ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

Aplicación para la Gestión de Archivos de Titulación Agata

PROYECTO INTEGRADOR

Previo la obtención del Título de:

Ingeniero en Computación

Presentado por:

Allison Brito

Erick Pulla

GUAYAQUIL - ECUADOR

Año: 2021

DEDICATORIA

Esta tesis va dedicada a toda mi familia, en especial a mi madre Irma y a mi hermana Arlette quienes me aconsejaron y lograron darme ánimos para culminar este proyecto, así como lograr culminar mi carrera universitaria.
En segundo lugar, a mi abuela Teresa y mi abuelo Enrique quienes no lograron en vida verme alcanzar mi objetivo, pero sé que están orgullosos desde el cielo por lograr esta meta.
Y por último a todos aquellos que me acompañaron durante toda mi etapa universitaria, sé que sin todos ustedes el camino hubiese sido más complicado.

Allison Brito Mendoza

DEDICATORIA

Esta tesis se la dedico a mi familia, especialmente a mi madre Dolores, mis abuelos José y Consuelo, mi hermano Gary, mi prima Milenky y mis mascotas Marty y Billy, quienes han estado conmigo siempre en todo desde que inicié la universidad.

También dedico la tesis a mis compañeros de universidad con los cuales siempre pasamos ayudándonos en los trabajos grupales.

Finalmente, dedico esta tesis a todas las personas que conocí a lo largo de mi carrera universitaria.

Erick Pulla Zambrano

AGRADECIMIENTOS

Agradezco en primer lugar a Dios porque sin su bendición y ayuda no hubiera logrado culminar este proyecto.
Agradezco a mi familia por la paciencia y por todas esas palabras de aliento cuando desfallaba.
Y por último a aquellos docentes que brindaron su tiempo y espacio para conmigo cuando necesité de alguna ayuda y/o sugerencia a lo largo de mi carrera universitaria.
Infinitas gracias, esto es para ustedes.

Allison Brito Mendoza

AGRADECIMIENTOS

Agradezco sobre todo a mi madre Dolores y a mi hermano Gary, quienes, gracias a su sabiduría, pude superar los retos tantos académicos como personales en esta gran etapa.

Agradezco a mis amigos César, Josue, Ricardo, Camilo y Harrison, quienes siempre me escucharon cuando más lo necesitaba.

Finalmente, agradezco a todos los profesores que he conocido a lo largo de esta travesía. Todos aportaron un granito de arena a mi conocimiento y estaré agradecido infinitamente por ello.

Erick Pulla

DECLARACIÓN EXPRESA

“Los derechos de titularidad y explotación, me(nos) corresponde conforme al reglamento de propiedad intelectual de la institución; Allison Brito Mendoza, Erick Pulla Zambrano y damos nuestro consentimiento para que la ESPOL realice la comunicación pública de la obra por cualquier medio con el fin de promover la consulta, difusión y uso público de la producción intelectual"

Allison Brito Mendoza

Erick Pulla Zambrano

EVALUADORES

LUCIA MARISOL VILLACRES FALCONI
Digitally signed by LUCIA MARISOL VILLACRES FALCONI
Date: 2022.03.18 13:13:26 -05'00'

Lucía Villacres, PhD.

PROFESOR DE LA MATERIA

Firmado electrónicamente por:
DAVID ALONSO JURADO MOSQUERA

David Alonso Jurado Mosquera, MSc.

PROFESOR TUTOR

RESUMEN

El proceso de titulación de estudiantes de ESPOL es manejado de forma informal, descentralizada y poco organizada. Por esta razón, la universidad crea un sistema que busca automatizar este proceso llamado AGATA, sin embargo, éste aún continua en etapa de desarrollo con inconsistencias en los requerimientos solicitados por GTSI.

El presente proyecto buscó modificar el sistema AGATA de tal forma que pueda ser desplegado en ambiente de producción de una forma que cumpla con las políticas existentes de ESPOL y requerimientos de GTSI, donde se integraron distintos servicios internos desarrollados por el mismo GTSI que permitieron acceder a la información necesaria para automatizar distintos procesos aun manuales obtenida con el framework Django y almacenada en una base de PostgreSQL, para luego ser desplegado en dos servidores LINUX.

Como resultado de este desarrollo, se tiene al sistema AGATA desplegado en los dos servidores LINUX mencionados utilizando una arquitectura descentralizada, donde un servidor posee el frontend desarrollado con React, y el otro posee el backend desarrollado con Django junto a la base de datos PostgreSQL. El backend permite conectarse con servicios internos proporcionados por el GTSI y externos como el Dspace y SAAC para centralizar el proceso de titulación en este sistema por medio de APIs. El frontend llama a las APIs del backend y, a través de un servidor de tunneling para evadir las políticas de CORS impuestas en los navegadores web modernos, recibir y muestra toda la información en su interfaz dinámica.

Palabras Clave: Titulación, Linux, DSpace, SAAC, Gestión Documental.

ABSTRACT

The graduation process for ESPOL students is managed in an informal, decentralized and poorly organized manner. For this reason, the university creates a system that seeks to automate this process called AGATA, however, it is still in the development stage with inconsistencies in the requirements requested by GTSI.

The present project sought to modify the AGATA system in such a way that it can be deployed in a production environment in a way that complies with existing ESPOL policies and GTSI requirements, where different internal services developed by the GTSI itself were integrated, which allowed access to the necessary information to automate different even manual processes obtained with the Django framework and stored in a PostgreSQL base, to later be deployed on two LINUX servers.

As a result of this development, the AGATA system is deployed on the two mentioned LINUX servers using a decentralized architecture, where one server has the frontend developed with React, and the other has the backend developed with Django together with the PostgreSQL database. The backend allows connecting with internal services provided by the GTSI and external services such as Dspace and SAAC to centralize the titling process in this system through APIs. The frontend calls the APIs of the backend and, through a tunneling server to avoid the CORS policies imposed on modern web browsers, receive and display all the information in its dynamic interface.

Keywords: Degree, Linux, DSpace, SAAC, Document Management.

ÍNDICE GENERAL

ABREVIATURAS

ESPOL	Escuela Superior Politécnica del Litoral
FIEC	Facultad de Ingeniería en Electricidad y Computación
GTSI	Gerencia de Tecnologías y Sistemas de Información
API	Application Programming Interface
SMTP	Protocolo para transferencia simple de correo

SIMBOLOGÍA

s	Segundo

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

CAPÍTULO 1

1. INTRODUCCIÓN

1.1 Descripción del problema

Según (Gonzalez,2019), el proceso de titulación de una tesis es complejo y se ven involucrados varias entidades para realizar la aprobación. En la Escuela de Ingeniería Mecánica Industrial de la Universidad de San Carlos de Guatemala ocurre este hecho, lo que le provoca un esfuerzo adicional al personal. Lo mismo ocurre en la Universidad Estatal del Sur de Manabí, para tener el control del proceso de sus estudiantes sobre la titulación es primordial y buscan mejorar usando herramientas tecnológicas (Salazar,2020) por tanto, no es extraño que en la Escuela Superior Politécnica del Litoral (ESPOL) ocurra lo mismo.

Actualmente, en la Facultad de Ingeniería en Electricidad y Computación (FIEC), se maneja el proceso de registro de los trabajos de titulación a través de servicios que se encuentran en la nube como One Drive y Google Drive. Sin embargo, utilizar diferentes espacios de almacenamiento produce inconsistencia en el manejo de la información.

Adicionalmente, al momento en que se realiza la revisión de estos documentos por parte de los encargados: tutor, revisor y personal administrativo, se evidencia un alto grado de desorganización y retraso en la gestión de los documentos, provocando retrasos no planificados según el cronograma.

Por otra parte, el proceso de titulación carece de un registro formal en cada una de sus fases; esto es, el manejo de la información por parte de los cinco actores del proceso: el estudiante, el tutor del proyecto, el docente, la secretaría y el subdecanato. Para iniciar el proceso, la secretaría crea en la nube una carpeta que es compartida con el estudiante, el tutor y el docente, para las respectivas revisiones y retroalimentaciones. Si el tutor emite alguna retroalimentación al trabajo que subió el estudiante, este lo corrige y dicho proceso se

repite hasta que el tutor emita su aprobación para que continúe el proceso. Como siguiente paso, se tiene a la revisión del docente, quien es el que emite la aprobación del proyecto.

Es importante mencionar que existen tres tipos de proyectos: integrador, multidisciplinario e investigación, por lo que se verifica qué tipo de documento se subió a la carpeta compartida; el proyecto de investigación consta de un documento adicional llamado paper de investigación a diferencia del proyecto integrador.

Por último, la secretaría descarga el documento final, genera el documento de acta de grado que debe enviarse al subdecanato para su firma y este lo devuelve a secretaría. La secretaría sube el acta de grado firmada y el documento del proyecto dependiendo del tipo de proyecto que sea.

Este proyecto se divide en dos fases, la primera realiza el seguimiento a cada estudiante y le indica en que etapa se encuentra dentro del proceso de titulación. En el proceso intervienen la secretaría, tutores y profesores de la materia integradora que pueden realizar observaciones que se envían al estudiante para que realice las correcciones necesarias al documento entregado como un trabajo integrador o artículo de investigación; además, el estudiante también puede subir su hoja de vida. Se envían notificaciones a cada uno de los usuarios para dar a conocer en qué etapa se encuentra el proceso. (Aguirre y Pincay, 2021)

En la segunda fase, se tiene una API (Application Programming Interface) que administra la plataforma y accede a servicios adicionales como información relacionada a estudiantes, tutores y profesores por parte de la secretaría. La secretaria es la encargada de revisar la hoja de vida que sube el estudiante, extraer el acta de grado desde el SAAC para enviarla al subdecanato y se proceda a la firma correspondiente, además de subir al DSpace el documento del proyecto. Por último, todos estos datos se envían al STA para que se emita el título de grado correspondiente. (Espinoza y Tigse, 2021)

Aplicación para la Gestión de Archivos de Titulación Agata, como es denominado este sistema, no está implementado en ambiente de

producción en ninguna de sus dos fases debido a que se está enfocado en la investigación del problema y en el diseño de la aplicación. Por otro lado, la API necesaria para automatizar todo el proceso se encuentra en etapa de desarrollo y pruebas, dicha API está a cargo de la Gerencia de Tecnologías y Sistemas de Información (GTSI), por lo tanto, no se encuentra integrada dentro de la plataforma en desarrollo.

Finalmente, el sistema AGATA se encuentra operando actualmente en un servidor de pruebas de manera integrada (FRONTEND y BACKEND), producto que no cumple con las políticas y los requerimientos del GTSI. Adicionalmente, el software actual no automatiza los procesos con los sistemas SAAC y DSpace, por lo que se debe realizar la carga de los documentos de forma manual en cada plataforma, es decir, requiere que se siga haciendo uso de varios sistemas para realizar cada paso del proceso de titulación.

1.2 Justificación del problema

En la actualidad, es muy común que una empresa tenga una aplicación web implementada ya que automatiza los procesos internos y/o externos dentro de la misma, lo que permite digitalizar su información además de mantener la comunicación eficiente entre sus distintas áreas. (Matute y Ávila, 2020)

Según Brown & Keller (2006), con la automatización de los procesos de gestión, se reduce en más de un 30% el tiempo invertido para el manejo de información por parte del personal administrativo. En este sentido, implementar un software que permita mantener la información a la mano de forma inmediata, permite reducir el tiempo en la gestión documental, tanto para administradores y estudiantes. Por lo tanto, para la FIEC es sumamente importante la implementación de este sistema ya que posibilita una reducción el porcentaje de error de procesamiento de datos y optimiza el manejo de la información.

Se propone una solución al problema descrito que consiste en desagregar una aplicación web existente, es decir, separar la

aplicación que actualmente funciona conjuntamente en dos componentes. Se desea tener un frontend y backend, logrando que a futuro se pueda reutilizar este último para integrarlo con sistemas ya existentes o inclusive nuevas interfaces. Todo esto para lograr unificar el proceso de titulación donde los distintos actores puedan verificar y monitorear el estado del proceso de titulación de cada estudiante.

Cabe resaltar que la aplicación web que se ofrece va a permitir retroalimentar a los usuarios sobre el estado del proceso a través de notificaciones desde la aplicación y por correo electrónico.

1.3 Objetivos

1.3.1 Objetivo General

- Desplegar el sistema AGATA en ambiente de producción cumpliendo determinadas políticas existentes de ESPOL e integrando distintos servicios desarrollados por el GTSI.

1.3.2 Objetivos Específicos

1. Separar la aplicación AGATA en distintos servidores
2. Consumir los servicios SAAC y DSpace para carga y descarga de actas de grado y proyectos de titulación e investigación respectivamente
3. Realizar mejoras al sistema administrativo existente como:
    - Implementar el envío de correo electrónico a cada usuario sobre las etapas en las que se encuentra el proyecto
    - Generar reportes más detallados de cada uno de los procesos de titulación.

1.4 Alcance del proyecto

En este trabajo de titulación se realizan algunas mejoras al sistema AGATA y su puesta en producción, para que gestione el manejo de la documentación de cada proceso de titulación de los estudiantes de la FIEC. El proyecto consiste en habilitar al sistema AGATA para que se ejecute en

ambiente de producción en dos servidores distintos (BACKEND y FRONTEND respectivamente), consumiendo adecuadamente las API's requeridas: CAS, DSpace, SAAC y otros servicios de ESPOL provistos por el GTSI.

El sistema AGATA permite que el estudiante suba documentación relacionada a su proceso de titulación como tesis, hoja de vida, artículos de investigación, entre otros. Por su parte, los docentes, tutores y secretarias deben validar la información a través de las revisiones, retroalimentaciones y aprobaciones periódicas según la agenda establecida. Además, los actores del proceso de titulación reciben notificaciones a través del sistema y el correo electrónico relacionados con el avance del proceso. Finalmente, ofrece un manejo más ordenado y consistente del proceso de titulación por parte de todos los actores, permitiendo que el personal administrativo (la secretaría y el subdecanato) puedan saber en qué etapa se encuentra dicho proceso.

1.5 Limitaciones del proyecto

- El sistema AGATA es un proyecto piloto que en esta fase sólo funcionará en la FIEC para efectos de verificar y comprobar su eficiencia y eficacia, lo que implica que por el momento no podrá generalizarse en el resto de las facultades.
- Para que el proceso sea más sencillo y automatizado se requiere tener acceso a los servicios proporcionados por el GTSI para comunicarse con los otros sistemas involucrados en el proceso de titulación, como el SAAC y el DSPACE.
- El acceso a los servidores de producción y prueba serán proporcionados por el GTSI.

1.6 Marco teórico

Dentro de los sistemas web, es importante conocer ciertos conceptos y ventajas que se mencionan a continuación como el protocolo HTTP, que nos permite realizar la comunicación por medio de la red; Reactjs, una librería desarrollada para crear interfaces de usuario dinámicas; es necesario conocer sobre el término Backend, debido a que contiene la lógica de todo el sistema y del acceso

a la información de la base de datos. Por último, el término API restful que se refiere a la forma de comunicación entre la interfaz de usuario y el backend.
Consideramos estos términos como los más destacables de este proyecto y se realiza una revisión de trabajos que han utilizado los componentes que se escogieron. Se tiene como proyectos similares a otros trabajos que implementan APIs Restful utilizando APIS de terceros y el uso de la misma arquitectura en el desarrollo de dichos proyectos.

1.6.1 Conceptos claves de desarrollo web

HTTP

HTTP (Hypertext Transfer Protocol) es un protocolo de aplicación para comunicaciones a través de una red. Actualmente, HTTP es el principal medio de comunicación en la red mundial o World Wide Web (Arcuri, 2019). Por lo que este protocolo nos permitirá conectarnos mediante peticiones (requests methods) entre nuestra interfaz gráfica y backend. Entre los métodos, destacan los siguientes que más se usarán en el proyecto como:

- GET: Se utiliza para obtener datos. Este método nos brindará información sobre el estudiante, listado de usuarios, listados de proyectos entre otros datos.
- POST: Utilizado principalmente para enviar registros o datos a un recurso determinado, agregando cambios al servidor. Este método nos permite enviar los registros de aprobación entre las fases del proceso de titulación, así como también el envío de los documentos entre los actores del proyecto.
- PUT: Este método reemplaza una representación actual de un recurso en el servidor con datos que se envían como petición. Será usado para actualizar la deuda de un estudiante que debe subir su trabajo final al DSpace.

Frontend

Según (Abdullah & Zeki, 2014), frontend se define como la parte en donde el usuario puede ver e interactuar los elementos de una interfaz gráfica. El desarrollo de esta interfaz se realiza mediante el uso de los lenguajes HTML, CSS y JavaScript, y se facilita con el uso de librerías como

ReactJS. El frontend se encarga de consumir las APIs proporcionadas por sistemas, como lo es el backend, de un servidor en la nube. Este servidor puede ser el mismo del backend, o uno distinto. Los datos obtenidos a través de las APIs se presentan en la interfaz web para el usuario, donde éste es capaz de manipular los datos a través de la conexión que tiene el frontend con el backend, según la interfaz lo permita.

Backend

Consta generalmente de tres partes: un servidor, una aplicación y una base de datos. El Backend funciona como un framework de desarrollo de APIs capaces de consumir servicios externos de cualquier sistema, como, por ejemplo, el GTSI, y, a su vez, ser consumidas por cualquier otro sistema en el mismo servidor o en uno diferente, siendo en este caso, la interfaz web de AGATA (Pérez y Cancio, 2014). En este proyecto se utiliza como framework a Django junto a una base de datos de PostgreSQL.

API RESTful y la conexión entre sistemas

Una API RESTful (Aplication Programming Interface Representational State Transfer) se describe como una interfaz de programación de aplicaciones o conjuntos de definiciones y protocolos que se utilizan para diseñar e integrar el software de las aplicaciones.

REST se refiere al conjunto de pautas arquitectónicas sobre la creación de servicios web encima de HTTP. Actualmente, las aplicaciones cliente-servidor deben satisfacer algunas restricciones para ser consideradas RESTful, como no tener estado, y los recursos deben indicar explícitamente si se pueden almacenar en caché. La información enviada al cliente es independiente del formato real del recurso o no y pueden estar representadas en formato JSON o XML (Arcuri, 2019).

Los servicios web RESTful son una forma de proporcionar interoperabilidad entre sistemas informáticos en la nube y permiten que los sistemas solicitantes acceder y manipular representaciones textuales de recursos web utilizando un conjunto uniforme y predefinido de operaciones sin estado (Chen, Ji, Fan & Zhan, 2017).

Gracias a estos servicios, las partes frontend y backend pueden comunicarse y compartir recursos e información mientras conservan la

seguridad, el control y la autenticación de los datos según los roles de cada usuario.

1.6.2 Proyectos similares

1. Implementación de API Restful para uso en APP de ofertas (CHAPLIST), desarrollada con Ionic. Es una plataforma móvil híbrida para la publicación de ofertas de supermercados como LaTorre. Para este proyecto, el uso de API Restful es similar ya que nos permite realizar la comunicación de la interfaz de usuario con el backend. Cada API desarrollada posee un identificador llamado endpoint y a su vez contiene un método descrito anteriormente que nos indica la acción que se realiza. Utiliza una arquitectura cliente-servidor en donde el frontend es el encargado de realizar las peticiones al backend, logrando que se tenga un modelo centralizado (Romero,2015) como ocurre con AGATA. Adicionalmente, también consumen API's de terceros como la de Facebook y Google (Vivas y Ubeda, 2016) que en nuestro caso serían las APIs del GTSI que proporcionan información sobre deuda del estudiante, información general del estudiante, proyecto realizado, entre otros.
2. Comparador de precios de los servicios de taxi. Aplicación móvil desarrollada en Android y backend desarrollado en Django. Muestra los precios de cada una de las aplicaciones de servicio de transporte como Cabify, Uber, Emov, MyTaxi, Car2go y compara según la tarifa proporcionada por los servicios de transporte. Este sistema consume diversas APIs contenidas en su framework de Backend (Django) para realizar consultas a su propia base de datos y consume APIs de servicios externos pertenecientes a las empresas de transporte mencionadas para obtener los precios y compararlos (Jiang, 2017). Respecto a nuestro proyecto, las APIS desarrolladas en Django consumen a su vez a las APIs del GTSI, es decir, en el backend se manejan los request hacia las APIs externas, en donde su principal interés

es almacenar la información de los estudiantes en general, así como información de su proyecto integrador logrando que se guarde en la base de datos de AGATA y evitar realizar consultas a su servidor cada que un usuario inicie sesión.

3. Desarrollo de una plataforma IoT para la gestión del riego de precisión. Este proyecto utiliza la misma arquitectura que AGATA con la diferencia de que almacena la información usando el motor de base de datos MySQL y utiliza sensores para enviar sus datos por medio de API Rest. Al igual que AGATA en el backend tienen la lógica de su aplicación utilizando el Modelo-Vista-Controlador. Del lado del frontend utilizan a Reactjs para conectarse por medio de las APIs al backend. El uso de estas herramientas hace que el proyecto pueda ser escalable. (Puig, Rodriguez y Gonzalez, 2021)

CAPÍTULO 2

En este capítulo se presentan los cambios y mejoras **que el cliente solicitó para** el sistema AGATA, tanto en su modelo lógico de base de datos como en su interfaz web de usuario. inicialmente, se describe el modelo de gestión de proyecto a seguir y el cronograma de planificación de actividades semanales. luego, se analizan los requerimientos adicionales del proyecto y se describen l**as decisiones de diseño que se tomaron para** satisfacerlos. finalmente, se muestran diagramas UML **y pantallas prototipando los cambios solicitados** con el fin de ilustrar **el proceso de diseño planificado para este** del proyecto.

2. METODOLOGÍA

2.1 Metodología del proyecto

Para el desarrollo de las modificaciones y mejoras del sistema AGATA, se decide utilizar el modelo de gestión incremental, el cual tiene como propósito, el desarrollo progresivo de un software, permitiendo realizar entregas más completas hasta conseguir la versión final donde los requerimientos del cliente han sido satisfechos en su totalidad (Yacelga, Espinoza y Vásquez, 2021).

2.2 Análisis de requerimientos

Para lograr definir los requerimientos en este proyecto, se realizaron pruebas con usuarios finales como estudiantes, secretaría, profesores y tutores. Cada uno de estos usuarios dieron retroalimentación luego de utilizar el sistema. Cada usuario de los distintos roles llenó un formulario proporcionado por la secretaría para ir recolectando información de usabilidad y funcionalidad.

2.2.1 Requerimientos funcionales

Dado que el sistema AGATA ya está desarrollado en casi su totalidad, entre los requerimientos que se necesita y basado en la retroalimentación de los involucrados en el proyecto. Se categorizan a los requerimientos funcionales en globales, estudiante, administrador, secretaría. Se llaman globales a aquellos requerimientos que afectan a cada uno de los roles que interactúan con el sistema AGATA. Los requerimientos funcionales sobre estudiante son modificaciones para realizar en el módulo del estudiante, así como, se tienen

requerimientos funcionales para secretaría y administrador, se realizan modificaciones para el módulo de secretaría y al administrador del sistema AGATA, respectivamente. El listado de requerimientos se resume en lo siguiente:

2.2.1.1 Requerimientos Globales

- Envío de correos electrónicos para: 1) comunicar las notificaciones automáticas generadas por el sistema, y 2) contactar a los estudiantes pendientes de realizar el proceso de titulación.
- Agregar un combobox para seleccionar el término académico y cambiar la ubicación, icono, etiqueta y botón de cerrar sesión de la interfaz dentro del sistema AGATA en todos los módulos.
- Permitir seleccionar el término académico a consultar.
- Actualizar en el modelo lógico de la base de datos del sistema AGATA para habilitar y deshabilitar usuarios en los distintos términos académicos.

2.2.1.2 Requerimientos en el módulo de Estudiante

- Agregar y eliminar archivos en la sección entregables

2.2.1.3 Requerimientos en el módulo de Secretaría

- Descargar un reporte en el módulo de secretaría con información general y específica como fecha y hora de cada etapa realizada y el número de veces en que los documentos fueron devueltos incluyendo a los usuarios por término de todos los proyectos.
- Visualizar proyectos por medio de filtros previamente determinados.
- Carga del documento final al DSpace.
- Carga y descarga del acta generada en el SAAC.
- Modificar los filtros existentes para no perder la búsqueda cada vez que se retrocede a la página anterior en la reportería.

2.2.1.4 Requerimientos en el módulo del Administrador

- Ocultar campos innecesarios en la información mostrada en paginación.
- Corregir los filtros de paginación en las distintas ventanas.

2.2.2 Requerimientos no funcionales

Actualmente, el sistema AGATA se encuentra desplegado en un solo servidor por tal motivo no cumple las políticas del GTSI en el que indican que necesita tener separado el frontend del backend en servidores distintos. Con el objetivo

de optimizar el modularidad y escalabilidad del sistema, se tienen los siguientes requerimientos no funcionales:

- Separar en dos servidores, uno para frontend y otro para backend.
- Corregir errores ortográficos dentro del sistema AGATA.

2.3 Aspectos técnicos y legales

- La propiedad intelectual del sistema AGATA y los derechos de uso de servicios internos y externos pertenecen exclusivamente a la ESPOL.
- La información proporcionada por el GTSI sobre los estudiantes mediante sus servicios internos y externos será empleada únicamente con el fin de realizar pruebas para el desarrollo del Sistema.
- Los datos generados en la base de datos del Sistema AGATA son confidenciales y no se divulgarán por ningún motivo a personal externo al GTSI.

2.4 Cronograma de actividades y reuniones

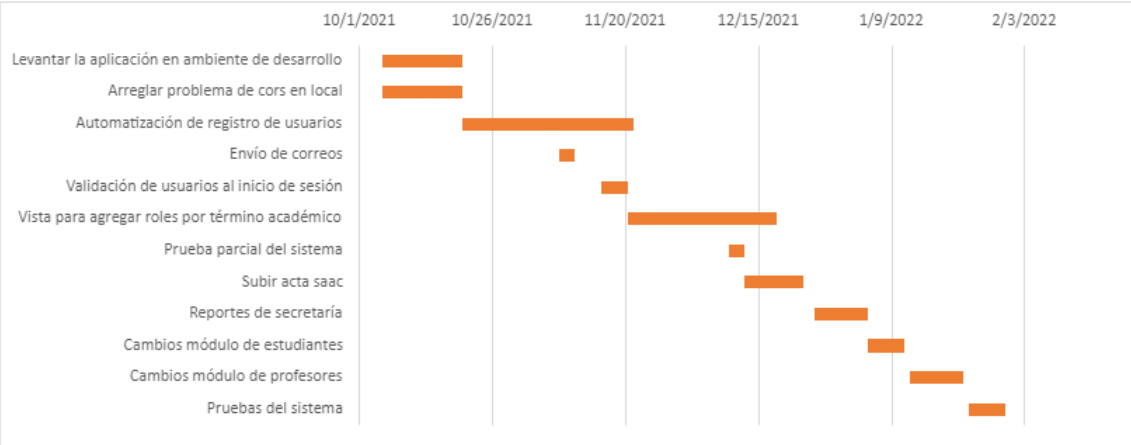

Figura 1 - Cronograma de actividades

En nuestra Figura 1, se planificó tareas semana a semana. También tenemos reuniones semanales mostrando avances de los requerimientos y tareas asignadas.

2.5 Propuesta de despliegue del sistema

Para el despliegue del sistema AGATA, se propone la separación de los apartados Backend y Frontend en diferentes servidores como se muestra a continuación:

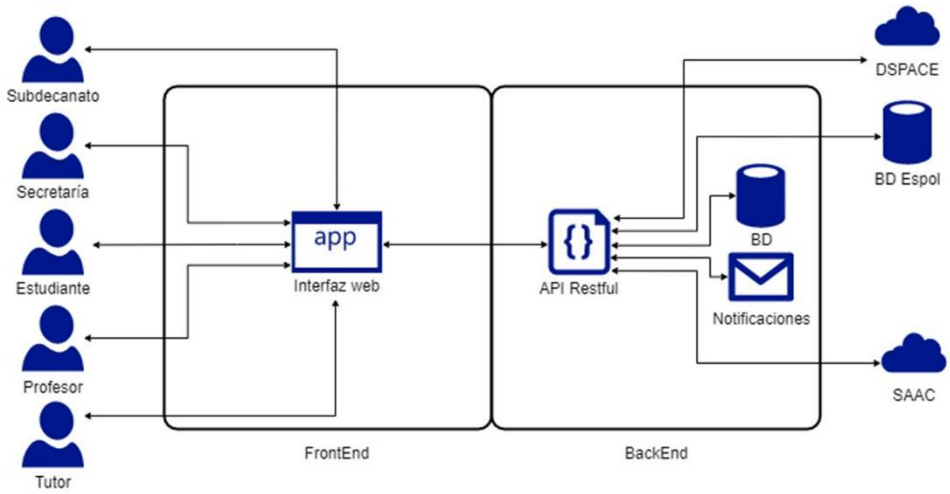

Figura 2 - Modelo de despliegue del sistema AGATA

El apartado Frontend será constituido por la interfaz web como se muestra en la **¡Error! No se encuentra el origen de la referencia.**, donde los usuarios i nteractúan con la aplicación desde un navegador. Por otro lado, el Backend se constituye de la aplicación de Django, que proporciona el servicio de APIs que el frontend consumirá. Aparte de manejar la base de datos interna del sistema, el Backend también será el encargado de manejar y encapsular las APIs de los servicios externos, incluyendo las desarrolladas por el GTSI.

2.6 Diagramas

A continuación, se observa en la **Figura 3** la secuencia que describe la interacción a realizar en este proyecto por parte del actor Secretaría por medio de la interfaz web. Cuando la Secretaría requiere filtrar la información de proyectos y estudiantes se necesita que los filtros descritos se mantengan a lo largo de la secuencia para mantener la búsqueda y evitar perder el registro actual cada vez que regresa a buscar más proyectos. Esta información se mantiene con la ayuda de la caché de los navegadores web.

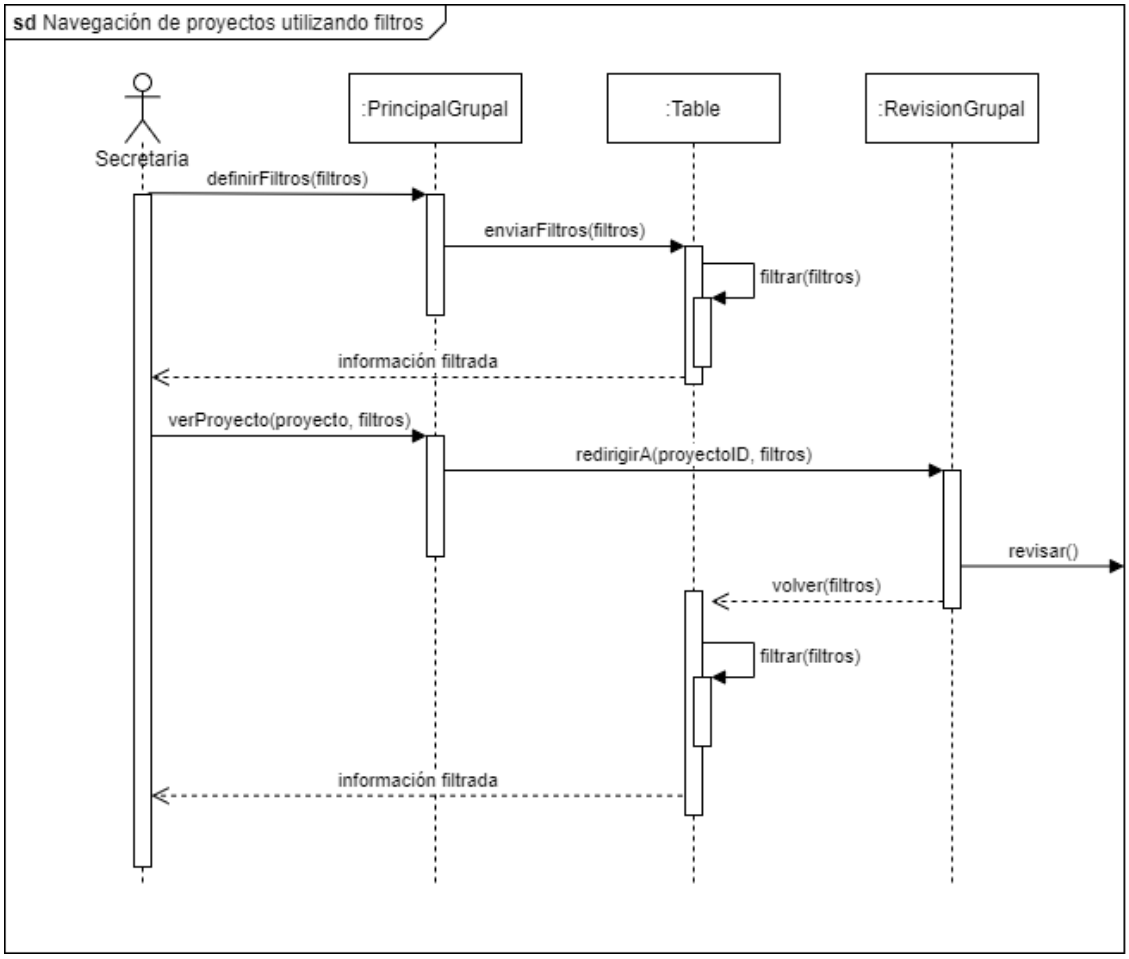

Figura 3 - Diagrama de secuencia de la navegación a implementar del usuario Secretaría filtrando la búsqueda

En la Figura 4**Figura 3** y Figura 5, se muestra el registro automatizado de la información del estudiante, tutor, profesor, materia integradora, carrera, facultad y proyecto cuando un usuario no se encuentra dentro del sistema AGATA. La lógica de esta secuencia se maneja mayoritariamente en el backend, utilizando el API “api/token”, el cual va ligado al componente AuthenticationCAS, donde, después de las verificaciones del servicio externo CAS, se comprueba si el usuario ha sido registrado como estudiante en el sistema AGATA. Si no ha sido registrado, comienza la secuencia donde se piden datos del estudiante a los servicios del GTSI para registrar al usuario. Con los datos devueltos, en donde se incluyen datos de facultad, carrera, materia integradora, proyecto integrador, periodo académico, carrera, coautores, profesor y tutor, se realiza la verificación de estos en la base de datos de AGATA. Si no se encuentran en la base de datos, se los guarda automáticamente.

El proceso completo dura aproximadamente 15 segundos en ejecutarse, y por ese motivo, se genera una pantalla de carga en la interfaz web donde espera hasta que el servicio de registro automático termine su ejecución y devuelva el token de autenticación.

Con este proceso automatizado, se evitan llamadas excesivas a servicios externos a AGATA, específicamente, los pertenecientes al GTSI y, a pesar de duplicar cierta información existente en la base de datos de ESPOL, no se perjudica la eficiencia del sistema ya que se maximizan las llamadas a su propia base de datos.

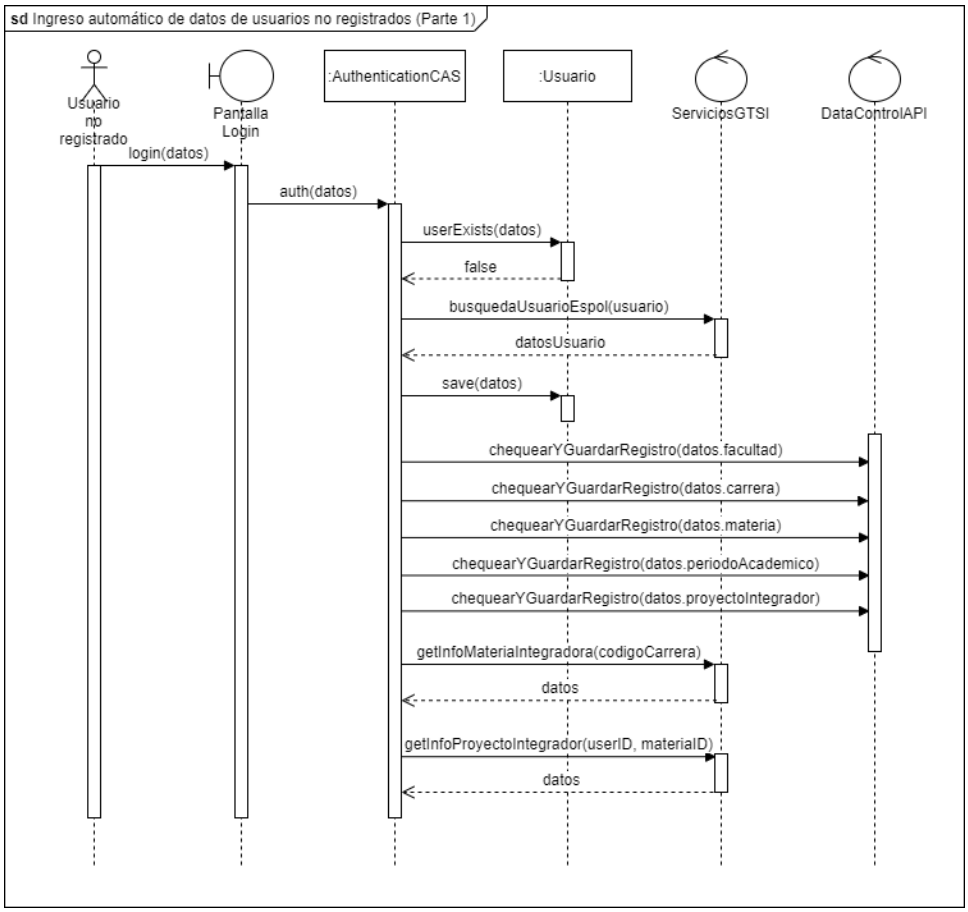

Figura 4 - Diagrama de secuencia del ingreso de datos automáticos cuando entra un usuario no registrado (Parte 1)

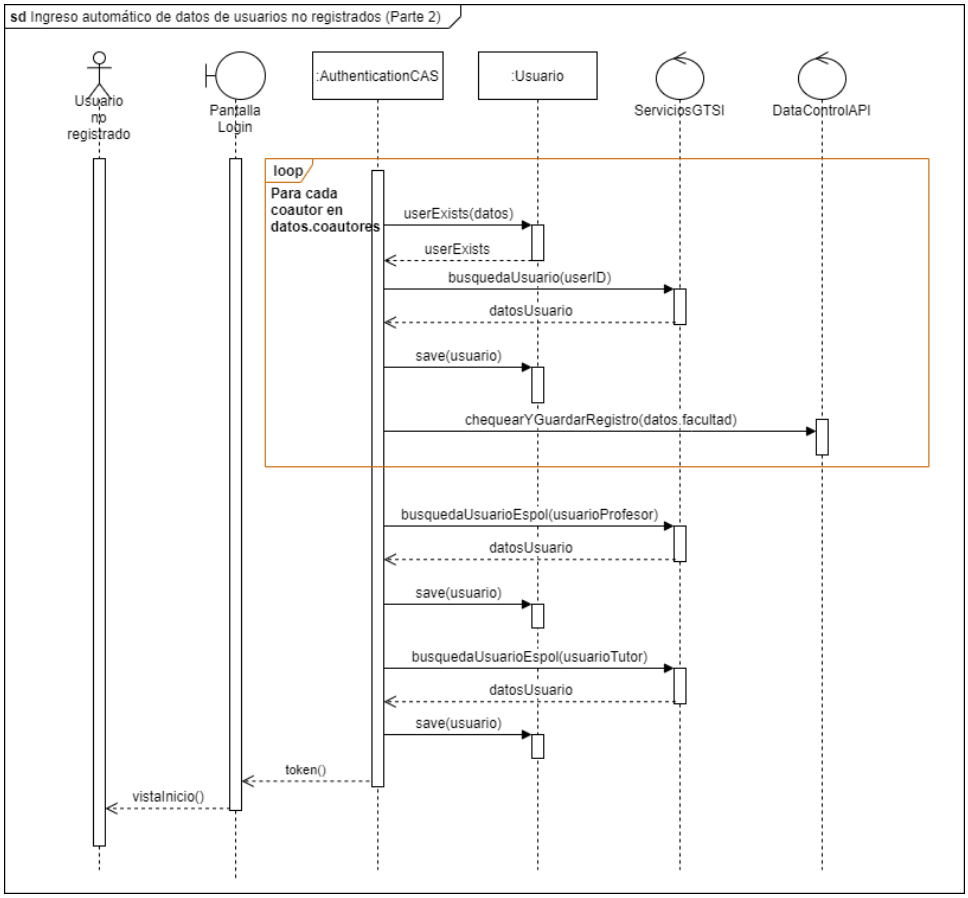

Figura 5 - Diagrama de secuencia del ingreso de datos automáticos cuando entra un usuario no registrado (Parte 2)

2.7 Modelo lógico

Dentro del diseño del modelo lógico que se tenía en el proyecto anterior, existe una tabla Rol, que posee los cinco roles que interactúan en el sistema AGATA. Existe también una tabla Usuarios y RolUsuario, que contiene la información de cada uno de los usuarios registrados, y la relación entre las tablas Rol y Usuario, así como la relación con la tabla PeriodoAcademico, respectivamente. Es importante mencionar, que un mismo usuario puede tener diferentes roles dentro de un mismo término académico y a su vez cada uno de los roles definidos tienen dependencia al periodo académico. En el modelo lógico de la base de datos del proyecto anterior, la tabla RolUsuario se encarga de proveer los permisos necesarios según el rol y un periodo académico correspondientes por lo que se agregó un campo llamado activo dentro de la tabla RolUsuario para poder manejar los usuarios por término académico como muestra la Figura 6

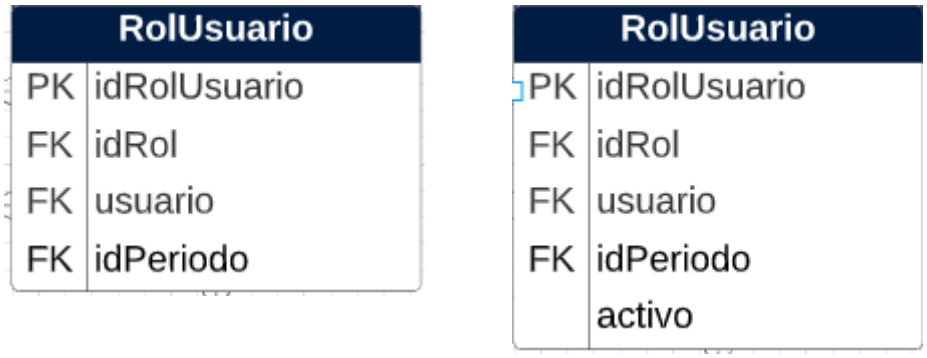

Figura 6 - Tabla modificada

2.8 Diseño de interfaces

En esta sección, se muestra el diseño de las nuevas ventanas a realizar y aquellas ventanas con modificaciones que no afectan al flujo actual dentro del sistema AGATA. Entre las ventanas, se tienen los mensajes de retroalimentación en los siguientes escenarios:

- Inicia sesión un usuario por primera vez en el sistema AGATA y se muestra un mensaje de carga.
- Un usuario intenta iniciar sesión en el sistema AGATA y no responden las APIs del GTSI
- Un usuario intenta iniciar sesión en el sistema AGATA y el sistema no responde.

Estas ventanas permiten darle al usuario una respuesta visual si llegase a fallar alguna petición dentro del sistema cuando intentan iniciar sesión.

Para la selección del término académico, se agregó una lista desplegable con los períodos académicos existentes para poder visualizar entre términos si se llega a alargar el proceso entre un semestre y otro. También, se muestran el formato de las notificaciones que se enviarán en el módulo de Secretaría para avisar de forma personalizada a los estudiantes que no han culminado ni iniciado el proceso de titulación. Por último, dentro del sistema administrativo de AGATA, se crean dos ventanas para habilitar y deshabilitar roles en cada término académico, que funciona en conjunto con la interacción de los usuarios con el frontend visualizando la información correspondiente por término académico.

2.8.1 Sistema AGATA

En el apartado de interfaz web, se muestran pantallas de error como la Figura 7 que indica si existe algún problema con las conexiones hacia la base de datos local del sistema AGATA o como se muestra en la Figura 8 cuando los servicios internos de GTSI no responden.

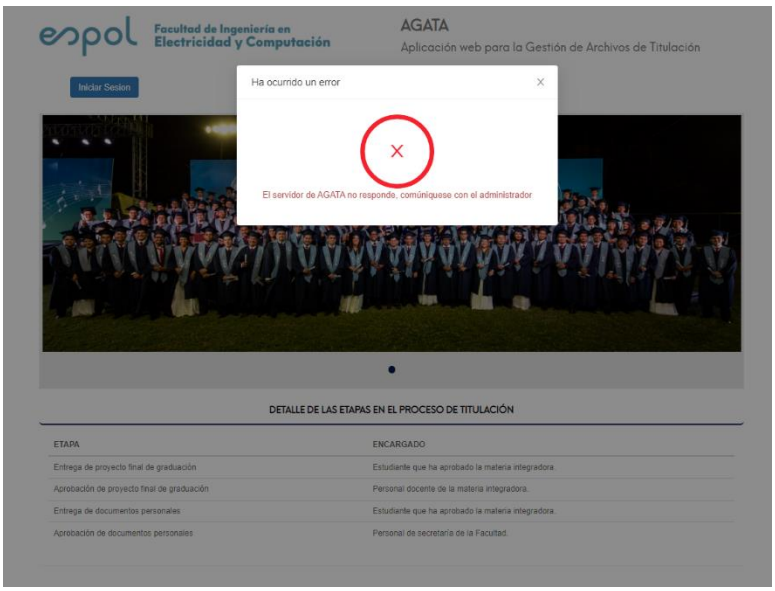

Figura 7 - Mensaje Modal cuando los servicios de Backend de AGATA no funcionan

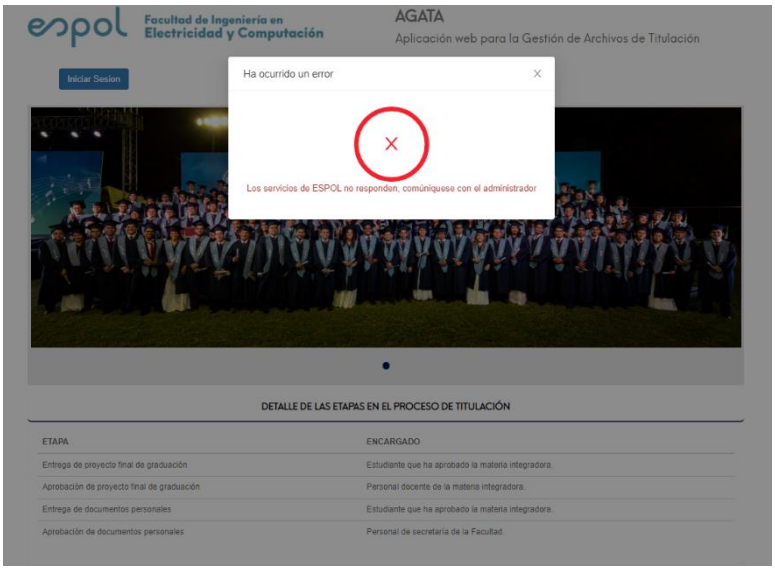

Figura 8 - Mensaje Modal cuando los servicios de GTSI no funcionan

También se muestra un modal cuando un usuario inicia sesión de forma exitosa al sistema.

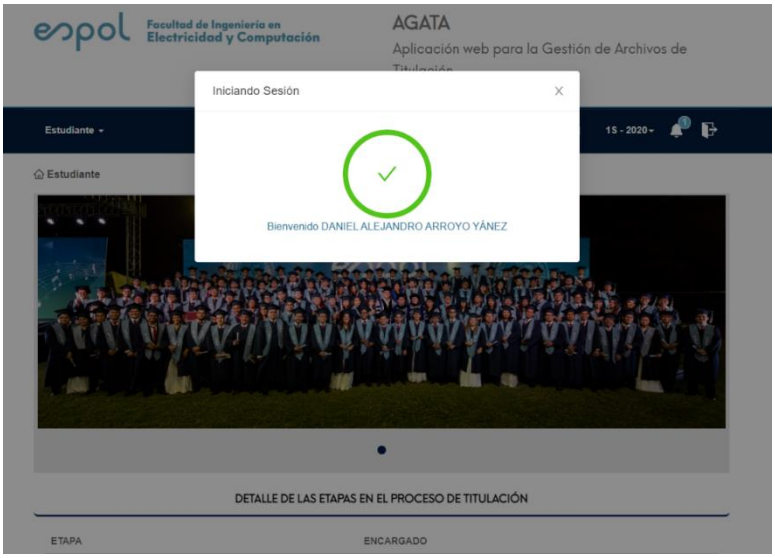

Figura 9 - Mensaje Modal cuando un usuario inicia sesión de forma exitosa

En la Figura 10, se muestra la pantalla de carga de datos cuando un estudiante es nuevo dentro del sistema AGATA. Esta operación tiene una duración aproximada de 15 segundos, aunque depende de la velocidad de respuesta de los servicios del GTSI.

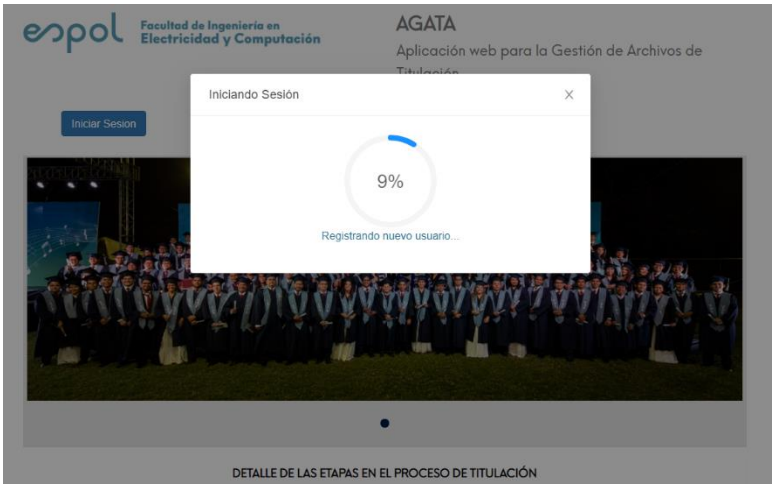

Figura 10 - Ventana de carga de datos al inicio de sesión de un usuario no registrado en el sistema AGATA

2.8.2 Selección del período académico

Un requerimiento de la secretaría de la ESPOL es la capacidad de elegir un término académico para cualquier usuario del sistema AGATA ya que no había forma de saber en qué término se referían los datos mostrados en las diferentes pantallas. Para satisfacer este requerimiento, se implementa un menú desplegable en la barra de navegación principal de la interfaz web como se

muestra en la Figura 11, donde se encontrará el último término disponible elegido por defecto. Como se muestra a continuación, al desplegarse el menú, se encuentran los demás términos disponibles:

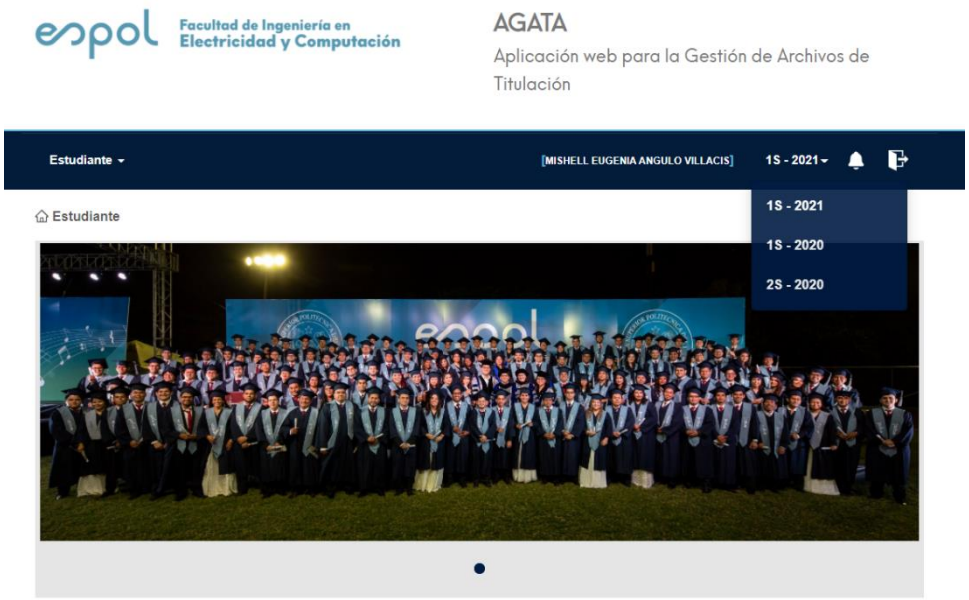

Figura 11 - Menú desplegable de los términos disponibles

Gracias a esta posibilidad de elegir el término, se aprovecha el manejo de los roles de usuario por término en la vista administrador. Más adelante se explica detalladamente el manejo de los permisos por rol y por término en el módulo administrador de AGATA.

2.8.3 Envío de notificaciones

El envío de notificaciones automáticas cada vez que los usuarios realizan un paso como entrega de documentos, rechazo o aprobación de estos se implementó por medio del uso de SMTP. Las notificaciones automáticas se envían por correo utilizando una plantilla. También en el módulo de Secretaría se envían mensajes de manera personalizada para notificar al estudiante que el proceso está detenido o alguna otra novedad. Ver secuencia en la Figura 12

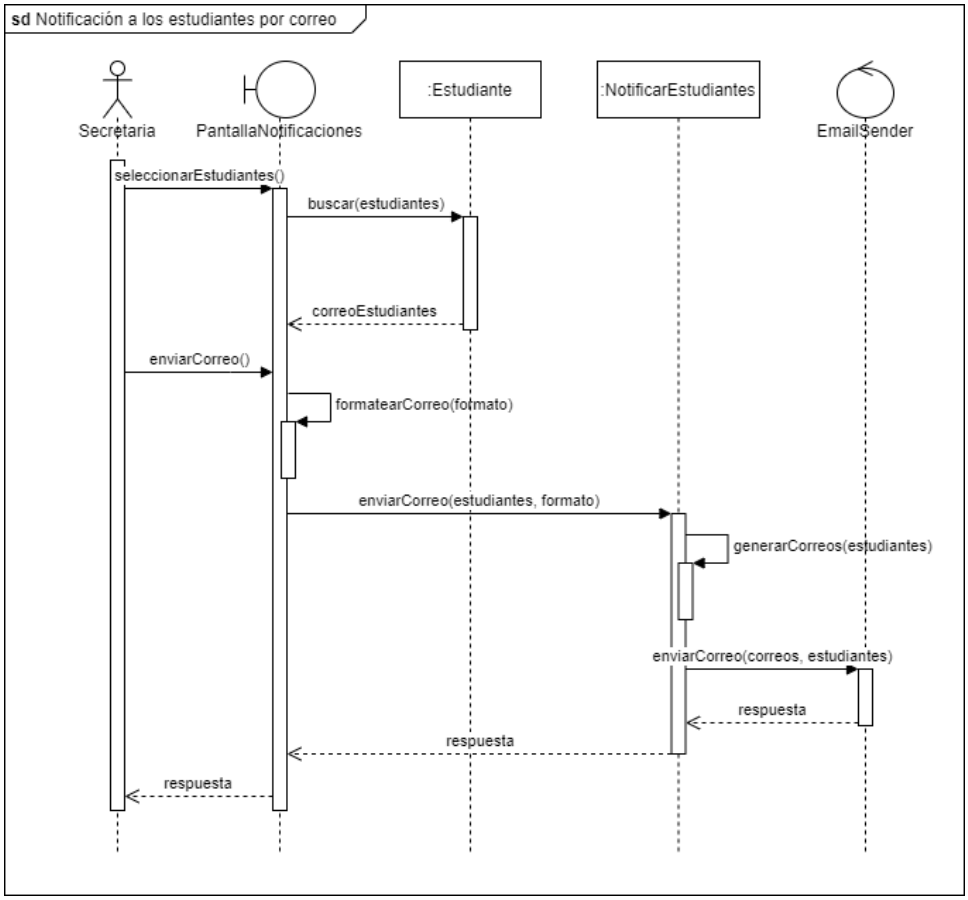

Figura 12 - Diagrama de secuencia del envío de notificaciones por correo a estudiantes

En el módulo Secretaría, ver Figura 13 y Figura 14, se puede notificar a los estudiantes con un mensaje de noreply utilizando un formato predeterminado por la secretaría de la ESPOL. Primero, se pueden escoger los usuarios estudiantes a los cuales se les va a enviar el mensaje.

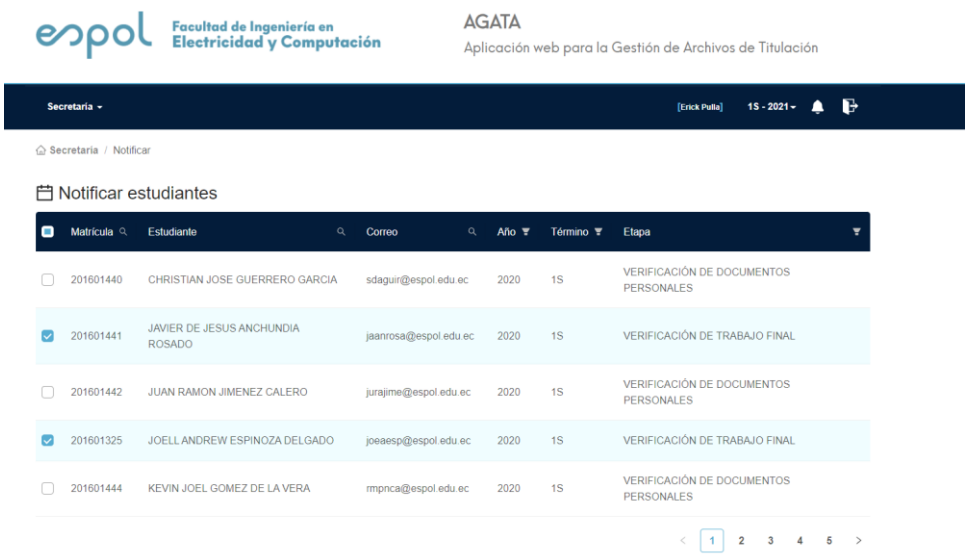

Figura 13 - Listado de estudiantes para seleccionar y enviar una notificación

Se escribe el asunto para que finalmente, se envía el mensaje con una plantilla automática. El usuario secretaria puede ver el mensaje predeterminado que se va a enviar a los estudiantes como se muestra a continuación:

Figura 14 - Formato de mensaje automático que envía la secretaria a los estudiantes seleccionados

2.8.4 Administrador

En este módulo se realiza el cambio de diseño al momento de registrar un nuevo término académico como se ve en la Figura 15. Se selecciona el año y el término

que puede ser 1T para primer término, 2T o 3T para el segundo término o tercer término, respectivamente.

Figura 15 - Formulario de registro de nuevo término académico

También se tienen dos formularios correspondientes por cada término académico para habilitar y deshabilitar permisos. Para habilitar los permisos como muestra la Figura 16, se escoge entre los roles estudiante, profesor, tutor, secretaría o subdecanato y se busca al usuario a habilitar dicho rol. En la búsqueda de los usuarios se mostrarán aquellos usuarios que no tienen asignado un rol en el término seleccionado. En la Figura 17, se tiene una lista que contiene los roles mencionados anteriormente y para la búsqueda de usuarios, se mostrará un listado con todos los usuarios que tienen un rol asignado en el término seleccionado para deshabilitarlo.

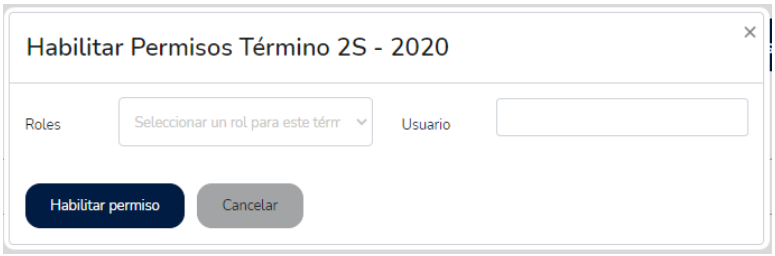

Figura 16 - Habilitar permisos por el término académico seleccionado

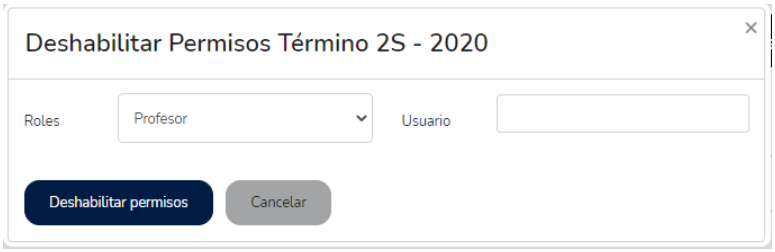

Figura 17 - Deshabilitar permisos por el término académico seleccionado

2.9 Correcciones adicionales

2.9.1 Correcciones al separar las aplicaciones en servidores distintos

Una de las correcciones más importantes fue modificar la clave que se encontraba alojada en el Localstorage y hacía uso del token de CSRF. El llamar a dicha clave permitía que realizar los peticiones con el método POST se ejecuten y evitar que el sistema considere que el usuario no está autorizado dentro de la aplicación. Además, también se realizó modificaciones en la clase de Autenticación del CAS para poder adaptarla al proyecto y lograr el redireccionamiento al sistema AGATA.

CAPÍTULO 3

3. RESULTADOS Y ANÁLISIS

En este capítulo se pretende mostrar los resultados al evaluar el tiempo de registro de estudiantes, tutores y profesores de forma manual comparándose a la automatización del registro cuando por primera vez un estudiante inicia sesión. Revisión del tiempo en que el acta es extraída desde el SAAC de forma manual en comparación con el consumo de la integración de la API del GTSI, así como la carga del documento del trabajo final al DSPACE.

3.1 Configuración y despliegue

3.1.1 Configuración de los servidores web

Como fue indicado con anterioridad, el GTSI proporcionó al proyecto con un servidor de la red interna de ESPOL, en el cuál AGATA fue desplegado en primera instancia como un proyecto unido. Para esta nueva fase, se provee un nuevo servidor para el despliegue de los apartados de Frontend y Backend de forma separada.

3.1.2 Instalación y configuración de los servidores

Los dos servidores utilizan la distribución del sistema operativo Linux Ubuntu, por lo que se realiza la respectiva actualización de librerías e instalación de la herramienta de control de versionamiento Git. El proyecto cuenta con sus apartados de Frontend y Backend en dos repositorios diferentes, los cuales serán clonados por separado en los dos servidores respectivamente.

3.1.2.1 Servidor frontend

El primer servidor (200.10.147.105) ya cuenta con una instalación previa de Apache, Python 3, Django, Reactjs, PostgreSQL y librerías requeridas por sistema AGATA unificado. En este primer servidor se designa la instalación del apartado de Frontend (Reactjs), por lo que lo demás es desactivado, incluyendo el proyecto unificado desplegado en Apache.

3.1.2.2 Servidor backend

En el segundo y más reciente servidor (200.10.147.29), se realiza la instalación de PostgreSQL, Python 3, Django y Apache para el despliegue del Backend.

3.1.3 Versiones de herramientas de desarrollo

A continuación, se muestra en Tabla 1 las versiones de las herramientas instaladas en ambos servidores junto a su versión:

Herramienta	Versión
Python	3.8.10
Django	3.4
Reactjs	17.0.2
react-dom	17.0.2
Apache	2.4.41
PostgreSQL	13.3

Tabla 1 - Versión de las herramientas utilizadas

3.2 Despliegue del sistema AGATA

El despliegue del sistema AGATA se inicia con archivos de configuración de Apache, los cuales se encargan de ejecutar el servicio de Reactjs y Django en los servidores de Frontend y Backend respectivamente. Estos servicios se ejecutan como procesos en segundo plano. Ambos servicios de Frontend y Backend son ejecutados en el puerto 80 (http) de sus respectivos servidores. Ver Figura 18 y Figura 19

```
<VirtualHost *:80>
        # The ServerName directive sets the request scheme, hostname and port that
        # the server uses to identify itself. This is used when creating
        # redirection URLs. In the context of virtual hosts, the ServerName
        # specifies what hostname must appear in the request's Host: header to
        # match this virtual host. For the default virtual host (this file) this
        # value is not decisive as it is used as a last resort host regardless.
        # However, you must set it for any further virtual host explicitly.
        #ServerName www.example.com

        ServerAdmin webmaster@localhost
        DocumentRoot /home/manager/FrontEnd/build

        <Directory /home/manager/FrontEnd/build>
                Options Indexes FollowSymLinks
                AllowOverride None
                Require all granted
                ErrorDocument 404 /index.html
        </Directory>

        # Available loglevels: trace8, ..., trace1, debug, info, notice, warn,
        # error, crit, alert, emerg.
        # It is also possible to configure the loglevel for particular
        # modules, e.g.
        #LogLevel info ssl:warn

        ErrorLog ${APACHE_LOG_DIR}/error.log
        CustomLog ${APACHE_LOG_DIR}/access.log combined

        # For most configuration files from conf-available/, which are
        # enabled or disabled at a global level, it is possible to
        # include a line for only one particular virtual host. For example the
        # following line enables the CGI configuration for this host only
        # after it has been globally disabled with "a2disconf".
        #Include conf-available/serve-cgi-bin.conf
</VirtualHost>

# vim: syntax=apache ts=4 sw=4 sts=4 sr noet
```

Figura 18 - Archivo de configuración de Apache para el Frontend

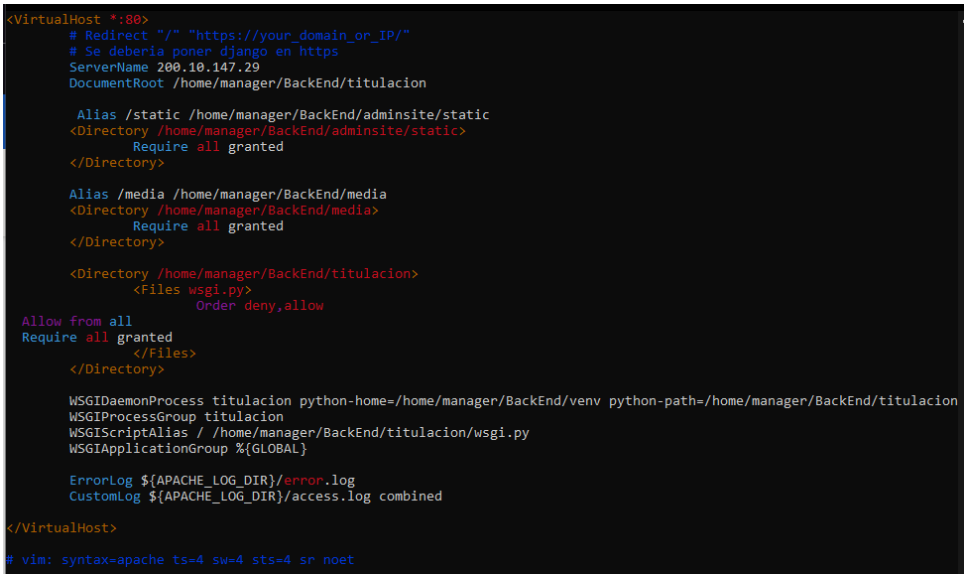

Figura 19 - Archivo de configuración de Apache para el Backend

3.2.1 Comunicación entre servidores de Frontend y Backend

Para la comunicación entre los servidores de Backend y Frontend de AGATA, surgieron varios cambios en la arquitectura de comunicación entre servidores, especialmente por problemas de políticas de CORS, y restricción de Cookies, ver Figura 20 y Figura 21. Esto ocurre tanto en Google Chrome, Opera, Mozilla y Microsoft Edge.

Según (Google Developers, 2020), en mayo del 2020, se anunció un cambio en las políticas de restricción de cookies de terceros y CORS para la protección y refuerzo de la privacidad de los usuarios. La aplicación Frontend consulta las APIs de un servidor separado en donde se encuentra la aplicación Backend, lo cual provoca que el navegador reconozca las consultas como una aplicación de terceros.

Figura 20 - Mensaje de error de CORS

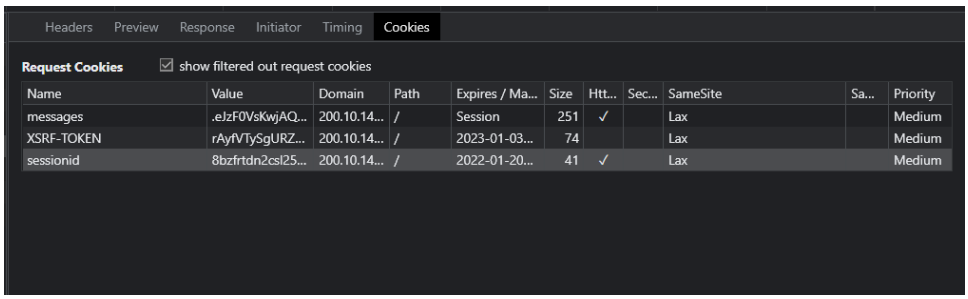

Figura 21 - Cookies son filtradas y excluidas

3.2.2 Solución con servidor de Tunneling

El origen del problema proviene de los navegadores web que bloquean el consumo por parte Frontend al Backend ya que el Backend es considerado como una aplicación de terceros al estar desplegado a un servidor distinto.

Por este motivo, se escoge como solución un servidor tunnel “socat” (comando de Linux) ejecutándose desde el servidor de Frontend. Este servidor se encarga de escuchar al puerto 80 del servidor de Backend y lo traduce a un puerto libre del servidor Frontend (en este caso, se escogió el puerto 443 por políticas del GTSI). Esto permite que las APIs del Backend emulen provenir del puerto 443 del propio servidor de Frontend (conectado al puerto 80 del servidor del Frontend) del propio servidor de Frontend, evitando las políticas de restricciones a aplicaciones de terceros ya que el Frontend se consume a sí mismo en un puerto diferente.

```
manager@srv01web-titulacion:/etc/apache2/sites-available$ sudo lsof -i -P -n | grep LISTEN
vsftpd        933            root    3u  IPv6    22442      0t0  TCP *:21 (LISTEN)
sshd          964            root    3u  IPv4    25683      0t0  TCP *:22 (LISTEN)
sshd          964            root    4u  IPv6    25694      0t0  TCP *:22 (LISTEN)
systemd-r  533357 systemd-resolve   13u  IPv4  1345912      0t0  TCP 127.0.0.53:53 (LISTEN)
postgres  2017559        postgres    5u  IPv4 16480187      0t0  TCP *:5432 (LISTEN)
postgres  2017559        postgres    6u  IPv6 16480188      0t0  TCP *:5432 (LISTEN)
socat     2738999            root    5u  IPv4 18416245      0t0  TCP *:443 (LISTEN)
apache2   3702132            root    4u  IPv6 20935992      0t0  TCP *:80 (LISTEN)
apache2   3839203        www-data    4u  IPv6 20935992      0t0  TCP *:80 (LISTEN)
apache2   3839241        www-data    4u  IPv6 20935992      0t0  TCP *:80 (LISTEN)
manager@srv01web-titulacion:/etc/apache2/sites-available$
```

Figura 22 - Servidor de tunnel “socat” haciendo un túnel entre el puerto 80 del servidor de Backend al puerto 443 del servidor de Frontend

3.2.3 Cambios en Arquitectura de despliegue Planteado

Debido a los problemas de restricciones de los navegadores web mencionados anteriormente, la arquitectura de despliegue sufre un ligero cambio. A continuación, se muestra el antes y después del diagrama de arquitectura del despliegue de AGATA:

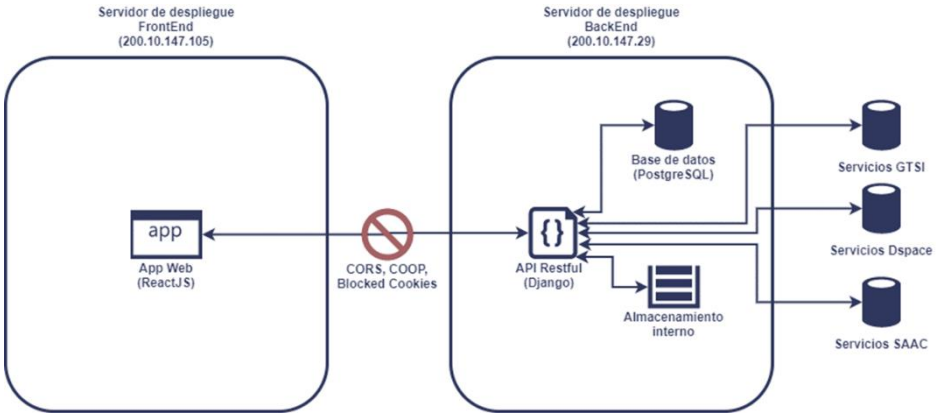

Figura 23 - Diagrama de despliegue original (problemas de restricciones)

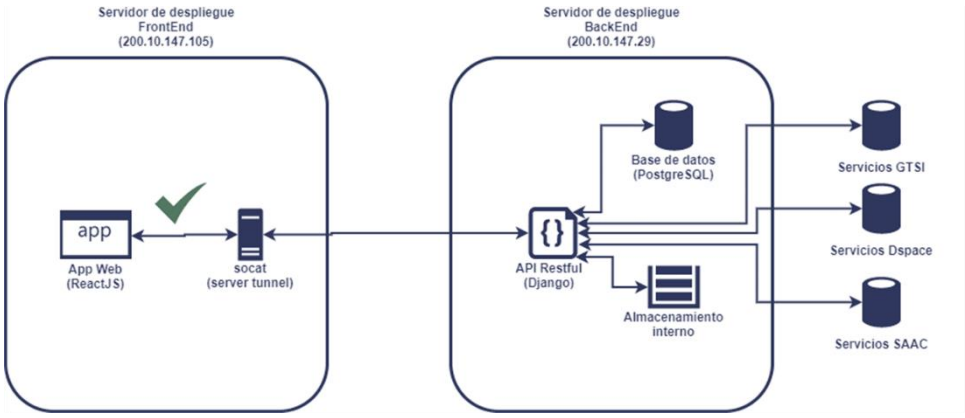

Figura 24 - Diagrama de despliegue modificado (con servidor de tunneling “socat”)

3.3 Consumo de APIs de terceros

3.3.1 APIs de GTSI

El GTSI proporcionó las siguientes 10 APIs, de las cuales 2 son para el manejo del acta de grado del estudiante y 8 que contienen información del estudiante. Estas APIs pertenecen a la ruta mostrada en la **¡Error! No se e ncuentra el origen de la referencia.** y de las cuales se detallan a continuación:

Figura 25 - URL del servidor del GTSI para pruebas

- **/api/MateriaEstudiante/InfoEstudiante/<usuarioEspol>**: esta API nos proporciona la información mostrada en la Figura 26 como la matrícula, cédula, nombres completos del estudiante, así como el código y nombre de la carrera e información de la facultad a la que pertenece dicho estudiante como el nombre completo y el código de la facultad.

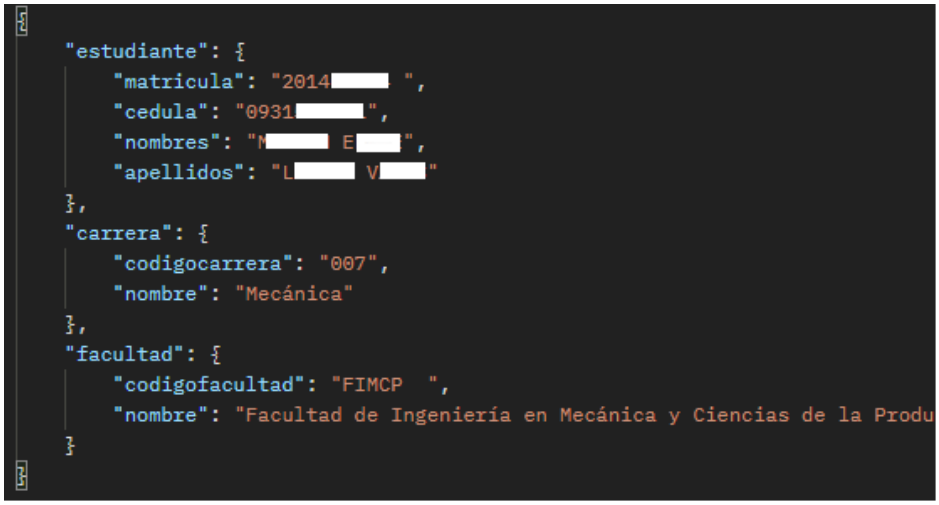

Figura 26 - Información del API InfoEstudiante

- **api/MateriaEstudiante/InformacionIntegradora/<codigoCarrera**>: esta API como se muestra en la Figura 27 devuelve la información del nombre de la materia integradora de la respectiva carrera, así como el código de la materia, año y término.

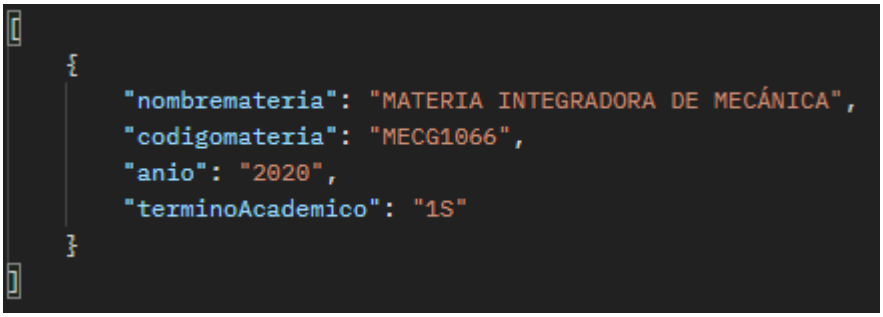

Figura 27 - API InformacionIntegradora

- **api/MateriaEstudiante/AproboIntegradora/<usuario>/<codigoMateria>**: esta API devuelve "0" si el estudiante consultado no ha aprobado la materia integradora como se muestra en la Figura 28 o el valor de "1" si el estudiante ha aprobado.

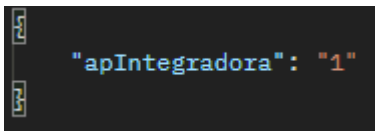

Figura 28 - API AproboIntegradora

3.3.2 APIs de DSPACE

El GTSI proporcionó la url y el formato de los datos a enviar usando DSpace, se utilizó la versión 5 de DSpace.

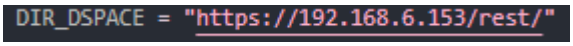

Figura 29 - URL de DSpace

- **/rest/login:** API para iniciar sesión y devolver el token. Las credenciales fueron provistas por el GTSI.
- **/rest/status:** API que devuelve el estado de la sesión
- **/rest/collections/{collectionID}/ítems:** API que devuelve el itemsId luego de ingresar el collectionID
- **/rest/item/{itemsid}/metadata:** API que almacena la metadata como autor, fecha, entre otros datos.
- **/rest/item/{itemsid}/bitstreams:** API que almacena el url del PDF del archivo subido al DSpace.

3.4 Costos

Para este proyecto el costo total es de $9600 debido a que el servidor, tecnologías utilizadas en este proyecto son open source por lo que no implica

gasto alguno. El valor obtenido es netamente considerando el desarrollo y/o mantenimiento al sistema. Ver Tabla 2.

Producto/Servicio	Unidad(es)	Duración	Valor Unitario	Valor Total
Desarrolladores	2	6 meses	$800	$9600
Servidores	2	Indefinido	$0	$0
			Total	$9600

Tabla 2 - Valores Costos del proyecto

3.5 Evaluaciones

Luego de los requerimientos realizados, se realizaron evaluaciones parametrizando por medio del tiempo de respuesta en el escenario del registro de un estudiante junto con su información correspondiente en el sistema. Ver Tabla 3.

Para el registro automático, el tiempo en que la información es almacenada en la base de datos del sistema AGATA es en promedio 5 segundos mientras que el registro manual realizado en el módulo del administrador demoraba en promedio 45 segundos. Para la carga del archivo al DSPACE demoraba por medio del proceso manual aproximadamente 3 minutos mientras que al consumir las APIs del DSPACE proveídas por el GTSI dura aproximadamente 20 segundos.

Subproceso	**Tiempo manual (s)**	**Tiempo automatizado (s)**	**Reducción (s)**	**Reducción (%)**
Registro de proyectos de titulación con sus actores (estudiantes, profesores y tutores)	480	40	440	92

Extracción de acta SAAC	120	40	80	66
Subida de archivos a DSpace y eliminación de deuda de no valor	300	15	285	95
Firma y registro de acta SAAC por parte de secretaría y subdecanato	172800	600	172200	99.3

Tabla 3 – Tabla comparativa del rendimiento antes y luego de la automatización

CAPÍTULO 4

Conclusiones Y Recomendaciones

Conclusiones

- Este proyecto logró ser desplegado en dos servidores diferentes tal como lo indican las políticas del GTSI, permitiendo así una mayor flexibilidad y escalabilidad que un sistema integrado.
- Fueron consumidas exitosamente todas las APIs provistas del GTSI tanto de la información del estudiante, acta saac y del DSpace, permitiendo automatizar gran parte del proceso en cuestión.
- Para los servidores desplegados es necesario realizar una recarga al servicio de apache para que los cambios en los sistemas puedan ser visualizados correctamente.
- Según (Google Developers, 2020), las políticas de restricción de cookies y CORS de los navegadores web restringen estrictamente a las aplicaciones de terceros a partir del mayo del 2020. Por ese motivo, para garantizar la comunicación eficiente entre servidores y evitar que los navegadores cataloguen al servidor del Backend como una aplicación de terceros, se utilizó un servidor de tunneling "socat" en el servidor del Frontend con el fin de garantizar una correcta comunicación con el Backend.
- En esta fase del proyecto se utilizó la API provista por el GTSI, lo cual permitió una reducción total del tiempo en los subprocesos que se realizaban de forma manual (reducción de 2 días con 11 segundos). Dichos subprocesos corresponden a: extraer y firmar acta SAAC, subir documentos a DSpace, eliminar deuda de no valor y registrar automáticamente los proyectos con sus usuarios.

Recomendaciones

- Para realizar pruebas y evitar problemas de caché, es necesario hacer uso de navegadores como Chrome. El uso del navegador Brave podría ocasionar que el sistema AGATA no funcione correctamente debido a que este navegador tiene problemas de cacheo de información.
- El url donde se encuentra alojado el documento final que se genera al subir archivos al DSpace debería poder visualizarse en la aplicación y no solo quede visible en los reportes que se generen en el módulo de la secretaría.
- Se recomienda la revisión del estado del servidor de tunneling “socat” cuando el servidor donde se aloja el Backend sufre fallos o caídas.
- Se recomienda una mejor y más rápida comunicación entre los desarrolladores del GTSI y de AGATA para agilizar la comunicación entre los servicios externos a AGATA como lo son DSpace y SAAC.

Referencias

- SALAZAR MERCHAN, J. J. (2020). *APLICACIÓN INFORMATICA PARA PROCESOS DE TITULACION DE LA UNIVERSIDAD ESTATAL DEL SUR DE MANABÍ* (Bachelor's thesis, Jipijapa. UNESUM).
- Matute, S. A., Avila-Pesantez, D., & Avila, M. (2020). Desarrollo de sistema Web basado en los frameworks de Laravel y VueJs, para la gestión por procesos: Un estudio de caso. Revista peruana de computación y sistemas, 3(2), 3-10.
- Luna, F., Millahual, C. P., & Iacono, M. (2018). PROGRAMACION WEB Full Stack 13-PHP: Desarrollo frontend y backend-Curso visual y práctico (Vol. 13). RedUsers.
- López Carranco, S. J. (2018). Desarrollo de un sistema web para la gestión y monitoreo de los procesos judiciales y de recursos humanos, con informes estadísticos de créditos, agencias y desempeño social para la cooperativa de ahorro y crédito mujeres unidas mediante la utilización del Framework Ruby On Rails y la Librería Reactjs (Bachelor's thesis).
- Vivas Carrillo, T. A., & Ubeda Arriaza, K. E. (2016). Implementación de API RESTFUL para uso en APP de ofertas (Chaplist), desarrollada con IONIC (Doctoral dissertation, Universidad de San Carlos de Guatemala).
- Jiang, T. (2017). Comparador de precios de los servicios de taxi (Doctoral dissertation, ETSI_Sistemas_Infor).
- Romero, J. L. V. (2015). *Instalación y configuración del software de servidor Web. IFCT0509*. IC Editorial.
- Pérez, E. M., & Cancio, S. A. P. (2014) Propuesta de arquitectura basada en componetes para plataformas de gestión de procesos desarrolladas en Django.
- Puig Pérez-Barquero, F., Rodríguez Díaz, J. A., González Perea, R., & Camacho Poyato, E. (2021). Desarrollo de una plataforma IoT para la gestión del riego de precisión.
- Muñoz, B., Ushca, L., & Martín, C. (2011). Implementación de una aplicación web para la automatización de procesos académicos y administrativos de Instituciones Educativas.
- González Lemus, E. J. (2019). *Sistema para la automatización del proceso de trabajo de graduación para los estudiantes de la Escuela de Ingeniería Mecánica Industrial, Facultad de Ingeniería, Universidad de San Carlos de Guatemala* (Doctoral dissertation, Universidad de San Carlos de Guatemala).
- De la Torre Sánchez, G. F. (2018). *Desarrollo de una aplicación web para la administración de procesos de graduación de una unidad de postgrados* (Master's thesis, Pontificia Universidad Católica del Ecuador).
- Aguirre Larrosa S., Pincay Jimenez R., (2021). *Sistema de gestión de documentos de titulación - Fase 1.* (Ingeniero en Computación, Escuela Superior Politécnica del Litoral)
- Espinoza J., Tigse E. (2021). *Sistema de gestión de documentos de titulación - Fase 2.* (Ingeniero en Computación, Escuela Superior Politécnica del Litoral)

- Brown, A. B., & Keller, A. (2006, April). A best practice approach for automating it management processes. In 2006 IEEE/IFIP Network Operations and Management Symposium NOMS 2006 (pp. 33-44). IEEE.
- Arcuri, A. (2019). RESTful API automated test case generation with EvoMaster. ACM Transactions on Software Engineering and Methodology (TOSEM), 28(1), 1-37.
- Chen, X., Ji, Z., Fan, Y., & Zhan, Y. (2017). Restful API architecture based on laravel framework. In Journal of Physics: Conference Series (Vol. 910, No. 1, p. 012016). IOP Publishing.
- Abdullah, H. M., & Zeki, A. M. (2014). Frontend and backend web technologies in social networking sites: Facebook as an example. In 2014 3rd international conference on advanced computer science applications and technologies (pp. 85-89). IEEE.
- Yacelga, A. R. L., Espinoza, J. L. A., & Vásquez, R. A. D. (2021). Aplicación de la metodología incremental en el desarrollo de sistemas de información. *Universidad y Sociedad*, *13*(5), 175-182.
- Google Developers (2020). , Get Ready for New SameSite=None; Secure Cookie Settings. Recuperado 5 de noviembre de 2021, de Google Search Central website: https://developers.google.com/search/blog/2020/01/get-ready-for-new-samesitenone-secure