for the application as well as bundling up all the JS files.
 |
 */


mix.sass('resources/sass/material-dashboard.scss', 'public/css/material-dashboard.css')
   .sass('resources/sass/multi-select.scss', 'public/css/multi-select.css')
   .copy('resources/js/jquery.multi-select.js', 'public/js/jquery.multi-select.js')
   .minify('public/js/jquery.multi-select.js')
   .combine([
   'resources/js/core/jquery.min.js',
   'resources/js/core/popper.min.js',
   'resources/js/core/bootstrap-material-design.min.js'
   ], 'public/js/material-dashboard-core.js')
   .combine([
      'resources/js/plugins/perfect-scrollbar.jquery.min.js',
      'resources/js/plugins/chartist.min.js',
      'resources/js/plugins/bootstrap-notify.js'
      ], 'public/js/material-dashboard-plugins.js')
   .copy('resources/js/material-dashboard.js', 'public/js/material-dashboard-control-center.js')
   .minify('public/js/material-dashboard-control-center.js')
   .version();
```

Figura 3.3.2.1.1: Ejemplo de Laravel Mix. Fuente: Elaboración propia.

3.3.2.2 Material Dashboard

Material Dashboard[31] es un tema de bootstrap basado en la filosofía de diseño material de Google. Se usará en esta iteración del SPA para proveer interfaces vibrantes y agradables para el usuario.

[30] https://laravel.com/docs/6.x/mix
[31] https://github.com/creativetimofficial/material-dashboard

Capítulo 4

CONCLUSIONES

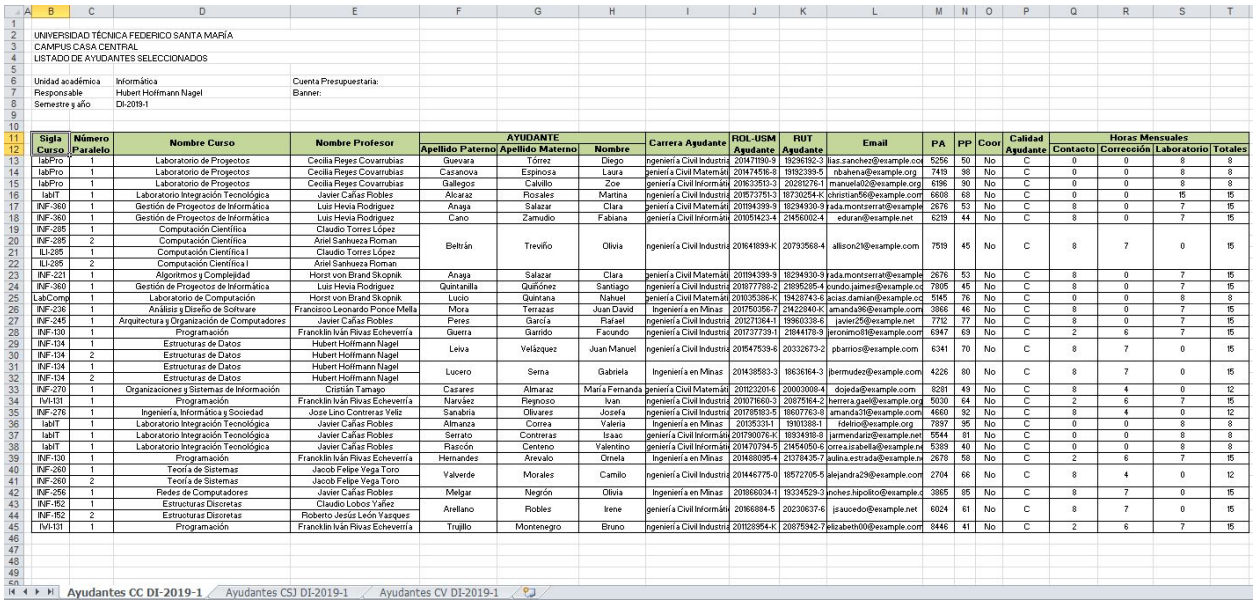
4.1 Cumplimiento de Objetivos

Dentro de los objetivos planteados en el primer capítulo, el más importante es el de mejorar la usabilidad del sistema existente, actualizándolo con herramientas modernas y buenas prácticas de desarrollo. Esto implica corregir los casos donde se detiene el proceso cada vez que la interfaz antigua no otorga la funcionalidad esperada por parte de sus usuarios. Respecto a este objetivo, los profesores ahora pueden deshacer la selección de ayudantes sin tener que recurrir al administrador del sistema y toda la funcionalidad que se espera de parte de los administradores está contenida dentro del nuevo diseño.

Con tal de mantener la simplicidad del sistema y ayudar a su futura mantención, se planteó que tuviera una alta mantenibilidad. Primero, con respecto a la sobreingeniería del modelo de datos: dado que se transformó un modelo de datos consistente de 49 tablas (de las cuales 20 se consideran inútiles) en un modelo de datos de 30 tablas, de las cuales 6 son referentes al framework (ver anexo C) o al sistema de permisos (ver sección 4.3.1.4) sin perder funcionalidades importantes, se puede concluir que este objetivo de corregir esta situación fue logrado en un 100%.

Respecto a la veracidad de datos académicos ingresados por los ayudantes, un objetivo de esta memoria es el de corregir esta situación pidiendo los datos directamente a la DTI. Sin embargo, debido a la política interna de la UTFSM y la DTI referente al uso y mantención de datos públicos y privados, no fue factible lograr esta meta. En cambio, se provee una solución parche que consiste en forzar a los ayudantes a actualizar sus datos periodo a periodo.

Respecto a las nóminas de datos mal formateados, se considera que este objetivo fue logrado completamente usando un diseño aprobado y validado por Secretaría y Subdirección de Pregrado para mostrar solo los datos pertinentes a las necesidades de cada quien, el cual puede apreciarse en la figura 5.1.1.



UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
CAMPUS CASA CENTRAL
LISTADO DE AYUDANTES SELECCIONADOS

Unidad académica: Informática — Cuenta Presupuestaria:
Responsable: Hubert Hoffmann Nagel — Banner:
Semestre y año: DI-2019-1

Sigla Curso	Número Paralelo	Nombre Curso	Nombre Profesor	AYUDANTE			Carrera Ayudante	ROL-USM Ayudante	RUT Ayudante	Email	PA	PP	Coor	Calidad Ayudante	Horas Mensuales			
				Apellido Paterno	Apellido Materno	Nombre									Contacto	Corrección	Laboratorio	Totales
labPro	1	Laboratorio de Proyectos	Cecilia Reyes Covarrubias	Guevara	Torrez	Diego	ngeniería Civil Industria	201471190-5	19296192-3	ilias.sanchez@example.co	5256	50	No	C	0	0	8	8
labPro	1	Laboratorio de Proyectos	Cecilia Reyes Covarrubias	Casanova	Espinoza	Laura	genieria Civil Matemáti	201474516-8	19192389-5	nbalvina@example.org	7459	98	No	C	0	0	8	8
labPro	1	Laboratorio de Proyectos	Cecilia Reyes Covarrubias	Gallegos	Calvillo	Zoe	genieria Civil Informátic	201633513-3	20281276-1	manuela02@example.org	6196	90	No	C	0	0	8	8
labIT	1	Laboratorio Integración Tecnológica	Javier Cañas Robles	Alcaraz	Rosales	Martina	ngeniería Civil Industria	201573751-3	18730254-K	christian56@example.com	6608	68	No	C	0	0	15	15
INF-360	1	Gestión de Progectos de Informática	Luis Hevia Rodriguez	Anaga	Salazar	Clara	genieria Civil Matemáti	201194399-9	18294930-9	rada.montserrat@example	2676	53	No	C	8	0	7	15
INF-360	1	Gestión de Progectos de Informática	Luis Hevia Rodriguez	Cano	Zamudio	Fabiana	genieria Civil Informátic	201051423-4	21456002-4	eduran@example.net	6219	44	No	C	8	0	7	15
INF-285	1	Computación Científica	Claudio Torres López	Beltrán	Treviño	Olivia	ngeniería Civil Industria	201641899-K	20793568-4	allison21@example.com	7519	45	No	C	8	7	0	15
INF-285	2	Computación Científica	Ariel Sanhueza Roman															
ILI-285	1	Computación Científica I	Claudio Torres López															
ILI-285	2	Computación Científica I	Ariel Sanhueza Roman															
INF-221	1	Algoritmos y Complejidad	Horst von Brand Skopnik	Anaga	Salazar	Clara	genieria Civil Matemáti	201194399-9	18294930-9	rada.montserrat@example	2676	53	No	C	8	0	7	15
INF-360	1	Gestión de Progectos de Informática	Luis Hevia Rodriguez	Quintanilla	Quiñónez	Santiago	ngeniería Civil Industria	201877788-2	21695285-4	cundo.jaimez@example.co	7805	45	No	C	8	0	7	15
LabComp	1	Laboratorio de Computación	Horst von Brand Skopnik	Lucio	Quintana	Nahuel	genieria Civil Matemáti	201035386-K	19425743-6	aciaz.damian@example.co	5145	76	No	C	0	0	8	8
INF-236	1	Análisis y Diseño de Software	Francisco Leonardo Ponce Mella	Mora	Terrazas	Juan David	Ingeniería en Minas	201750356-7	21422840-K	amanda96@example.com	3866	46	No	C	8	0	7	15
INF-245	1	Arquitectura y Organización de Computadores	Javier Cañas Robles	Peres	García	Rafael	ngeniería Civil Industria	201271364-1	19960338-6	javier25@example.net	7712	77	No	C	8	0	7	15
INF-130	1	Programación	Franoklin Iván Rivas Echeverría	Guerra	Garrido	Facundo	ngeniería Civil Industria	201737739-1	21844178-9	jeronimo81@example.com	6947	69	No	C	2	6	7	15
INF-124	1	Estructuras de Datos	Hubert Hoffmann Nagel	Leiva	Velázquez	Juan Manuel	ngeniería Civil Industria	201547539-6	20332673-2	pbarrios@example.com	6341	70	No	C	8	7	0	15
INF-124	2	Estructuras de Datos	Hubert Hoffmann Nagel															
INF-124	1	Estructuras de Datos	Hubert Hoffmann Nagel	Lucero	Serna	Gabriela	Ingeniería en Minas	201438583-3	18636164-3	jbermudez@example.com	4226	80	No	C	8	7	0	15
INF-124	2	Estructuras de Datos	Hubert Hoffmann Nagel															
INF-270	1	Organizaciones y Sistemas de Información	Cristián Tamago	Casares	Almaraz	María Fernanda	genieria Civil Matemáti	201223201-6	20003008-4	dojeda@example.com	8281	49	No	C	8	4	0	12
IVI-131	1	Programación	Franoklin Iván Rivas Echeverría	Narváez	Reynoso	Ivan	ngeniería Civil Industria	201071660-3	20875164-2	herrera.gael@example.org	5030	64	No	C	2	6	7	15
INF-276	1	Ingeniería, Informática y Sociedad	Jose Lino Contreras Veliz	Sanabria	Olivares	Josefa	ngeniería Civil Industria	201795183-5	18607763-8	amanda39@example.com	4660	92	No	C	8	4	0	12
labIT	1	Laboratorio Integración Tecnológica	Javier Cañas Robles	Almanza	Correa	Valeria	Ingeniería en Minas	201353331-1	19101388-1	fdelrio@example.org	7897	95	No	C	0	0	8	8
labIT	1	Laboratorio Integración Tecnológica	Javier Cañas Robles	Serrato	Contreras	Isaac	genieria Civil Informátic	201790076-K	18934918-8	jarmendariz@example.net	5544	81	No	C	0	0	8	8
labIT	1	Laboratorio Integración Tecnológica	Javier Cañas Robles	Rascón	Centeno	Valentino	genieria Civil Informátic	201470794-5	21454050-6	correa.isabella@example.ne	5389	40	No	C	0	0	8	8
INF-130	1	Programación	Franoklin Iván Rivas Echeverría	Hernandes	Arevalo	Ornela	Ingeniería en Minas	201488095-4	21376435-7	paulina.estrada@example.ne	2678	58	No	C	2	6	7	15
INF-260	1	Teoría de Sistemas	Jacob Felipe Vega Toro	Valverde	Morales	Camilo	ngeniería Civil Industria	201446775-0	18572705-5	alejandra29@example.com	2704	66	No	C	8	4	0	12
INF-260	2	Teoría de Sistemas	Jacob Felipe Vega Toro															
INF-256	1	Redes de Computadores	Javier Cañas Robles	Melgar	Negrón	Olivia	Ingeniería en Minas	201866034-1	19334529-3	inohes.hipolito@example.c	3865	85	No	C	8	7	0	15
INF-152	1	Estructuras Discretas	Claudio Lobos Yañez	Arellano	Robles	Irene	genieria Civil Informátic	201668984-5	20230637-6	jsaucedo@example.net	6024	61	No	C	8	7	0	15
INF-152	2	Estructuras Discretas	Roberto Jesús León Vasques															
IVI-131	1	Programación	Franoklin Iván Rivas Echeverría	Trujillo	Montenegro	Bruno	ngeniería Civil Industria	201128954-K	20875942-7	elizabeth00@example.com	8446	41	No	C	2	6	7	15

Ayudantes CC DI-2019-1 | Ayudantes CSJ DI-2019-1 | Ayudantes CV DI-2019-1

Figura 4.1.1: Nueva nómina de ayudantes seleccionados. Generado automáticamente por el SPA.
Los datos usados aquí no representan ayudantes reales. Fueron generados aleatoriamente usando Faker (ver 4.3.1.10).

4.2 Herramientas Usadas en el Desarrollo

Primero, para todas las herramientas usadas en el desarrollo, se averiguó su coste. Se buscó utilizar herramientas con licencias permisivas o libres, para no agregar un costo de mantención adicional después de entregar el sistema al DI.

En cuanto al lenguaje de programación, los candidatos fueron Ruby, Python y PHP. Se buscó utilizar un framework MVC de estos lenguajes para el desarrollo. En particular, estos frameworks serían Rails (Ruby), Django (Python) y Laravel (PHP). Estos no son los únicos frameworks MVC de los lenguajes candidatos, pero son de los más populares, lo cual implica que tienen una gran cantidad de herramientas y librerías disponibles.

La decisión de escoger un lenguaje por sobre otro termina siendo completamente subjetiva. Si bien se puede argumentar que Ruby podría ser más lento debido a que es un lenguaje interpretado, o que Python es el lenguaje con el crecimiento más veloz de los últimos años, el desempeño del sistema se verá más afectado por la calidad del código escrito en el desarrollo que por las sutiles diferencias entre un lenguaje y otro. Tomando esto en cuenta, se elige desarrollar usando Laravel después de comparar la documentación de todos los frameworks, así como los tutoriales disponibles.

Haber elegido Laravel como framework para desarrollar el sistema significó que se tuvo que elegir también un DBMS[32] compatible. Laravel viene con soporte para MySQL, PostgreSQL, SQLite y Microsoft SQL Server por defecto, los cuales están en el ranking de los 10 RDBMS[33] más populares[34]. Si bien existen extensiones para agregar compatibilidad con otros DBMS, se decidió escoger entre uno de los cuatro para evitar tener muchas dependencias

[32] Database Management System.
[33] Relational Database Management System.
[34] https://db-engines.com/en/ranking/relational+dbms

innecesarias. Microsoft SQL Server y MySQL fueron descartados como opciones debido a su licencia comercial. Tanto SQLite como PostgreSQL tienen una licencia Open-Source y fueron elegidos para el desarrollo del sistema.

Si bien SQLite es veloz debido a ser un DBMS muy liviano y sencillo, no provee soporte para todo tipo de datos y tiene problemas de desempeño con *queries* complejas. Sin embargo, necesita una configuración mínima, lo cual lo convirtió en el DBMS ideal para realizar el testing mientras que PostgreSQL es más robusto y tiene mayor desempeño así que se utilizó para la base de datos principal del sistema.

4.3 Impacto de la Solución en la Organización

Gracias a la nueva versión del SPA, se reduce considerablemente el tiempo y la carga administrativa de Secretaría y del Administrador, eliminando en lo posible los datos erróneos y facilitando su uso tanto para alumnos como para profesores. El proceso de carga masiva, que hasta entonces era realizado manualmente por el Administrador línea por línea, es facilitado enormemente al imponer un formato único de carga masiva con validación incluida, como se ve en la figura 5.2.1. El Administrador también era responsable de enviarle la nómina de ayudantes seleccionados a Secretaría, que tenía que transcribirla a otra hoja de cálculo debido a su formato (ver sección 1.2) pero ahora Secretaría posee un rol activo dentro del sistema que le permite descargar la nómina cuando la necesite.

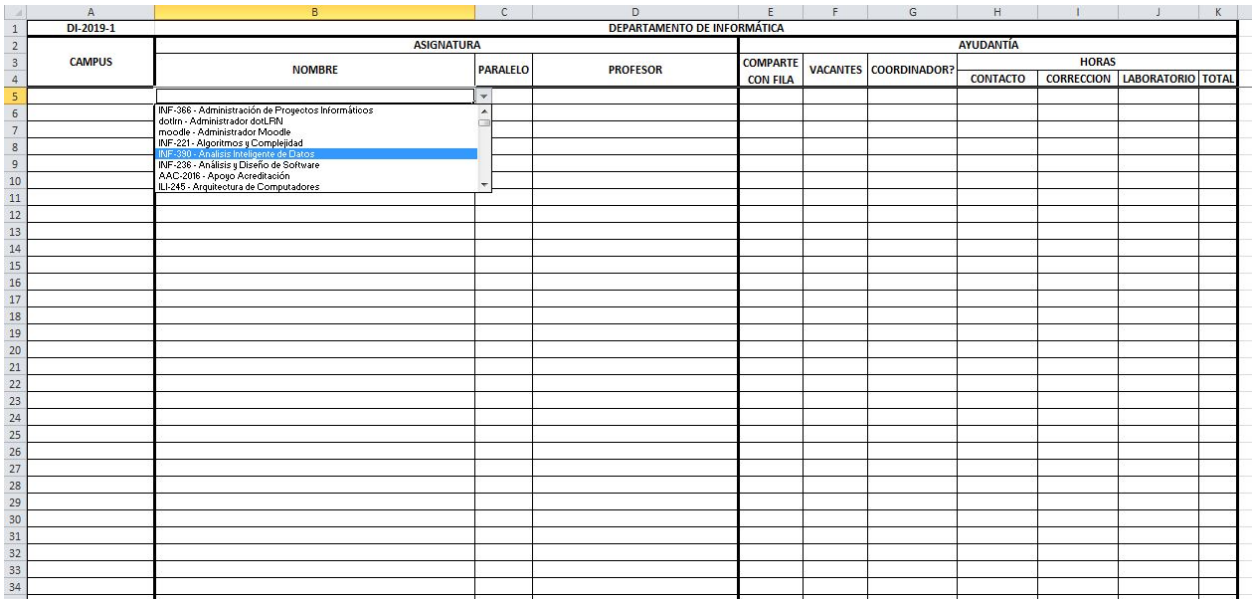


Figura 4.3.1: Nuevo formato unificado de Carga Masiva. Generado automáticamente por el SPA.

4.4 Proyecciones a Futuro

4.4.1 Integración con DEA[35]

Para optar a las becas de Ayudantías de la UTFSM, se requiere estar cursando, tener aprobado (o no tener reprobado) el curso virtual de la Escuela de Asistentes de Aprendizaje[36] (EAA). Esto es un requerimiento reciente. Actualmente se debe descargar una nómina desde el sitio del DEA e ir verificando si los ayudantes cumplen o no las condiciones. Este proceso podría simplificarse de manera similar al problema con los datos académicos de los ayudantes, con una conexión a la base de datos que estén usando, o con una API que le permita al SPA consultar el estado de cada postulante cuando lo necesite. En el futuro se podría ver la factibilidad de automatizar la obtención de los datos de EAA.

4.4.2 Integración con otros Departamentos

La nueva versión del SPA fue desarrollada pensando en que sea flexible y mantenible, fácilmente extensible. Gracias a la estructura de roles descrita en la figura [2.1], extender el sistema para que otros departamentos puedan usarlo para gestionar sus propios procesos de ayudantías docentes es factible, lo cual le provee al DI la oportunidad de arrendar este sistema, siendo así contratados para administrar y gestionar el proceso de postulación a ayudantías docentes de otros departamentos.

Gracias a la modularidad de Laravel, al momento de recibir autorización de parte de la DTI para usar el SSO de la universidad (posibilidad de iniciar sesión usando Pasaporte USM), implementar y cambiar el tipo de autenticación de usuario de un esquema usuario/contraseña al Pasaporte USM no será complicado. De igual manera, se está abierto a la posibilidad de incluir una conexión a la base de datos institucional de la UTFSM para cerciorarse de la veracidad de los datos académicos de cada ayudante, y así facilitar aún más la gestión de los procesos. Si suficientes departamentos de la UTFSM se adhieren al SPA exitosamente, podría verse la posibilidad de integrar este sistema directamente en el SIGA y así acceder directamente a los datos institucionales.

[35] **D**irección de **E**nseñanza y **A**prendizaje.
[36] http://dea.usm.cl/escuela-de-asistentes-de-aprendizaje/

ANEXOS

A. Migraciones

Los archivos de migración son como un control de versiones para la base de datos, permitiéndole a los desarrolladores modificar y compartir fácilmente el esquema de la base de datos de una aplicación. En Laravel, las migraciones se redactan usando la clase Schema, la cual provee una notación única y agnóstica a la base de datos que se esté utilizando.

Por defecto, *Laravel* provee soporte para los motores de base de datos MySQL, PostgreSQL, SQLite, y SQL Server.

Tabla 'users'

```
Schema::create('users', function (Blueprint $table) {
    $table->increments('id');
    $table->string('nombre');
    $table->string('apellido_paterno');
    $table->string('apellido_materno')->nullable();
    $table->string('genero', 1);
    $table->string('email')->unique();
    $table->string('rut')->nullable()->default(null);
    $table->timestamp('email_verified_at')->nullable();
    $table->string('telefono')->nullable();
    $table->string('avatar')->nullable();
    $table->string('password');
    $table->string('api_token', '60')->default(Str::random(60));
    $table->longText('bio')->nullable();
    $table->rememberToken();
    $table->timestamps();
    $table->softDeletes();
});
```

Tabla 'password_resets'

```
Schema::create('password_resets', function (Blueprint $table) {
    $table->string('email')->index();
    $table->string('token');
    $table->timestamp('created_at')->nullable();
});
```

Tabla 'noticias'

```
Schema::create('noticias', function (Blueprint $table) {
    $table->increments('id');
    $table->string('titulo');
    $table->longText('contenido');
    $table->timestamps();
    $table->softDeletes();
});
```

Tabla ‘etiquetas’

```
Schema::create('etiquetas', function (Blueprint $table) {
    $table->increments('id');
    $table->string('nombre')->unique();
    $table->timestamps();
    $table->softDeletes();
});
```

Tabla ‘etiqueta_noticia’

```
Schema::create('etiqueta_noticia', function (Blueprint $table) {
    $table->increments('id');
    $table->unsignedInteger('etiqueta_id');
    $table->unsignedInteger('noticia_id');
    $table->timestamps();

    $table->foreign('etiqueta_id')->references('id')->on('etiquetas')->onDelete('cascade');
    $table->foreign('noticia_id')->references('id')->on('noticias')->onDelete('cascade');
});
```

Tabla ‘campus’

```
Schema::create('campus', function (Blueprint $table) {
    $table->increments('id');
    $table->string('nombre');
    $table->string('codigo');
    $table->timestamps();
    $table->softDeletes();
});
```

Tabla ‘departamentos’

```
Schema::create('departamentos', function (Blueprint $table) {
    $table->increments('id');
    $table->string('nombre');
    $table->string('codigo')->unique();
    $table->timestamps();
    $table->softDeletes();
});
```

Tabla ‘areas’

```
Schema::create('areas', function (Blueprint $table) {
    $table->increments('id');
    $table->unsignedInteger('departamento_id')->nullable();
    $table->string('nombre');
    $table->string('codigo')->nullable();
    $table->timestamps();
    $table->softDeletes();

    $table->foreign('departamento_id')->references('id')->on('departamentos')->onDelete('set null');
});
```

Tabla ‘tipos_asignatura’

```
Schema::create('tipos_asignatura', function (Blueprint $table) {
    $table->increments('id');
    $table->string('nombre');
    $table->timestamps();
    $table->softDeletes();
});
```

Tabla ‘asignaturas’

```
Schema::create('asignaturas', function (Blueprint $table) {
    $table->increments('id');
    $table->unsignedInteger('area_id')->nullable();
    $table->unsignedInteger('departamento_id');
    $table->unsignedInteger('tipo_asignatura_id')->nullable();
    $table->string('nombre');
    $table->string('sigla')->unique();
    $table->timestamps();
    $table->softDeletes();

    $table->foreign('area_id')->references('id')->on('areas')->onDelete('set null');
    $table->foreign('departamento_id')->references('id')->on('departamentos')->onDelete('restrict');
    $table->foreign('tipo_asignatura_id')->references('id')->on('tipos_asignatura')
    ->onDelete('set null');
});
```

Tabla ‘etapas_proceso’

```
Schema::create('etapas_proceso', function (Blueprint $table) {
    $table->increments('id');
    $table->string('nombre')->unique();
    $table->jsonb('descripcion');
    $table->timestamps();
});
```

Tabla ‘periodos’

```
Schema::create('periodos', function (Blueprint $table) {
    $table->increments('id');
    $table->year('año');
    $table->smallInteger('semestre');
    $table->timestamps();
    $table->softDeletes();
});
```

Tabla ‘procesos’

```
Schema::create('procesos', function (Blueprint $table) {
    $table->increments('id');
    $table->unsignedInteger('departamento_id');
    $table->unsignedInteger('etapa_id');
    $table->unsignedInteger('periodo_id');
    $table->string('slug')->nullable();
    $table->timestamps();
    $table->softDeletes();

    $table->foreign('departamento_id')->references('id')->on('departamentos')->onDelete('restrict');
    $table->foreign('etapa_id')->references('id')->on('etapas_proceso')->onDelete('restrict');
    $table->foreign('periodo_id')->references('id')->on('periodos')->onDelete('restrict');
    $table->unique(['departamento_id', 'periodo_id']);
});
```

Tabla ‘campus_departamento’

```
Schema::create('campus_departamento', function (Blueprint $table) {
    $table->increments('id');
    $table->unsignedInteger('campus_id')->nullable();
    $table->unsignedInteger('departamento_id')->nullable();
    $table->unsignedInteger('jefe_carrera_id')->nullable();
    $table->string('codigo_presupuestario');
    $table->timestamps();
    $table->softDeletes();

    $table->foreign('campus_id')->references('id')->on('campus')->onDelete('set null');
    $table->foreign('departamento_id')->references('id')->on('departamentos')->onDelete('set null');
    $table->foreign('jefe_carrera_id')->references('id')->on('users')->onDelete('set null');
});
```

Tabla ‘paralelos’

```
Schema::create('paralelos', function (Blueprint $table) {
    $table->increments('id');
    $table->unsignedInteger('asignatura_id')->nullable();
    $table->unsignedInteger('campus_id')->nullable();
    $table->unsignedInteger('proceso_id')->nullable();
    $table->unsignedInteger('profesor_id')->default(92);
    $table->integer('paralelo');
    $table->timestamps();
    $table->softDeletes();

    $table->foreign('asignatura_id')->references('id')->on('asignaturas')->onDelete('set null');
    $table->foreign('campus_id')->references('id')->on('campus')->onDelete('set null');
    $table->foreign('proceso_id')->references('id')->on('procesos')->onDelete('set null');
    $table->foreign('profesor_id')->references('id')->on('users')->onDelete('set null');
    $table->unique(['asignatura_id', 'campus_id', 'proceso_id', 'paralelo']);
    $table->index('profesor_id');
});
```

Tabla Bouncer ‘abilities’

```
Schema::create(Models::table('abilities'), function (Blueprint $table) {
    $table->increments('id');
    $table->string('name');
    $table->string('title')->nullable();
    $table->integer('entity_id')->unsigned()->nullable();
    $table->string('entity_type')->nullable();
    $table->boolean('only_owned')->default(false);
    $table->json('options')->nullable();
    $table->integer('scope')->nullable()->index();
    $table->timestamps();
});
```

Tabla Bouncer ‘roles’

```
Schema::create(Models::table('roles'), function (Blueprint $table) {
    $table->increments('id');
    $table->string('name');
    $table->string('title')->nullable();
    $table->integer('level')->unsigned()->nullable();
    $table->integer('scope')->nullable()->index();
    $table->timestamps();

    $table->unique(['name', 'scope'], 'roles_name_unique');
});
```

Tabla Bouncer ‘assigned_roles’

```
Schema::create(Models::table('assigned_roles'), function (Blueprint $table) {
    $table->increments('id');
    $table->integer('role_id')->unsigned()->index();
    $table->integer('entity_id')->unsigned();
    $table->string('entity_type');
    $table->integer('restricted_to_id')->unsigned()->nullable();
    $table->string('restricted_to_type')->nullable();
    $table->integer('scope')->nullable()->index();

    $table->index(['entity_id', 'entity_type', 'scope'], 'assigned_roles_entity_index');
    $table->foreign('role_id')->references('id')->on(Models::table('roles'))
    ->onUpdate('cascade')->onDelete('cascade');
});
```

Tabla Bouncer ‘permissions’

```
Schema::create(Models::table('permissions'), function (Blueprint $table) {
    $table->increments('id');
    $table->integer('ability_id')->unsigned()->index();
    $table->integer('entity_id')->unsigned()->nullable();
    $table->string('entity_type')->nullable();
    $table->boolean('forbidden')->default(false);
    $table->integer('scope')->nullable()->index();

    $table->index(['entity_id', 'entity_type', 'scope'], 'permissions_entity_index');
    $table->foreign('ability_id')->references('id')->on(Models::table('abilities'))
    ->onUpdate('cascade')->onDelete('cascade');
});
```

Tabla ‘tipos_hora’

```
Schema::create('tipos_hora', function (Blueprint $table) {
    $table->increments('id');
    $table->string('nombre')->unique();
    $table->unsignedInteger('valor');
    $table->timestamps();
    $table->softDeletes();
});
```

Tabla ‘vacantes’

```
Schema::create('vacantes', function (Blueprint $table) {
    $table->increments('id');
    $table->unsignedInteger('cantidad');
    $table->boolean('coordinador')->default(false);
    $table->timestamps();
    $table->softDeletes();
});
```

Tabla ‘tipo_hora_vacante’

```
Schema::create('tipo_hora_vacante', function (Blueprint $table) {
    $table->increments('id');
    $table->unsignedInteger('tipo_hora_id');
    $table->unsignedInteger('vacante_id');
    $table->unsignedInteger('horas');
    $table->timestamps();
    $table->softDeletes();

    $table->foreign('tipo_hora_id')->references('id')->on('tipos_hora')->onDelete('set null');
    $table->foreign('vacante_id')->references('id')->on('vacantes')->onDelete('set null');
});
```

Tabla ‘vacante_paralelo’

```
Schema::create('vacante_paralelo', function (Blueprint $table) {
    $table->increments('id');
    $table->unsignedInteger('paralelo_id');
    $table->unsignedInteger('vacante_id');
    $table->timestamps();
    $table->softDeletes();

    $table->foreign('paralelo_id')->references('id')->on('paralelos')->onDelete('set null');
    $table->foreign('vacante_id')->references('id')->on('vacantes')->onDelete('set null');
});
```

Tabla ‘calificaciones’

```
Schema::create('calificaciones', function (Blueprint $table) {
    $table->increments('id');
    $table->unsignedInteger('nota');
    $table->boolean('recomendado');
    $table->string('comentario');
    $table->timestamps();
    $table->softDeletes();
});
```

Tabla ‘estados_postulacion’

```
Schema::create('estados_postulacion', function (Blueprint $table) {
    $table->increments('id');
    $table->string('nombre');
    $table->timestamps();
    $table->softDeletes();
});
```

Tabla ‘motivaciones’

```
Schema::create('motivaciones', function (Blueprint $table) {
    $table->increments('id');
    $table->string('nombre');
    $table->timestamps();
    $table->softDeletes();
});
```

Tabla ‘postulaciones’

```
Schema::create('postulaciones', function (Blueprint $table) {
    $table->increments('id');
    $table->unsignedInteger('calificacion_id')->nullable();
    $table->unsignedInteger('estado_postulacion_id')->default(1);
    $table->unsignedInteger('motivacion_id')->nullable();
    $table->unsignedInteger('postulante_id');
    $table->unsignedInteger('vacante_id');
    $table->unsignedInteger('prioridad')->nullable();
    $table->timestamps();
    $table->softDeletes();

    $table->foreign('calificacion_id')->references('id')->on('calificaciones')->onDelete('set null');
    $table->foreign('estado_postulacion_id')->references('id')->on('estados_postulacion')
    ->onDelete('set null');
    $table->foreign('motivacion_id')->references('id')->on('motivaciones')->onDelete('set null');
    $table->foreign('postulante_id')->references('id')->on('users')->onDelete('set null');
    $table->foreign('vacante_id')->references('id')->on('vacantes')->onDelete('set null');
    $table->unique(['postulante_id', 'vacante_id']);
});
```

Tabla ‘datos_academicos’

```
Schema::create('datos_academicos', function (Blueprint $table) {
    $table->increments('id');
    $table->unsignedInteger('alumno_id');
    $table->unsignedInteger('periodo_id');
    $table->string('rol');
    $table->string('carrera');
    $table->string('campus');
    $table->unsignedInteger('prioridad_academica');
    $table->unsignedInteger('promedio_ponderado');
    $table->unsignedInteger('vtr2');
    $table->unsignedInteger('vtr3');
    $table->string('url_resumen_academico');
    $table->string('certificado_dea');
    $table->timestamps();
    $table->softDeletes();

    $table->foreign('alumno_id')->references('id')->on('users')->onDelete('set null');
    $table->foreign('periodo_id')->references('id')->on('periodos')->onDelete('set null');
});
```

Tabla ‘experiencias’

```
Schema::create('experiencias', function (Blueprint $table) {
    $table->increments('id');
    $table->unsignedInteger('ayudante_id');
    $table->jsonb('data');
    $table->timestamps();
    $table->softDeletes();

    $table->foreign('ayudante_id')->references('id')->on('users')->onDelete('set null');
});
```

B. Bouncer

Otorgar permisos a un usuario

```
Bouncer::allow($user)->to('ban-users');
Bouncer::allow($user)->to('edit', Post::class);
Bouncer::allow($user)->to('delete', $post);

Bouncer::allow($user)->everything();
Bouncer::allow($user)->toManage(Post::class);
Bouncer::allow($user)->toManage($post);
Bouncer::allow($user)->to('view')->everything();

Bouncer::allow($user)->toOwn(Post::class);
Bouncer::allow($user)->toOwnEverything();
```

Quitar permisos a un usuario

```
Bouncer::disallow($user)->to('delete', $post);
Bouncer::disallow($user)->toManage(Post::class);
Bouncer::disallow($user)->toOwn(Post::class);
```

Otorgar y quitar permisos a un rol

```
Bouncer::allow('admin')->to('ban-users');
Bouncer::disallow('admin')->to('ban-users');
```

Prohibir y quitar prohibición de permisos a un usuario

```
Bouncer::forbid($user)->to('delete', $post);
Bouncer::unforbid($user)->to('delete', $post);
```

Sincronizar permisos de un usuario

```
Bouncer::sync($user)->abilities($abilities);
```

Otorgar y quitar roles a un usuario

```
Bouncer::assign('admin')->to($user);
Bouncer::retract('admin')->from($user);
```

Otorgar rol a un grupo de usuarios por su id

```
Bouncer::assign('admin')->to([1,2,3]);
```

Sincronizar roles de un usuario

```
Bouncer::sync($user)->roles($roles);
```

Revisar si el usuario actual tiene un permiso

```
$boolean = Bouncer::can('ban-users');
$boolean = Bouncer::can('edit', Post::class);
$boolean = Bouncer::can('delete', $post);

$boolean = Bouncer::cannot('ban-users');
$boolean = Bouncer::cannot('edit', Post::class);
$boolean = Bouncer::cannot('delete', $post);
```

Revisar los roles de un usuario.

```
$boolean = Bouncer::is($user)->a('subscriber');
$boolean = Bouncer::is($user)->an('admin');
$boolean = Bouncer::is($user)->notA('subscriber');
$boolean = Bouncer::is($user)->notAn('admin');
$boolean = Bouncer::is($user)->a('moderator', 'editor');
$boolean = Bouncer::is($user)->all('moderator', 'editor');
```

Administrar caché de permisos.

```
Bouncer::cache();
Bouncer::dontCache();

Bouncer::refresh();
Bouncer::refreshFor($user);
```

Acceder a funciones anteriores a partir del modelo de Usuario.

```
$user->allow('ban-users');
$user->allow('edit', Post::class);
$user->allow('delete', $post);

$user->disallow('ban-users');
$user->disallow('edit', Post::class);
$user->disallow('delete', $post);

$user->assign('admin');
$user->retract('admin');

$boolean = $user->isAn('admin');
$boolean = $user->isAn('editor', 'moderator');
$boolean = $user->isAll('moderator', 'editor');
$boolean = $user->isNotAn('admin', 'moderator');
```

Filtrar usuarios por roles

```
$users = User::whereIs('superadmin')->get();
$users = User::whereIs('superadmin', 'admin')->get();
$users = User::whereIsAll('sales', 'marketing')->get();

$abilities = $user->getAbilities();
$forbidden = $user->getForbiddenAbilities();
```

C. Modelo de Datos Completo

Además de lo que se muestra en la figura [4.1.1], este modelo de datos incluye las 4 tablas creadas por Bouncer y 2 tablas extra proveídas por Laravel:

- **migrations**: Es una tabla que guarda información acerca de las migraciones descritas en el Anexo A para que el sistema sepa si debe correrlas o no.
- **password_resets**: Es una tabla que guarda información una traza de todos los usuarios (a través de sus emails) que han tenido que resetear sus contraseñas.

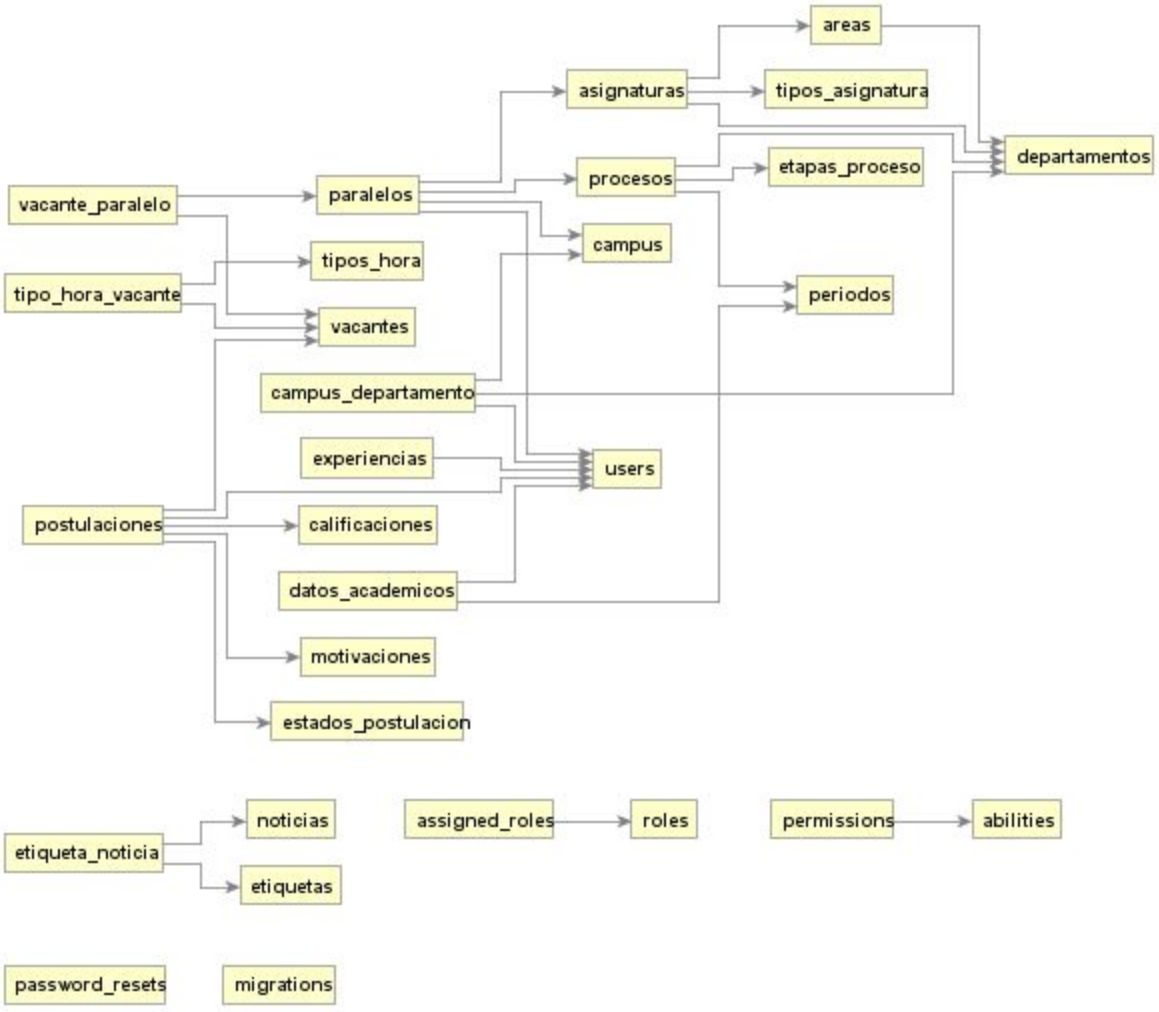


Figura C.1: Modelo de datos completo. Fuente: Elaboración propia usando DbVisualizer

D. Casos de Uso

CU1 Registrar cuenta nueva	
Rol(es)	Usuario Guest (Sin cuenta)
Descripción	Usuario puede crearse una cuenta nueva llenando y enviando un formulario con su nombre e email en la página de registro, accesible desde la página principal.
Diferencias por Rol	No aplicable.

CU2 Iniciar Sesión	
Rol(es)	Usuario Guest (Sin cuenta)
Descripción	Usuario puede iniciar sesión colocando sus datos (email y contraseña) y haciendo click en el botón de **Login** en la página principal.
Diferencias por Rol	No aplicable.

CU3 Recuperar Contraseña	
Rol(es)	Usuario Guest (Sin cuenta)
Descripción	Usuario puede recuperar su contraseña en caso de olvido haciendo click en el botón **Recuperar Contraseña** en la página principal.
Diferencias por Rol	No aplicable.

CU4 Cerrar Sesión	
Rol(es)	Todos
Descripción	Usuario puede cerrar su sesión haciendo click en el botón de **Logout**, ubicado al fondo de la barra lateral y en un menú desplegable bajo su foto de perfil en la esquina superior derecha.
Diferencias por Rol	Ninguna.

CU5 Ver Noticias	
Rol(es)	Todos
Descripción	Usuario puede ver las noticias del sistema desde la página principal.
Diferencias por Rol	Ninguna.

CU5.1 Crear Noticias	
Rol(es)	Subdirección de Pregrado/Administrador de Departamento
Descripción	Usuario puede crear un noticia nueva haciendo click en el botón **Crear Noticia**, llenando y enviando el formulario desplegable.
Diferencias por Rol	Ninguna.

CU5.2 Editar Noticias	
Rol(es)	Subdirección de Pregrado/Administrador de Departamento
Descripción	Usuario puede editar una noticia haciendo click en el botón para ello (representado por un lápiz amarillo bajo el título de la noticia (CU5)), llenando y enviando el formulario desplegable.
Diferencias por Rol	Ninguna.

CU5.3 Eliminar Noticias	
Rol(es)	Subdirección de Pregrado/Administrador de Departamento
Descripción	Usuario puede editar una noticia haciendo click en el botón para ello (representado por una cruz roja bajo el título de la noticia (CU5)) y confirmando su decisión en el mensaje desplegable.
Diferencias por Rol	Ninguna.

CU5.4 Restaurar Noticias	
Rol(es)	Subdirección de Pregrado/Administrador de Departamento
Descripción	Usuario puede editar una noticia haciendo click en el botón para ello (representado por una flecha verde bajo el título de la noticia (CU5)) y confirmando su decisión en el mensaje desplegable.
Diferencias por Rol	Ninguna.

CU6 Ver Perfil Propio	
Rol(es)	Todos
Descripción	Usuario puede ver su propio perfil haciendo click en el botón **Ver Perfil** ubicado en el menú lateral.
Diferencias por Rol	Ninguna.

CU6.1 Actualizar Perfil	
Rol(es)	Todos
Descripción	Usuario puede actualizar su foto de perfil y agregar una descripción corta de sí mismo haciendo click en el botón **Actualizar Perfil** ubicado en la página de perfil. (CU6)
Diferencias por Rol	Ninguna.

CU6.2 Actualizar Datos Académicos	
Rol(es)	Ayudante
Descripción	Usuario puede actualizar sus datos académicos haciendo click en el botón **Actualizar Datos**, ubicado en la pestaña **Datos Académicos** de su perfil (CU6) y llenando el formulario desplegable.
Diferencias por Rol	No aplica.

CU6.2 Agregar Experiencias Previas	
Rol(es)	Ayudante
Descripción	Usuario puede actualizar sus datos académicos haciendo click en el botón **Agregar Experiencia**, ubicado en la pestaña **Experiencias** de su perfil (CU6) y llenando el formulario desplegable. Las experiencias de departamentos adheridos al SPA serán actualizadas de manera automática al cerrar cada proceso.
Diferencias por Rol	No aplica.

CU7 Ver Perfil Ajeno	
Rol(es)	Todos
Descripción	Usuario puede ver perfiles ajenos haciendo click en el nombre de otras personas que aparezcan en diversos listados.
Diferencias por Rol	Ninguna.

CU7.1 Ver Datos Académicos Ajenos	
Rol(es)	Profesor/Secretaría de Pregrado/Subdirección de Pregrado/Administrador de Departamento
Descripción	Usuario puede ver los datos académicos de un usuario ayudante en su perfil (CU7) haciendo click en la pestaña **Datos Académicos**.
Diferencias por Rol	Ninguna.

CU7.1.1 Descargar Resumen Académico Ajeno	
Rol(es)	Profesor/Secretaría de Pregrado/Subdirección de Pregrado/Administrador de Departamento
Descripción	Usuario puede descargar el resumen académico de un usuario ayudante en su perfil (CU7) haciendo click en el botón **Ver Resumen Académico** ubicado entre sus datos académicos (CU7.1)
Diferencias por Rol	Ninguna.

CU8 Ver lista de procesos	
Rol(es)	Ayudante/Profesor/Subdirección de Pregrado/Administrador de Departamento
Descripción	Usuario puede ver lista de procesos haciendo click en el botón **Procesos Activos** ubicado en el menú lateral. En caso de que solo exista un proceso, será redirigido al listado de asignaturas y paralelos (CU8.1)
Diferencias por Rol	➢ Ayudante: Puede ver todos los procesos que estén activos ➢ Profesor: Puede ver sólo los procesos de sus departamentos que estén activos. ➢ Subdirección de Pregrado/Administrador de Departamento: Pueden ver solo los procesos de sus departamento.

CU8.1 Ver listado de asignaturas y paralelos de un proceso	
Rol(es)	Ayudante/Profesor/Subdirección de Pregrado/Administrador de Departamento
Descripción	Usuario puede ver un listado filtrable de las asignaturas de un proceso, sus paralelos y profesores asignados haciendo click en uno de los procesos de la lista (CU8)
Diferencias por Rol	➢ Ayudante: Puede ver todas las asignaturas y paralelos activos ➢ Profesor: Puede ver sólo las asignaturas en las cuales está asignados a algún paralelo. ➢ Subdirección de Pregrado/Administrador de Departamento: Pueden ver todas las asignaturas.

CU8.2 Editar una Asignatura	
Rol(es)	Subdirección de Pregrado/Administrador de Departamento
Descripción	Usuario puede editar una asignatura del listado (CU8.1) haciendo click en el botón para editarla (representado por un lápiz amarillo a la derecha de la asignatura), llenando y enviando el formulario desplegable.
Diferencias por Rol	Ninguna.

CU8.3 Eliminar una Asignatura	
Rol(es)	Subdirección de Pregrado/Administrador de Departamento
Descripción	Usuario puede eliminar una asignatura del listado (CU8.1) haciendo click en el botón para eliminarla (representado por una cruz roja a la derecha de la asignatura) y confirmando su decisión en el mensaje desplegable.
Diferencias por Rol	Ninguna.

CU8.4 Restaurar una Asignatura eliminada	
Rol(es)	Subdirección de Pregrado/Administrador de Departamento
Descripción	Usuario puede restaurar una asignatura eliminada del listado (CU8.1) haciendo click en el botón para restaurarla (representado por una flecha verde a la derecha de la asignatura, visible sólo para asignaturas eliminadas) y confirmando su decisión en el mensaje desplegable.
Diferencias por Rol	Ninguna.

CU8.5 Crear un Paralelo para una Asignatura	
Rol(es)	Subdirección de Pregrado/Administrador de Departamento
Descripción	Usuario puede crear un paralelo para una asignatura del listado (CU8.1) haciendo click en el botón para ello (representado por un signo “+” verde a la derecha de la asignatura), llenando y enviando el formulario desplegable.
Diferencias por Rol	Ninguna.

CU8.6 Editar un Paralelo	
Rol(es)	Subdirección de Pregrado/Administrador de Departamento
Descripción	Usuario puede editar una asignatura del listado (CU8.1) haciendo click en el botón para desplegar los paralelos (representado por una signo “+” verde a la izquierda de la asignatura), luego en el botón para editar el paralelo (representado por un lápiz amarillo a la derecha del paralelo), llenando y enviando el formulario desplegable.
Diferencias por Rol	Ninguna.

CU8.7 Eliminar un Paralelo de una Asignatura	
Rol(es)	Subdirección de Pregrado/Administrador de Departamento
Descripción	Usuario puede editar una asignatura del listado (CU8.1) haciendo click en el botón para desplegar los paralelos (representado por una signo “+” verde a la izquierda de la asignatura), luego en el botón para eliminar el paralelo (representado por una cruz roja a la derecha del paralelo) y confirmando su decisión en el mensaje desplegable.
Diferencias por Rol	Ninguna.

CU8.8 Restaurar un Paralelo de una Asignatura	
Rol(es)	Subdirección de Pregrado/Administrador de Departamento
Descripción	Usuario puede editar una asignatura del listado (CU8.1) haciendo click en el botón para desplegar los paralelos (representado por una signo “+” verde a la izquierda de la asignatura), luego en el botón para restaurar el paralelo (representado por una flecha verde a la derecha del paralelo, visible sólo para paralelos eliminados) y confirmando su decisión en el mensaje desplegable.
Diferencias por Rol	Ninguna.

CU8.9 Ver Vacantes de un Paralelo	
Rol(es)	Ayudante/Profesor/Subdirección de Pregrado/Administrador de Departamento
Descripción	Usuario puede ver las vacantes de un paralelo de una asignatura del listado (CU8.1) haciendo click en el botón para desplegar los paralelos (representado por una signo “+” verde a la izquierda de la asignatura) y luego en el botón para ver más información del paralelo (representado por una lupa celeste a la derecha del paralelo)
Diferencias por Rol	Ninguna.

CU8.9.1 Agregar Vacantes a un Paralelo	
Rol(es)	Subdirección de Pregrado/Administrador de Departamento
Descripción	Usuario agregar una vacante a un paralelo en la página de éste último (CU8.9) haciendo click en el botón **Agregar Vacante**, llenando y enviando el formulario desplegable.
Diferencias por Rol	Ninguna.

CU8.9.2 Editar Vacantes de un Paralelo	
Rol(es)	Subdirección de Pregrado/Administrador de Departamento
Descripción	Usuario editar una vacante de un paralelo en la página de éste último (CU8.9) haciendo click en el botón para editarla (representado por un lápiz amarillo a la derecha de la vacante), llenando y enviando el formulario desplegable.
Diferencias por Rol	Ninguna.

CU8.9.3 Eliminar Vacante de un Paralelo	
Rol(es)	Subdirección de Pregrado/Administrador de Departamento
Descripción	Usuario eliminar una vacante de un paralelo en la página de éste último (CU8.9) haciendo click en el botón para eliminarla (representado por una cruz roja a la derecha de la vacante) y confirmando su decisión en el mensaje desplegable.
Diferencias por Rol	Ninguna.

CU8.9.4 Postular a Vacante de un Paralelo	
Rol(es)	Ayudante
Descripción	Usuario puede postular a una vacante de un paralelo en la página de éste último (CU8.9) haciendo click en el botón para agregar postulación (representado por un signo “+” verde a la derecha de la vacante), llenando y enviando el formulario desplegable siempre y cuando el Proceso se encuentre en una fase que permita la postulación.
Diferencias por Rol	Ninguna.

CU8.9.5 Ver Postulantes de una Vacante	
Rol(es)	Profesor/Subdirección de Pregrado/Administrador de Departamento
Descripción	Usuario puede ver una lista re-ordenable de postulantes a una vacante de un paralelo en la página de éste último (CU8.9) haciendo click en el botón para ver postulantes (representado por una lupa azul a la derecha de la vacante).
Diferencias por Rol	Ninguna.

CU8.9.5.1 Seleccionar Postulante	
Rol(es)	Profesor/Subdirección de Pregrado/Administrador de Departamento
Descripción	Usuario puede seleccionar un postulante de la lista (CU8.9.5) haciendo click en la casilla de verificación a la derecha de sus datos. Siempre y cuando: ➢ El Proceso esté en una fase que permita la selección/des-selección de ayudantes. ➢ La Vacante todavía tenga cupos libres.
Diferencias por Rol	Ninguna.

CU8.9.5.2 Deshacer Selección	
Rol(es)	Profesor/Subdirección de Pregrado/Administrador de Departamento
Descripción	Usuario puede des-seleccionar un postulante de la lista (CU8.9.5) haciendo click en la casilla de verificación a la derecha de sus datos. Siempre y cuando el Proceso esté en una fase que permita la selección/des-selección de ayudantes.
Diferencias por Rol	Ninguna.

CU8.9.5.3 Ver Experiencia previa de Postulante	
Rol(es)	Profesor/Subdirección de Pregrado/Administrador de Departamento
Descripción	Usuario puede ver las experiencias previas de un postulante de la lista (CU8.9.5) haciendo click en el botón para ello en la columna de Experiencias.
Diferencias por Rol	Ninguna.

CU8.9.5.4 Ver otras postulaciones activas de Postulante	
Rol(es)	Profesor/Subdirección de Pregrado/Administrador de Departamento
Descripción	Usuario puede ver las experiencias previas de un postulante de la lista (CU8.9.5) haciendo click en el botón para ello en la columna de Postulaciones.
Diferencias por Rol	Ninguna.

CU9 Ver Postulaciones propias	
Rol(es)	Ayudante
Descripción	Usuario puede ver un listado de sus postulaciones en los procesos activos en el sistema haciendo click en el botón **Mis Postulaciones** ubicado en el menú lateral.
Diferencias por Rol	No aplica.

CU9.1 Asignar prioridad a Postulaciones propias	
Rol(es)	Ayudante
Descripción	Usuario puede reordenar el listado de sus postulaciones (CU9) con lo cual se actualiza su prioridad asignada.
Diferencias por Rol	No aplica.

CU10 Gestionar Roles de Usuario	
Rol(es)	Subdirección de Pregrado/Administrador de Departamento
Descripción	Usuario puede ver un listado de los usuarios activos del sistema y sus respectivos roles haciendo click en el botón **Gestionar Roles** ubicado en el menú lateral, dentro del submenú **Administración**.
Diferencias por Rol	Ninguna.

CU10.1 Cambiar Roles de Usuario	
Rol(es)	Subdirección de Pregrado/Administrador de Departamento
Descripción	Usuario puede cambiar los roles de un usuario en la página para ello (CU10), seleccionándolo de la lista **Usuario** y haciendo click en los roles del listado de **Roles Disponibles** para asignárselos al usuario o haciendo click en los roles del listado de **Roles Usuario** para quitárselos y finalmente haciendo click en **Actualizar Roles** al terminar.
Diferencias por Rol	Ninguna.

CU11 Ver listado de Procesos gestionables	
Rol(es)	Secretaría de Pregrado/Subdirección de Pregrado/Administrador de Departamento
Descripción	Usuario puede ir a la página para gestionar los Procesos del sistema haciendo click en el botón **Gestionar Procesos** ubicado en el menú lateral, en el submenú **Administración**.
Diferencias por Rol	Ninguna.

CU11.1 Crear nuevo Proceso	
Rol(es)	Subdirección de Pregrado/Administrador de Departamento
Descripción	Usuario puede crear un proceso nuevo en la página de listado de procesos gestionables (CU11) haciendo click en el botón **Abrir Proceso Nuevo**, llenando y enviando el formulario desplegable. Al finalizar, será redirigido a la página de gestión del Proceso.
Diferencias por Rol	Ninguna.

CU11.2 Ver página de gestión de Proceso	
Rol(es)	Subdirección de Pregrado/Administrador de Departamento
Descripción	Usuario puede ver la página de gestión de un Proceso haciendo click en su nombre en el listado de procesos gestionables (CU11).
Diferencias por Rol	Ninguna.

CU11.2.1 Descargar Nómina de Ayudantes del Proceso	
Rol(es)	Subdirección de Pregrado/Administrador de Departamento
Descripción	Usuario puede descargar la nómina de ayudantes seleccionados del proceso desde la página de gestión de un Proceso (CU11.2) haciendo click en el enlace **Descargar Nómina de Ayudantes Seleccionados**.
Diferencias por Rol	Ninguna.

CU11.2.2 Descargar Formato de Carga Masiva para el Proceso	
Rol(es)	Subdirección de Pregrado/Administrador de Departamento
Descripción	Usuario puede descargar la nómina de ayudantes seleccionados del proceso desde la página de gestión de un Proceso (CU11.2) haciendo click en el enlace **Descargar Formato Carga Masiva**.
Diferencias por Rol	Ninguna.

CU11.2.3 Cambiar Estado Proceso	
Rol(es)	Subdirección de Pregrado/Administrador de Departamento
Descripción	Usuario puede descargar la nómina de ayudantes seleccionados del proceso desde la página de gestión de un Proceso (CU11.2) eligiendo un estado de la lista **Fase Actual** y haciendo click en el botón **Actualizar Estado**.
Diferencias por Rol	Ninguna.

CU11.2.4 Archivar Proceso	
Rol(es)	Subdirección de Pregrado/Administrador de Departamento
Descripción	Usuario puede descargar la nómina de ayudantes seleccionados del proceso desde la página de gestión de un Proceso (CU11.2) haciendo click en el botón **Archivar Proceso** y confirmando su decisión en el mensaje desplegable.
Diferencias por Rol	Ninguna.

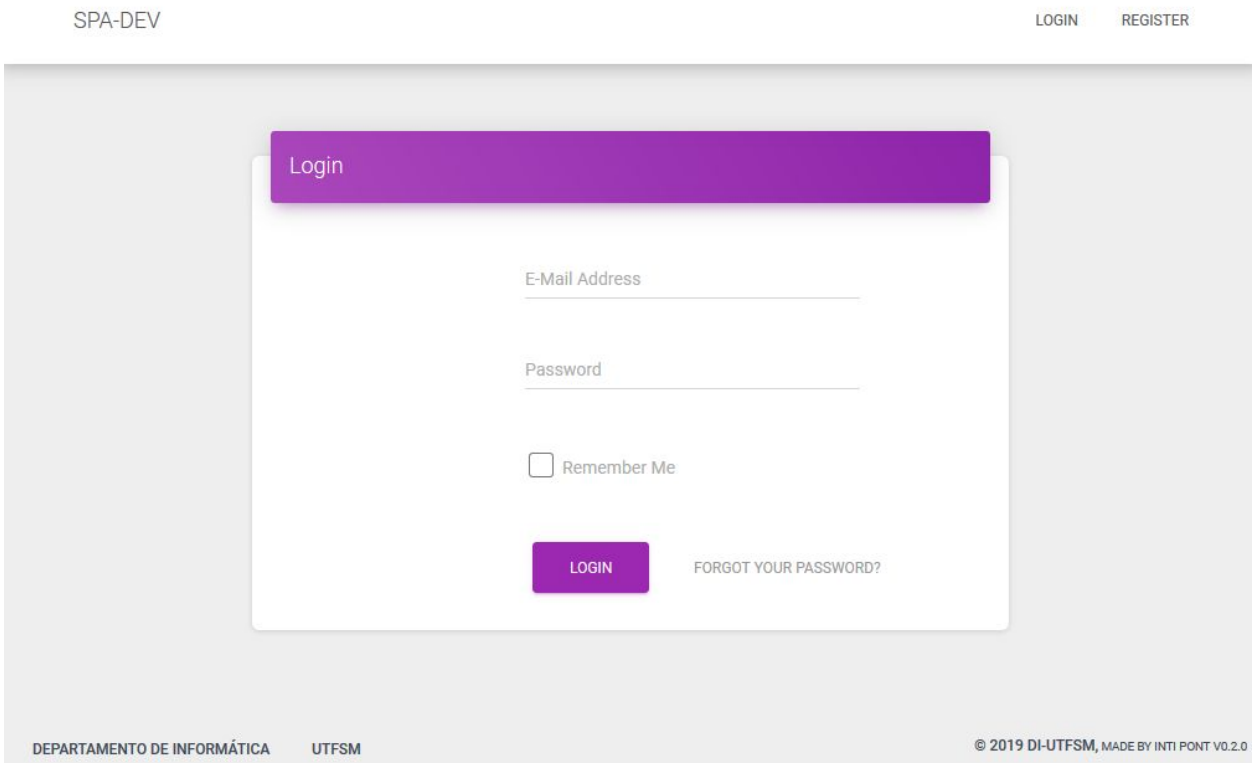
CU12 Ir a página de Carga Masiva	
Rol(es)	Subdirección de Pregrado/Administrador de Departamento
Descripción	Usuario puede ir a la página para realizar la carga masiva haciendo click en el botón **Carga Masiva** ubicado en el menú lateral, en el submenú **Administración**.
Diferencias por Rol	Ninguna.

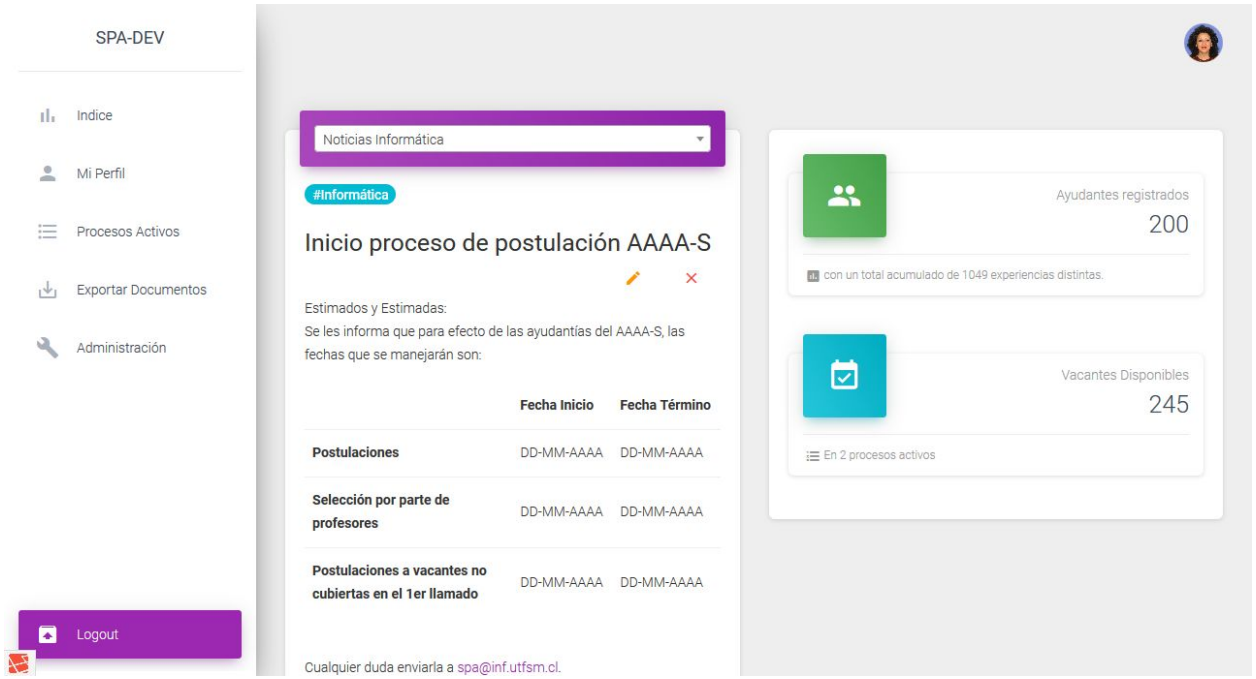
CU12.1 Realizar Carga Masiva	
Rol(es)	Subdirección de Pregrado/Administrador de Departamento
Descripción	Usuario puede realizar la carga masiva desde su página (CU12), haciendo click en el botón **Seleccione Archivo** para subir el archivo con formato de Carga Masiva (CU11.2.2) completado y haciendo click en el botón **Cargar**.
Diferencias por Rol	Ninguna.

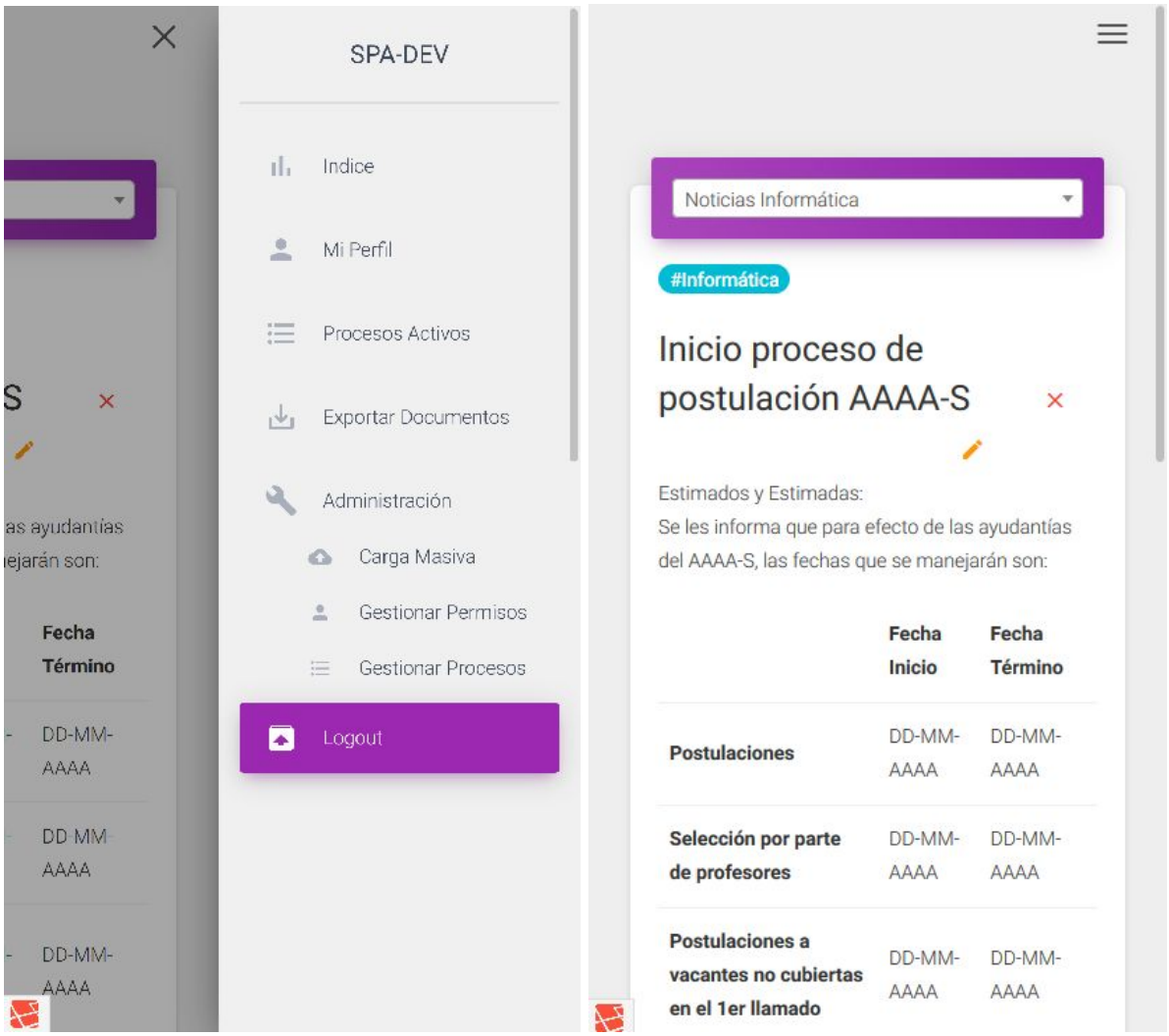
CU12.1.1 Ver resultados de Carga Masiva	
Rol(es)	Subdirección de Pregrado/Administrador de Departamento
Descripción	Usuario puede ver los resultados de la Carga Masiva en su página (CU12), en una lista que se desplegará justo después de realizarla (CU12.1).
Diferencias por Rol	Ninguna.

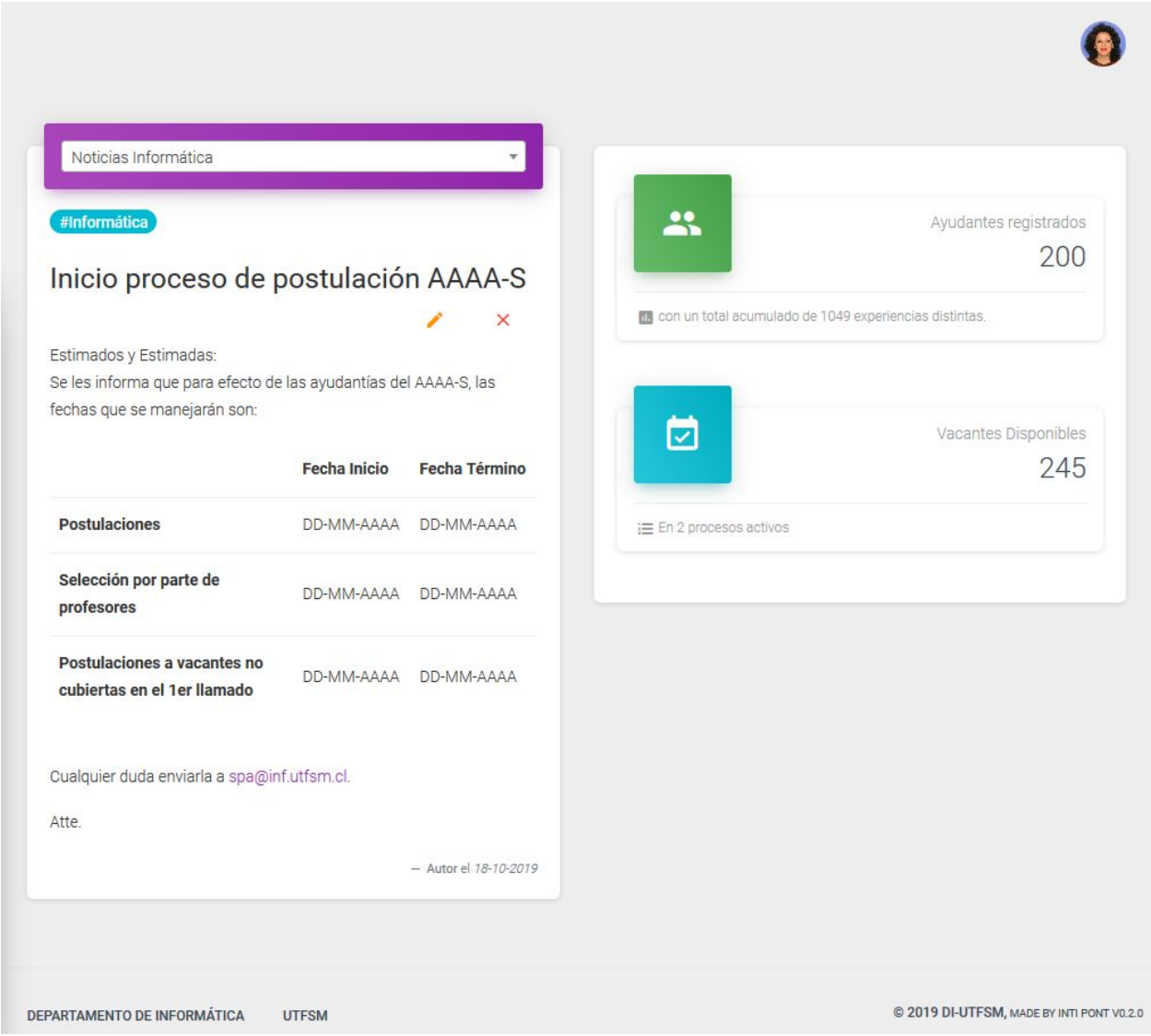
CU13 Calificar Ayudantes	
Rol(es)	Profesor
Descripción	Usuario puede calificar a sus ayudantes con unos cuantos clicks respondiendo una mini-encuesta desplegable que le será enviada al término de cada proceso.
Diferencias por Rol	No Aplica.

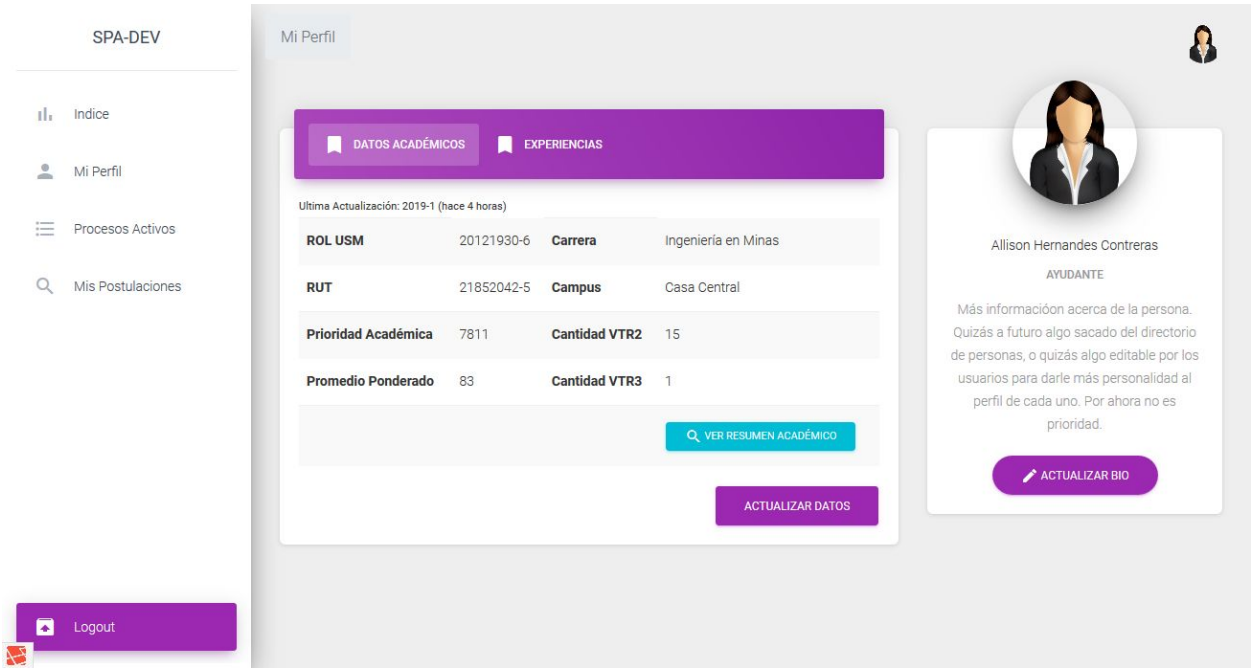
E. Interfaz

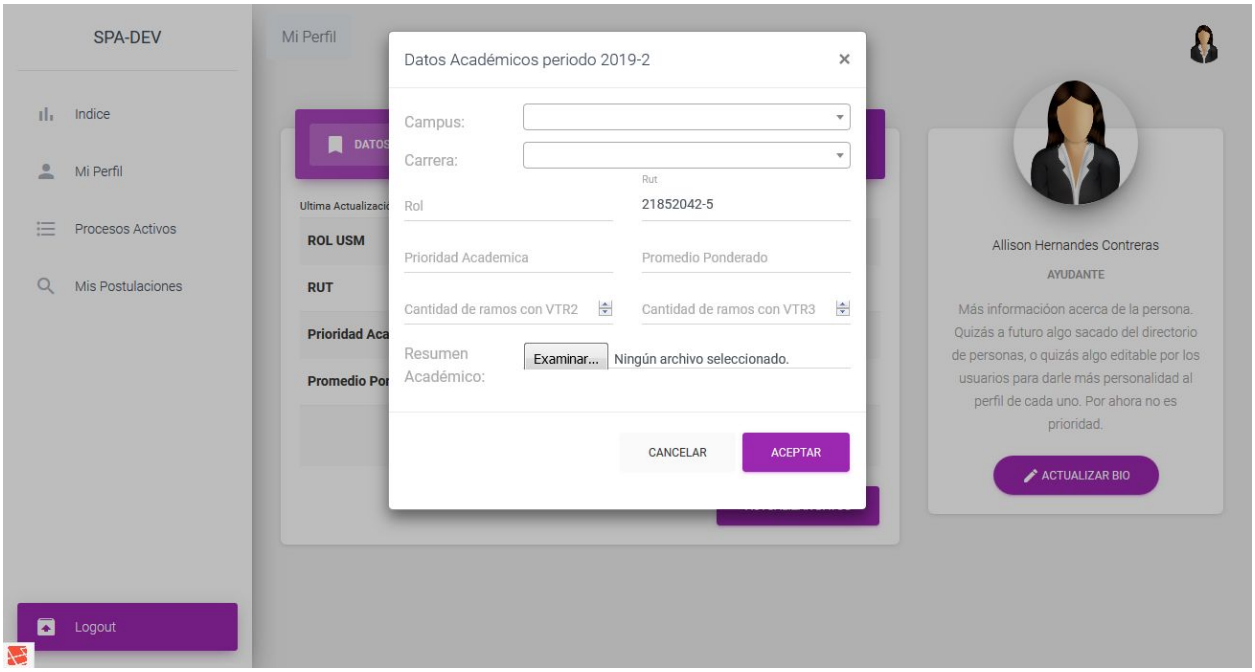

SPA-DEV
LOGIN
REGISTER
Login
E-Mail Address
Password
Remember Me
LOGIN
FORGOT YOUR PASSWORD?
DEPARTAMENTO DE INFORMÁTICA
UTFSM
© 2019 DI-UTFSM, MADE BY INTI PONT V0.2.0

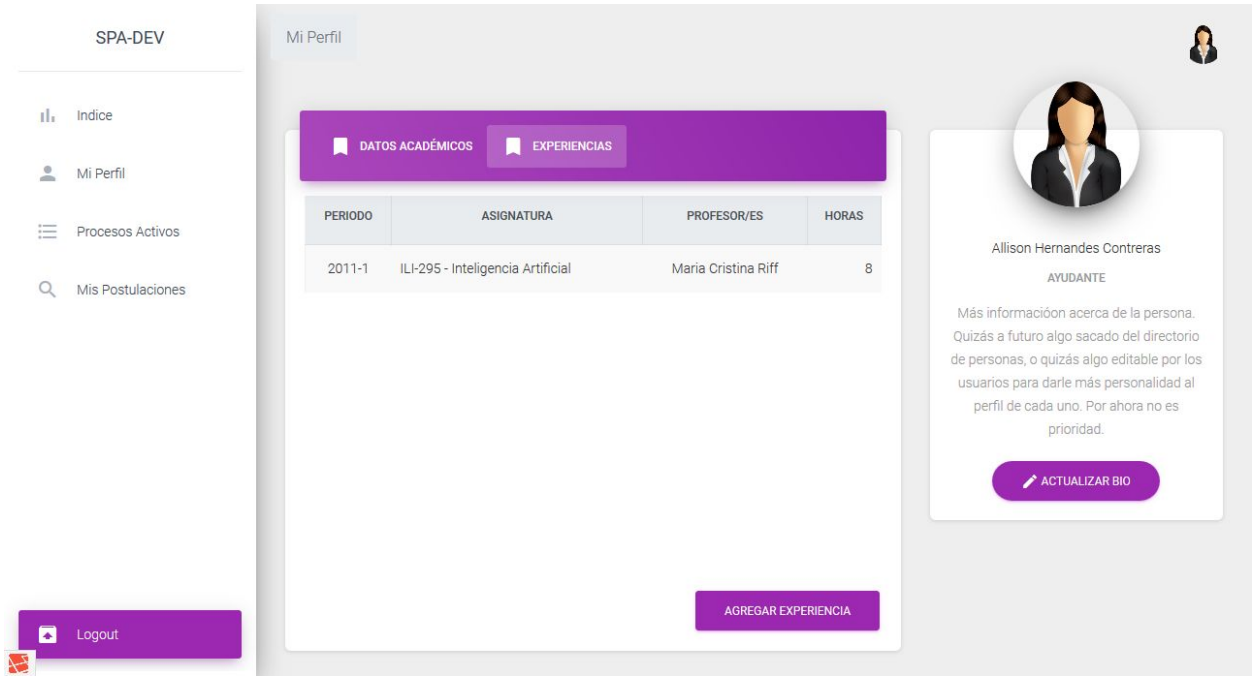

SPA-DEV
Indice
Mi Perfil
Procesos Activos
Exportar Documentos
Administración
Logout
Noticias Informática
#Informática
Inicio proceso de postulación AAAA-S
Estimados y Estimadas:
Se les informa que para efecto de las ayudantías del AAAA-S, las fechas que se manejarán son:
Fecha Inicio
Fecha Término
Postulaciones
DD-MM-AAAA
DD-MM-AAAA
Selección por parte de profesores
DD-MM-AAAA
DD-MM-AAAA
Postulaciones a vacantes no cubiertas en el 1er llamado
DD-MM-AAAA
DD-MM-AAAA
Cualquier duda enviarla a spa@inf.utfsm.cl.
Ayudantes registrados
200
con un total acumulado de 1049 experiencias distintas.
Vacantes Disponibles
245
En 2 procesos activos

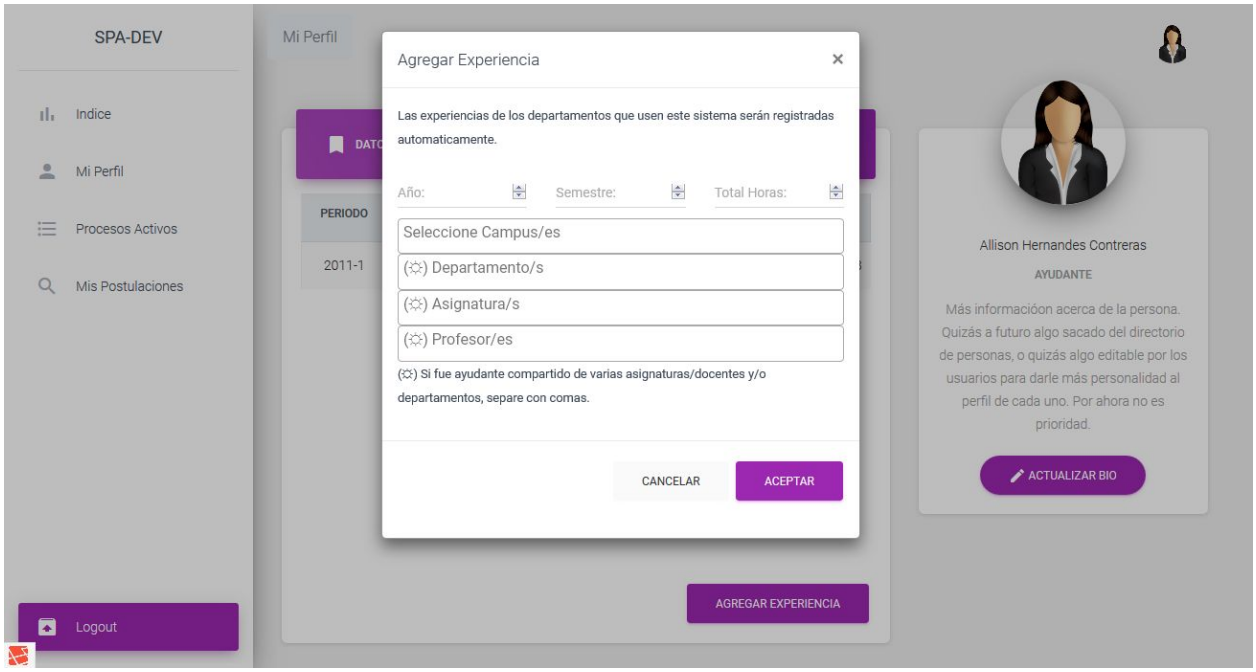

SPA-DEV
Indice
Mi Perfil
Procesos Activos
Exportar Documentos
Administración
Carga Masiva
Gestionar Permisos
Gestionar Procesos
Logout
Noticias Informática
#Informática
Inicio proceso de postulación AAAA-S
Estimados y Estimadas:
Se les informa que para efecto de las ayudantías del AAAA-S, las fechas que se manejarán son:
Fecha Inicio
Fecha Término
Postulaciones
DD-MM-AAAA
DD-MM-AAAA
Selección por parte de profesores
DD-MM-AAAA
DD-MM-AAAA
Postulaciones a vacantes no cubiertas en el 1er llamado
DD-MM-AAAA
DD-MM-AAAA

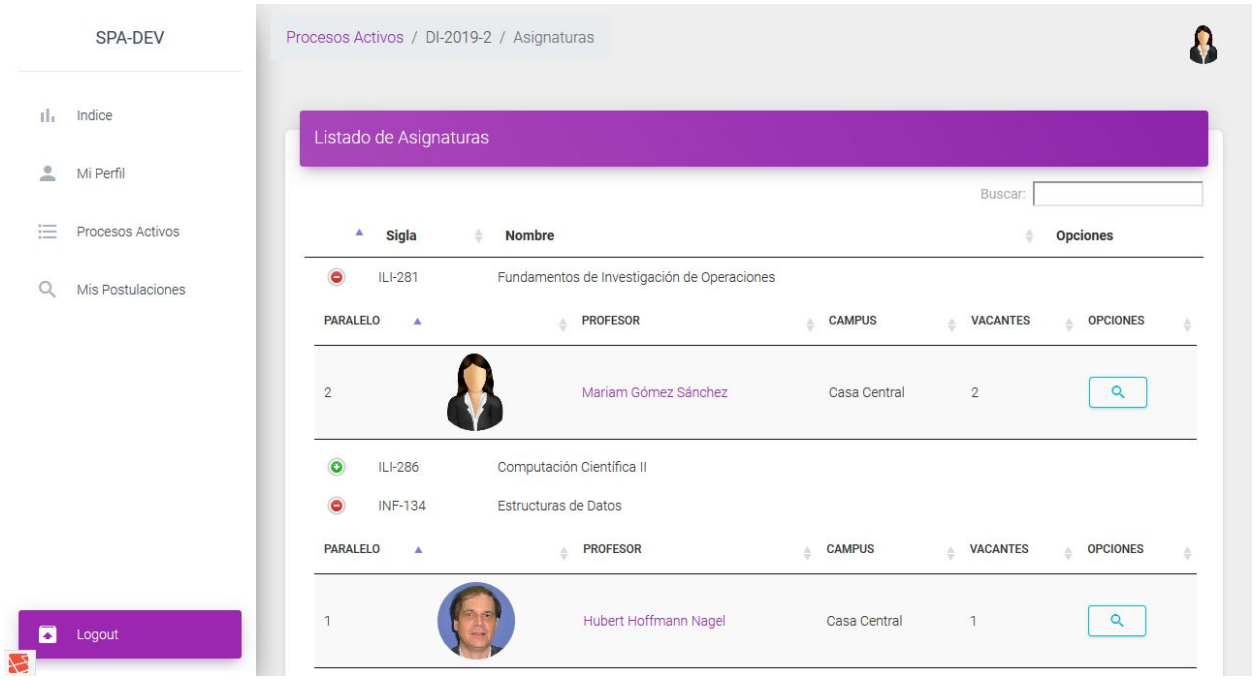

Noticias Informática
#Informática
Inicio proceso de postulación AAAA-S
Estimados y Estimadas:
Se les informa que para efecto de las ayudantías del AAAA-S, las fechas que se manejarán son:
Fecha Inicio
Fecha Término
Postulaciones
DD-MM-AAAA
DD-MM-AAAA
Selección por parte de profesores
DD-MM-AAAA
DD-MM-AAAA
Postulaciones a vacantes no cubiertas en el 1er llamado
DD-MM-AAAA
DD-MM-AAAA
Cualquier duda enviarla a spa@inf.utfsm.cl.
Atte.
— Autor el 18-10-2019
Ayudantes registrados
200
con un total acumulado de 1049 experiencias distintas.
Vacantes Disponibles
245
En 2 procesos activos
DEPARTAMENTO DE INFORMÁTICA
UTFSM
© 2019 DI-UTFSM, MADE BY INTI PONT V0.2.0

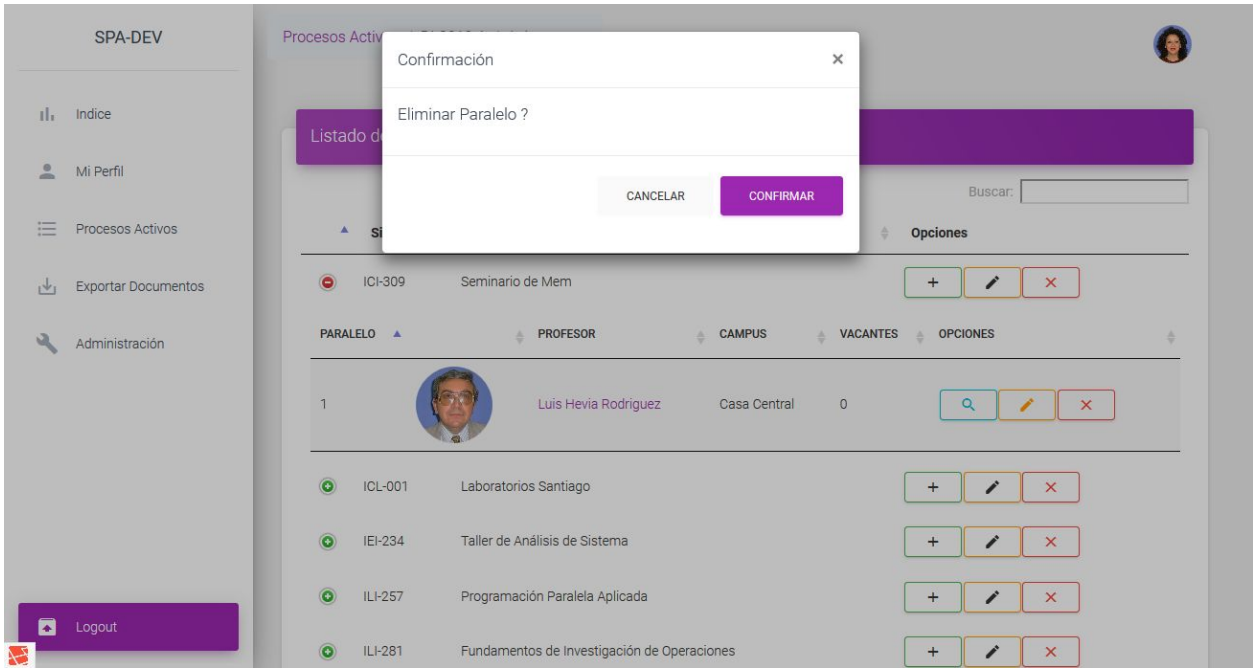

SPA-DEV
Indice
Mi Perfil
Procesos Activos
Mis Postulaciones
Logout
Mi Perfil
DATOS ACADÉMICOS
EXPERIENCIAS
Ultima Actualización: 2019-1 (hace 4 horas)
ROL USM 20121930-6 Carrera Ingeniería en Minas
RUT 21852042-5 Campus Casa Central
Prioridad Académica 7811 Cantidad VTR2 15
Promedio Ponderado 83 Cantidad VTR3 1
VER RESUMEN ACADÉMICO
ACTUALIZAR DATOS
Allison Hernandes Contreras
AYUDANTE
Más informacióon acerca de la persona. Quizás a futuro algo sacado del directorio de personas, o quizás algo editable por los usuarios para darle más personalidad al perfil de cada uno. Por ahora no es prioridad.
ACTUALIZAR BIO

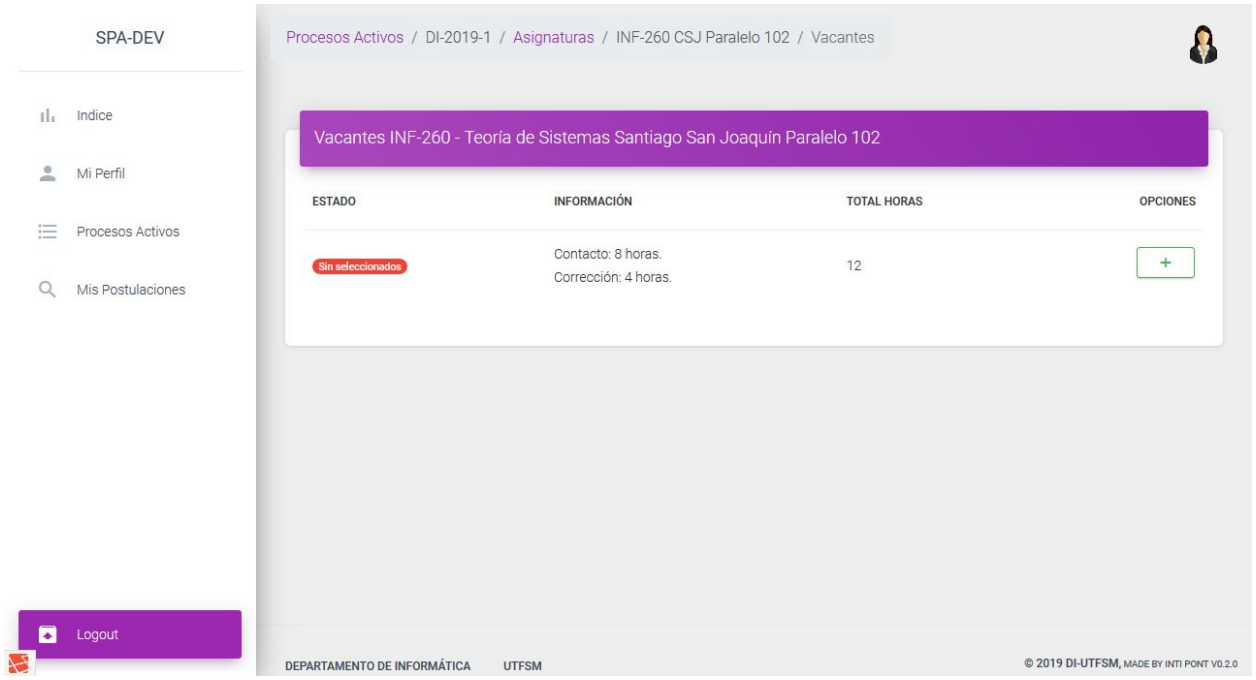

SPA-DEV
Indice
Mi Perfil
Procesos Activos
Mis Postulaciones
Logout
Mi Perfil
Datos Académicos periodo 2019-2
Campus:
Carrera:
Rut
Rol
21852042-5
Prioridad Academica
Promedio Ponderado
Cantidad de ramos con VTR2
Cantidad de ramos con VTR3
Resumen Académico:
Examinar...
Ningún archivo seleccionado.
CANCELAR
ACEPTAR
Allison Hernandes Contreras
AYUDANTE
Más informacióon acerca de la persona. Quizás a futuro algo sacado del directorio de personas, o quizás algo editable por los usuarios para darle más personalidad al perfil de cada uno. Por ahora no es prioridad.
ACTUALIZAR BIO

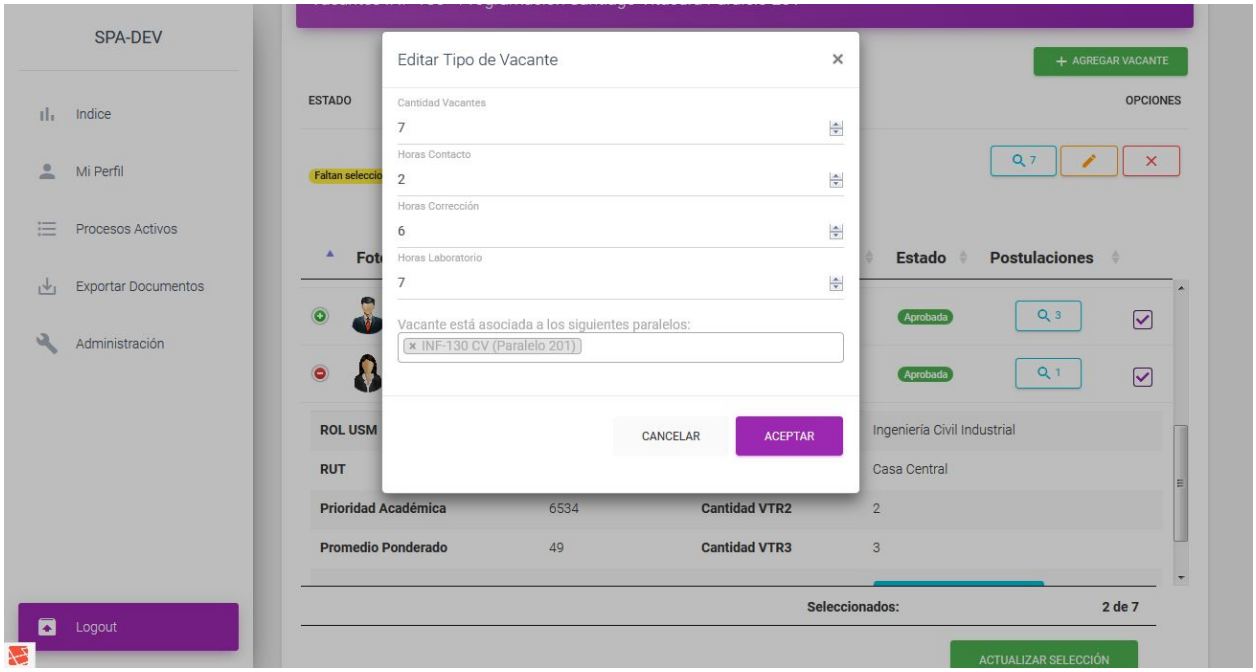

SPA-DEV
Indice
Mi Perfil
Procesos Activos
Mis Postulaciones
Logout
Mi Perfil
DATOS ACADÉMICOS
EXPERIENCIAS
PERIODO
ASIGNATURA
PROFESOR/ES
HORAS
2011-1
ILI-295 - Inteligencia Artificial
Maria Cristina Riff
8
AGREGAR EXPERIENCIA
Allison Hernandes Contreras
AYUDANTE
Más informacióon acerca de la persona. Quizás a futuro algo sacado del directorio de personas, o quizás algo editable por los usuarios para darle más personalidad al perfil de cada uno. Por ahora no es prioridad.
ACTUALIZAR BIO

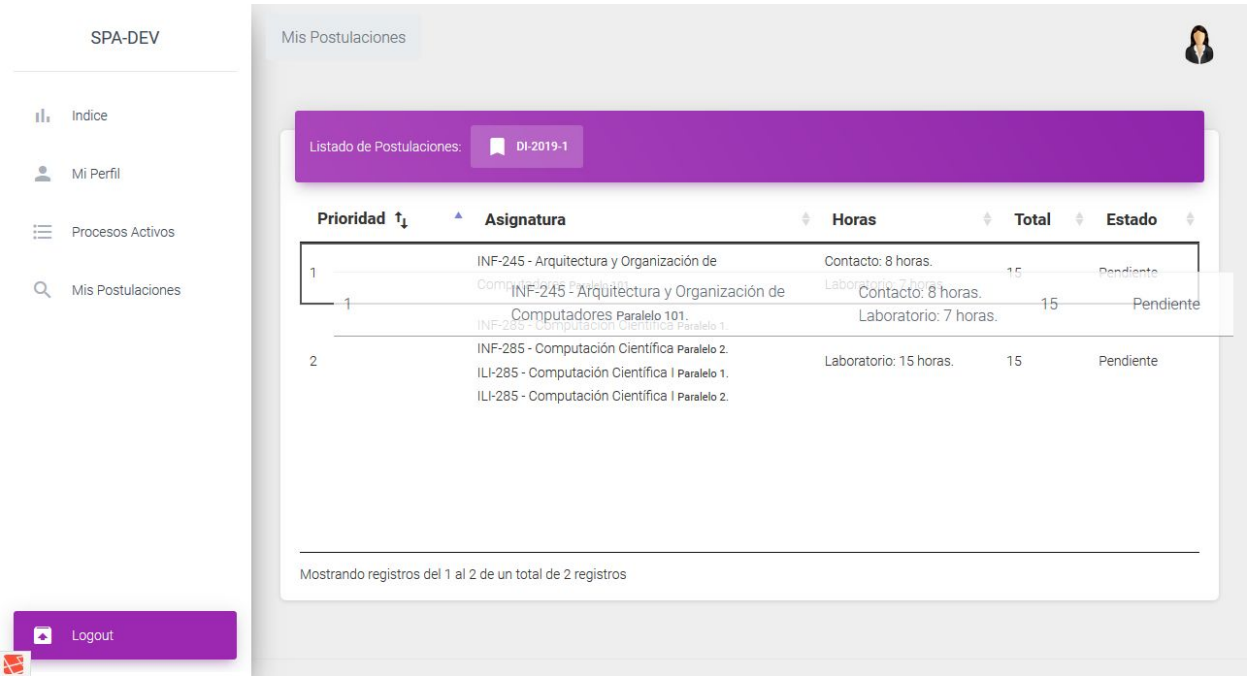

SPA-DEV
Indice
Mi Perfil
Procesos Activos
Mis Postulaciones
Logout
Mi Perfil
Agregar Experiencia
Las experiencias de los departamentos que usen este sistema serán registradas automaticamente.
Año:
Semestre:
Total Horas:
Seleccione Campus/es
(☼) Departamento/s
(☼) Asignatura/s
(☼) Profesor/es
(☼) Si fue ayudante compartido de varias asignaturas/docentes y/o departamentos, separe con comas.
CANCELAR
ACEPTAR
PERIODO
2011-1
AGREGAR EXPERIENCIA
Allison Hernandes Contreras
AYUDANTE
Más informacióon acerca de la persona. Quizás a futuro algo sacado del directorio de personas, o quizás algo editable por los usuarios para darle más personalidad al perfil de cada uno. Por ahora no es prioridad.
ACTUALIZAR BIO

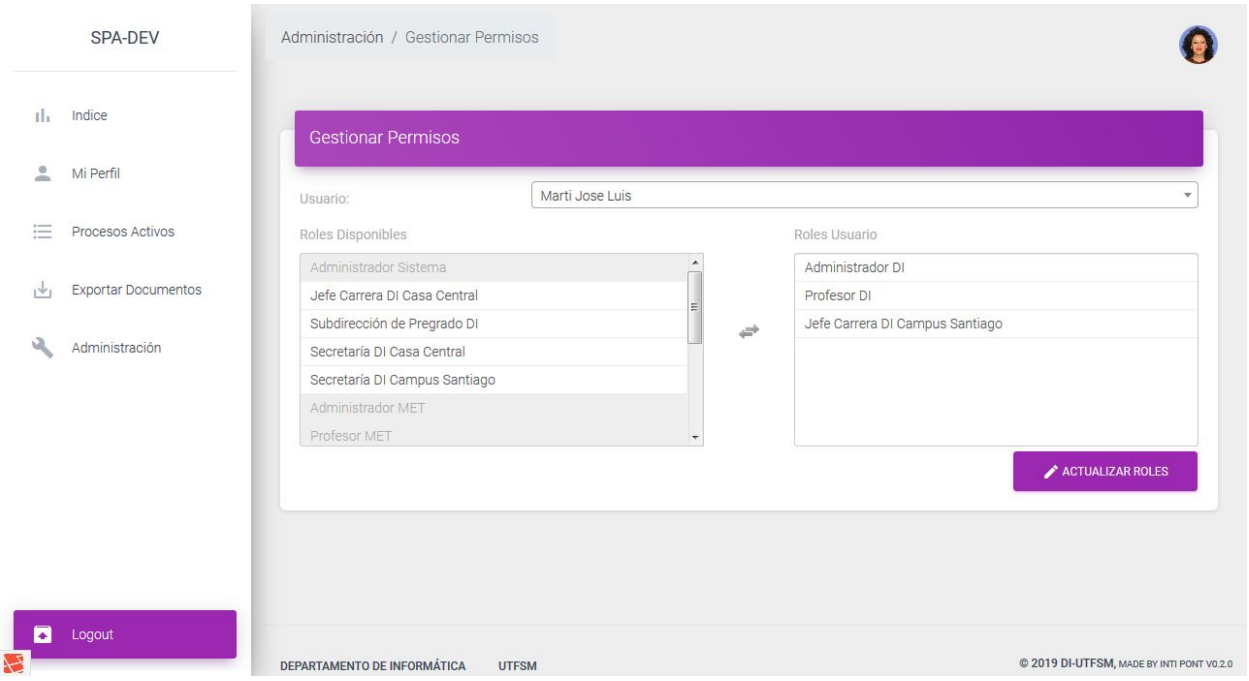

SPA-DEV
Indice
Mi Perfil
Procesos Activos
Mis Postulaciones
Logout
Procesos Activos / DI-2019-2 / Asignaturas
Listado de Asignaturas
Buscar:
Sigla
Nombre
Opciones
ILI-281
Fundamentos de Investigación de Operaciones
PARALELO
PROFESOR
CAMPUS
VACANTES
OPCIONES
2
Mariam Gómez Sánchez
Casa Central
2
ILI-286
Computación Científica II
INF-134
Estructuras de Datos
PARALELO
PROFESOR
CAMPUS
VACANTES
OPCIONES
1
Hubert Hoffmann Nagel
Casa Central
1

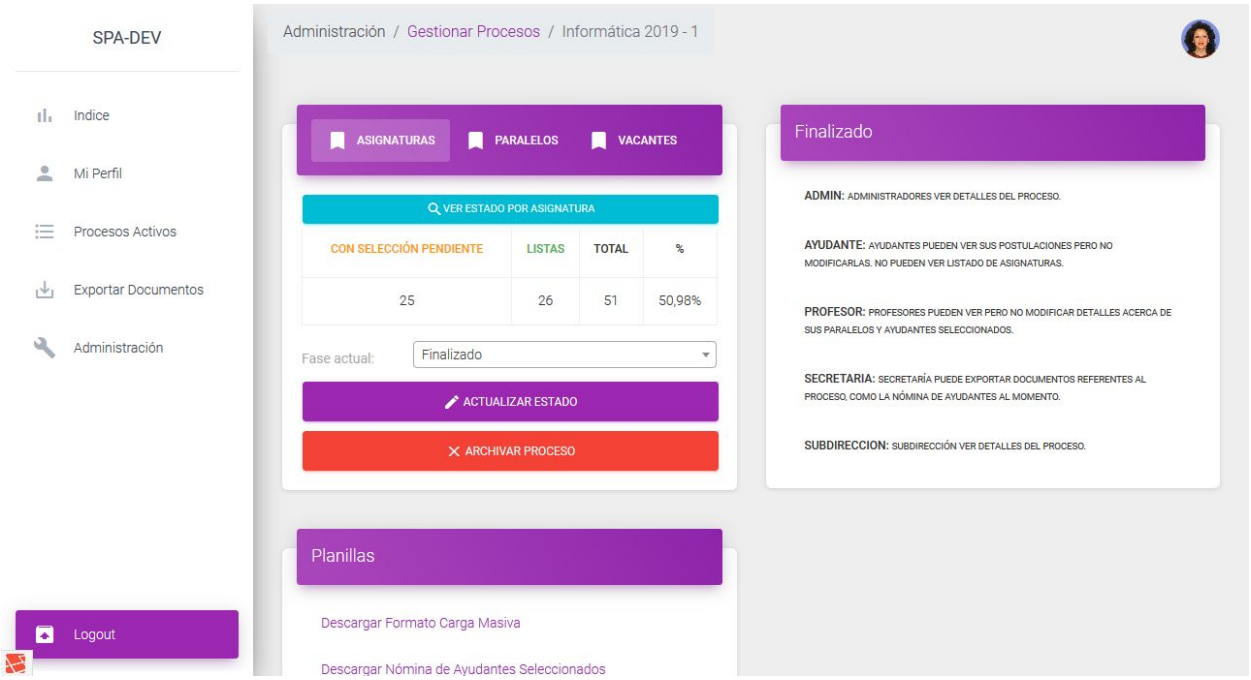

SPA-DEV
Indice
Mi Perfil
Procesos Activos
Exportar Documentos
Administración
Logout
Confirmación
Eliminar Paralelo ?
CANCELAR
CONFIRMAR
Buscar:
Opciones
ICI-309
Seminario de Mem
PARALELO
PROFESOR
CAMPUS
VACANTES
OPCIONES
1
Luis Hevia Rodriguez
Casa Central
0
ICL-001
Laboratorios Santiago
IEI-234
Taller de Análisis de Sistema
ILI-257
Programación Paralela Aplicada
ILI-281
Fundamentos de Investigación de Operaciones

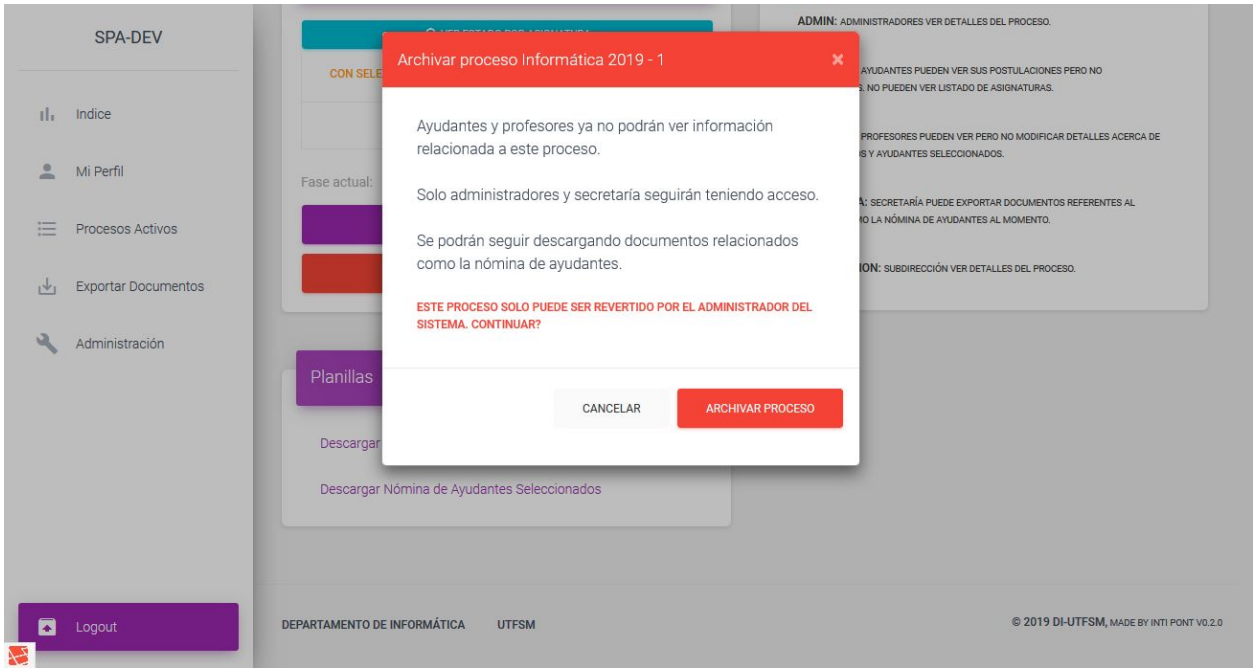

SPA-DEV
Indice
Mi Perfil
Procesos Activos
Mis Postulaciones
Logout
Procesos Activos / DI-2019-1 / Asignaturas / INF-260 CSJ Paralelo 102 / Vacantes
Vacantes INF-260 - Teoría de Sistemas Santiago San Joaquín Paralelo 102
ESTADO
INFORMACIÓN
TOTAL HORAS
OPCIONES
Sin seleccionados
Contacto: 8 horas.
Corrección: 4 horas.
12
+
DEPARTAMENTO DE INFORMÁTICA
UTFSM
© 2019 DI-UTFSM, MADE BY INTI PONT V0.2.0

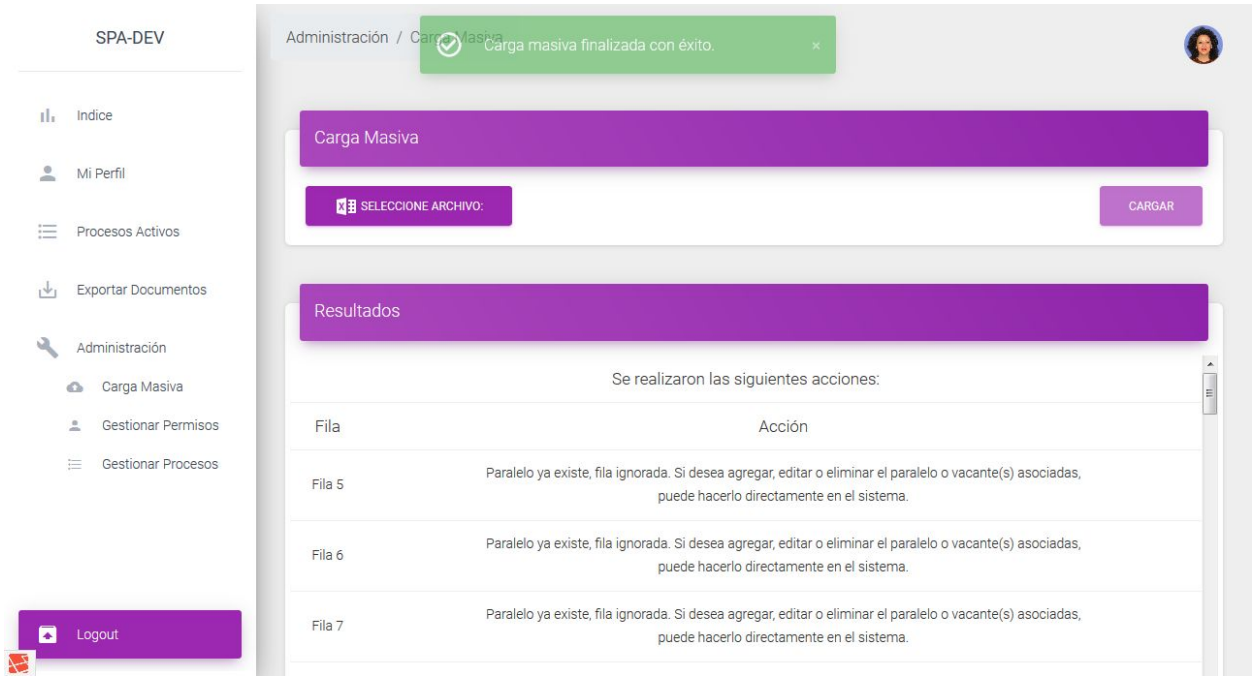

SPA-DEV
Indice
Mi Perfil
Procesos Activos
Exportar Documentos
Administración
Logout
Editar Tipo de Vacante
Cantidad Vacantes
7
Horas Contacto
2
Horas Corrección
6
Horas Laboratorio
7
Vacante está asociada a los siguientes paralelos:
INF-130 CV (Paralelo 201)
CANCELAR
ACEPTAR
+ AGREGAR VACANTE
ESTADO
OPCIONES
Estado
Postulaciones
Aprobada
Aprobada
ROL USM
Ingeniería Civil Industrial
RUT
Casa Central
Prioridad Académica
6534
Cantidad VTR2
2
Promedio Ponderado
49
Cantidad VTR3
3
Seleccionados:
2 de 7
ACTUALIZAR SELECCIÓN


SPA-DEV
Indice
Mi Perfil
Procesos Activos
Mis Postulaciones
Logout
Mis Postulaciones
Listado de Postulaciones:
DI-2019-1
Prioridad
Asignatura
Horas
Total
Estado
1
INF-245 - Arquitectura y Organización de Computadores Paralelo 101.
Contacto: 8 horas.
Laboratorio: 7 horas.
15
Pendiente
2
INF-285 - Computación Científica Paralelo 2.
ILI-285 - Computación Científica I Paralelo 1.
ILI-285 - Computación Científica I Paralelo 2.
Laboratorio: 15 horas.
15
Pendiente
Mostrando registros del 1 al 2 de un total de 2 registros


SPA-DEV
Indice
Mi Perfil
Procesos Activos
Exportar Documentos
Administración
Logout
Administración / Gestionar Permisos
Gestionar Permisos
Usuario:
Marti Jose Luis
Roles Disponibles
Administrador Sistema
Jefe Carrera DI Casa Central
Subdirección de Pregrado DI
Secretaría DI Casa Central
Secretaría DI Campus Santiago
Administrador MET
Profesor MET
Roles Usuario
Administrador DI
Profesor DI
Jefe Carrera DI Campus Santiago
ACTUALIZAR ROLES
DEPARTAMENTO DE INFORMÁTICA
UTFSM
© 2019 DI-UTFSM, MADE BY INTI PONT V0.2.0


SPA-DEV
Indice
Mi Perfil
Procesos Activos
Exportar Documentos
Administración
Logout
Administración / Gestionar Procesos / Informática 2019 - 1
ASIGNATURAS
PARALELOS
VACANTES
VER ESTADO POR ASIGNATURA
CON SELECCIÓN PENDIENTE
LISTAS
TOTAL
%
25
26
51
50,98%
Fase actual:
Finalizado
ACTUALIZAR ESTADO
ARCHIVAR PROCESO
Finalizado
ADMIN: ADMINISTRADORES VER DETALLES DEL PROCESO.
AYUDANTE: AYUDANTES PUEDEN VER SUS POSTULACIONES PERO NO MODIFICARLAS. NO PUEDEN VER LISTADO DE ASIGNATURAS.
PROFESOR: PROFESORES PUEDEN VER PERO NO MODIFICAR DETALLES ACERCA DE SUS PARALELOS Y AYUDANTES SELECCIONADOS.
SECRETARIA: SECRETARÍA PUEDE EXPORTAR DOCUMENTOS REFERENTES AL PROCESO, COMO LA NÓMINA DE AYUDANTES AL MOMENTO.
SUBDIRECCION: SUBDIRECCIÓN VER DETALLES DEL PROCESO.
Planillas
Descargar Formato Carga Masiva
Descargar Nómina de Ayudantes Seleccionados


SPA-DEV
Indice
Mi Perfil
Procesos Activos
Exportar Documentos
Administración
Logout
Archivar proceso Informática 2019 - 1
Ayudantes y profesores ya no podrán ver información relacionada a este proceso.
Solo administradores y secretaría seguirán teniendo acceso.
Se podrán seguir descargando documentos relacionados como la nómina de ayudantes.
ESTE PROCESO SOLO PUEDE SER REVERTIDO POR EL ADMINISTRADOR DEL SISTEMA. CONTINUAR?
CANCELAR
ARCHIVAR PROCESO
Planillas
Descargar Nómina de Ayudantes Seleccionados
DEPARTAMENTO DE INFORMÁTICA
UTFSM


SPA-DEV
Indice
Mi Perfil
Procesos Activos
Exportar Documentos
Administración
Carga Masiva
Gestionar Permisos
Gestionar Procesos
Logout
Administración / Carga Masiva
Carga masiva finalizada con éxito.
Carga Masiva
SELECCIONE ARCHIVO:
CARGAR
Resultados
Se realizaron las siguientes acciones:
Fila
Acción
Fila 5
Paralelo ya existe, fila ignorada. Si desea agregar, editar o eliminar el paralelo o vacante(s) asociadas, puede hacerlo directamente en el sistema.
Fila 6
Paralelo ya existe, fila ignorada. Si desea agregar, editar o eliminar el paralelo o vacante(s) asociadas, puede hacerlo directamente en el sistema.
Fila 7
Paralelo ya existe, fila ignorada. Si desea agregar, editar o eliminar el paralelo o vacante(s) asociadas, puede hacerlo directamente en el sistema.

BIBLIOGRAFIA

- Deacon, J. (1995), Model-View-Controller (MVC) Architecture.
- Martin, R. (2000), *Design Principles and Design Patterns*.
- Urso, A. (2012), *Desarrollo e Implementación de la nueva versión del Sistema de Postulación y Selección de Ayudantes Docentes del DI*.UTFSM, Chile.
- Muñoz, M. (2016). *Modelado de Procesos de la Subdirección de Pregrado del Departamento de Informática de la UTFSM*. Memoria para optar al título de Ingeniero Civil en Informática, UTFSM, Chile.
- Zuñiga, V. (2018), *Plan de Calidad de Software para el Sistema Intranet de una Unidad Académica.* UTFSM, Chile
- https://laravel.com/docs/6.x
- https://laravel.com/docs/6.x/dusk
- https://laravel.com/api/6.x/
- https://docs.laravel-excel.com/
- https://phpspreadsheet.readthedocs.io/en/latest/
- https://www.php.net/manual/en/