Análisis, Diseño e Implementación de Componentes para los Módulos de Registro y Adjudicación del Servicio de Comedores Universitarios UIS.

Juan Esteban Duarte Rueda y Juan José Bayona Sepúlveda

Trabajo de Grado para optar al título de Ingeniero de Sistemas

Director

Fernando Antonio Rojas Morales
Magister en Ciencias computacionales

Codirector
Jathinson Meneses Mendoza
Especialista en Gerencia de Proyectos

Tutor
Jacksson Sonny González Bayona
Líder de equipo de desarrollo DTIC

Universidad Industrial de Santander

Facultad de Ingenierías

Fisicomecánicas

Escuela de Ingeniería de Sistemas e Informática

Bucaramanga

2024

Dedicatoria

A la Universidad Industrial de Santander y su equipo de trabajo íntegro y humano. A mi esposa, por su confianza y amor incondicional. A mi familia por su incansable apoyo y compromiso.

Juan Duarte

A mi madre por apoyarme incondicionalmente y por creer en mí. A las Residencias Universitarias UIS por recibirme y darme la oportunidad de culminar mis estudios en este bello lugar.

Juan José Bayona

Agradecimientos

Al profesor Fernando Rojas por su compromiso al aceptar este proyecto y darnos su enfoque para realizarlo con dedicación y excelencia.

Al Ingeniero Jacksson por su apoyo y compromiso con el proyecto y con la universidad.

Al profesor Gilberto Diaz, quien dio apoyo y recursos para la realización de este proyecto.

Al profesor Jathinson por su enseñanza, educación y dirección en el proceso de construcción del software.

Juan Duarte

A mis padres por su ejemplo de perseverancia y compromiso que me guiaron a ser un mejor ser humano.

A mi esposa, por su apoyo, confianza y sabiduría que han fortalecido mi carácter profesional.

Juan José Bayona

A mi madre y a mis hermanos, su confianza y apoyo fueron pilares fundamentales durante este proceso. A todos los integrantes de las Residencias Universitarias UIS por acogerme en esta gran familia.

Tabla de contenido

Lista de Figuras

Lista de Tablas

Resumen

Título: Análisis, Diseño e Implementación de los Componentes para los módulos de Registro y Adjudicación del Servicio de Comedores Universitarios UIS. *

Autores: Juan Esteban Duarte Rueda, Juan José Bayona Sepúlveda **

Palabras Clave: Comedores universitarios, plataforma Web, Estudiantes UIS.

Descripción: El sistema de gestión de los comedores universitarios UIS en la actualidad viene presentando una serie de desafíos que afectan el uso de los encargados desde su área administrativa, e incluso de los estudiantes en los diferentes procesos que desde aquí se pueden abordar. Esto se debe a una interfaz de usuario poco atractiva, poco intuitiva y difícil de conocer y aprender. Por otro lado, los estudiantes no encuentran una forma clara de realizar procesos simples como el registro al servicio y, en ciertos casos, resulta un proceso repetitivo y poco claro. Para abordar estos problemas, este proyecto propone el diseño e implementación de un sistema web que permita, tanto a administrativos como estudiantes, tener una herramienta que integre los procesos primarios tales como el registro estudiantil y la adjudicación de estos al sistema de comedores. Esta propuesta tiene como propósito optimizar los tiempos de respuesta en los procesos, mejorar la interacción entre los distintos usuarios con el sistema y permitir actualizar los sistemas de información a partir del proceso de renovación de tecnologías institucionales.

---

* Trabajo de grado

** Facultad de Ingenierías Fisicomecánicas. Escuela de Ingeniería de Sistemas e Informática.

Director: Fernando Antonio Rojas Morales. MSc en Ciencias de la Computación.

Abstract

Title: Analysis, Design, and Implementation of Components for the Registration and Allocation Modules of the UIS University Dining Service.[*]

Authors: Juan Esteban Duarte Rueda, Juan José Bayona Sepúlveda [**]

Keywords: University Dining Halls, Web Platform, UIS Students.

Description: The UIS university dining hall management system currently faces a series of challenges that affect its use by administrators from their respective areas, as well as by students in various processes handled by the system. This is due to an unattractive, non-intuitive user interface that is difficult to understand and learn. Additionally, students struggle to find a clear way to perform simple tasks such as registering for the service, and in some cases, the process is repetitive and unclear. To address these issues, this project proposes the design and implementation of a web-based system that enables both administrators and students to have a tool that integrates primary processes such as student registration and their assignment to the dining hall system. The goal of this proposal is to optimize response times in processes, improve interaction between different users and the system, and enable the updating of information systems through the renewal of institutional technologies.

---

[*] Bachelor Thesis

[**] Faculty of Physicomechanical Engineering. School of Systems and Computer Engineering. Director: Fernando Antonio Rojas Morales. MSc in Computer Science.

Introducción

Los sistemas de información son herramientas vitales en los procesos productivos y/o corporativos de cualquier empresa. Un sistema computarizado es conformado por una serie de piezas que suman valor al sistema global. Por lo general, estos sistemas trabajan con la finalidad de producir, guardar, procesar y proveer información de valor para el ente interesado.

Un sistema de información bien estructurado, ejecutado e incorporado en las dinámicas empresariales juega un papel fundamental en cualquier organización. Estos sistemas cumplen con el propósito de automatizar, modernizar e incluso, eliminar ciertas capas de los procesos productivos de un área en específico de una organización. Si bien estos sistemas pueden resultar claves en las dinámicas empresariales; su diseño e incorporación van de la mano con un ambiente tecnológico que se acomode a las necesidades que dicho sistema demanda.

Las dinámicas cambiantes en el sector TI alrededor del mundo proveen infraestructura moderna, herramientas de hardware y software optimizadas, redes informáticas avanzadas, recursos de cómputo especializados, almacenamiento estructurado y seguro de la información, entre muchas ventajas que deben ser adoptadas por las empresas emergentes y, más aún, por aquellas referentes en el plano nacional e internacional de sus sectores productivos.

La Universidad Industrial de Santander, en su dependencia de bienestar universitario, ha venido prestando un servicio de alimentación estudiantil a los estudiantes de bajos recursos por más de 30 años; y, en este trabajo se ha incorporado un sistema de información que brinda la facilidad de controlar de una manera más sencilla y optima este servicio. Sin embargo, este sistema no logra alcanzar todas las ventajas que podría proporcionar al usuario final debido a su atraso temporal en función de las tecnologías que emplea, demoras en tiempos de respuesta por las características del desarrollo, escasez de diseño para mejorar las interacciones del usuario con el

sistema, entre otros motivos.

Por estas razones, la Universidad Industrial de Santander, busca crear un nuevo sistema que, además de solucionar todas las falencias anteriormente mencionadas, permita garantizar características de escalabilidad, mantenibilidad, usabilidad y solidez a los procesos que busca apoyar.

Este proyecto puntual se encargó de proveer los servicios e interfaces necesarias para el registro y adjudicación al servicio de comedores universitarios UIS; haciendo énfasis y cumplimiento en los requisitos funcionales y no funcionales que se requirieron.

1. Planteamiento Y Justificación Del Problema

Actualmente, la Universidad Industrial de Santander cuenta con varios sistemas de información que satisfacen las necesidades de sus diferentes dependencias. No obstante, muchos de estos sistemas fueron desarrollados hace más de una década con tecnologías que en su momento eran innovadoras, pero que han perdido vigencia con el paso del tiempo debido a la aparición de mejores soluciones en el mercado y a la falta de una comunidad que brinde soporte a su crecimiento. Algunas de estas tecnologías son Java 5, JSP (JavaServer Pages) y IBM Informix. Esta situación plantea un desafío en cuanto a escalabilidad, mejora y soporte de los sistemas existentes, siendo este, junto con la tarea de eliminar la incompatibilidad y redundancia de información, el principal objetivo del proyecto de Sistemas de Información (RSI); articulando los procesos de la UIS con el sistema de información.

El sistema de información actualmente implementado en el servicio de comedores universitarios de la UIS permite la gestión de procesos como el registro, adjudicación, seguimiento y cronograma relacionados con este servicio de bienestar estudiantil. Aunque el sistema cumple con su propósito no está diseñado de forma modular, tiene tiempos de respuesta prolongados, carece de un diseño intuitivo para el usuario final y permite procesos que no deberían ser realizados por parte de los usuarios.

El problema que se solvento con este proyecto fue el inicio de la renovación para los sistemas de bienestar estudiantil; iniciando con el sistema de comedores. El desarrollo de los módulos de registro y adjudicación permite establecer una base sólida del software, en función de expandir el desarrollo completo e integral de este sistema en cuestión y, además, iniciar el proceso de renovación de todos los sistemas asociados a bienestar estudiantil. Por otro lado, este desarrollo

se une a una gran cantidad de proyectos que se encuentran actualmente en desarrollo, asociados a una gran iniciativa de RENOVACIÓN DE SISTEMAS DE INFORMACIÓN (RSI) el cual es supervisado y avalado por la Universidad Industrial de Santander en cabeza del señor rector Hernán Porras Diaz.

2. Objetivos

2.1 Objetivo general

Realizar el análisis, diseño, planeación e implementación del proceso de registro y adjudicación para el sistema de información asociado al servicio de comedores universitarios de la Universidad Industrial de Santander.

2.2 Objetivos específicos

Identificar los requerimientos funcionales y no funcionales que conllevan la realización del registro y adjudicación del sistema de comedores.

Formular un diseño enfocado a responder las necesidades de los usuarios administrativos para garantizar el proceso de registro y adjudicación.

Desarrollar los respectivos módulos planteados a partir de una infraestructura de microservicios que respondan al diseño propuesto.

Validar el funcionamiento de las piezas de software implementadas por medio de distintas pruebas de calidad.

Tabla 1

Tabla de cumplimiento de objetivos.

	Objetivo	Alcance	Ubicación
1	Identificar los requerimientos funcionales y no funcionales que conllevan la realización del registro y adjudicación del sistema de comedores.	Levantamiento de requerimientos a partir de reuniones con el cliente, en donde se identificaron las necesidades que debe suplir el nuevo sistema. Se definió de manera escrita dichos requerimientos y se categorizaron como requerimientos funcionales o no funcionales.	Página 30 a 35
2	Formular un diseño enfocado a responder las necesidades de los usuarios administrativos para garantizar el proceso de registro y adjudicación.	Se estableció un flujo lógico de datos para llevar a cabo los procesos administrativos requeridos el registro de estudiantes y su posterior adjudicación. Se diseño un modelo de base de datos que permite almacenar la información requerida, en función de soportar los procesos internos de los usuarios administrativos y el respectivo registro de los estudiantes.	Página 36 a 43
3	Desarrollar los respectivos módulos planteados a partir de una infraestructura de microservicios que respondan al diseño propuesto.	Se realizó la implementación de los servicios requeridos para soportar las acciones básicas de los procesos administrativos que garantizan el estado optimo e inicial del registro de los estudiantes. Posteriormente, se realizó la implementación del servicio de registro, realizando las respectivas validaciones	Página 44 a 66

		transversales, haciendo uso de los diferentes sistemas académico-administrativos de la universidad.	
4	Validar el funcionamiento de las piezas de software implementadas por medio de distintas pruebas de calidad.	Se llevaron a cabo una serie de pruebas unitarias con la finalidad de probar los casos de uso existentes en los módulos desarrollados. Se realizaron pruebas por parte de un equipo transversal de QA para validar el correcto funcionamiento del software desarrollado.	Página 67 a 69 Y Página 77 a 95

3. Marco de Referencia

3.1 Marco Conceptual

El proceso de desarrollo de software cuenta con una serie de procesos en donde buscamos, a partir de una serie de pasos y/o etapas en donde podemos reconocer y optar por un conjunto de herramientas que nos darán la capacidad de alcanzar una solución *(2023, Amazon Web Services).*

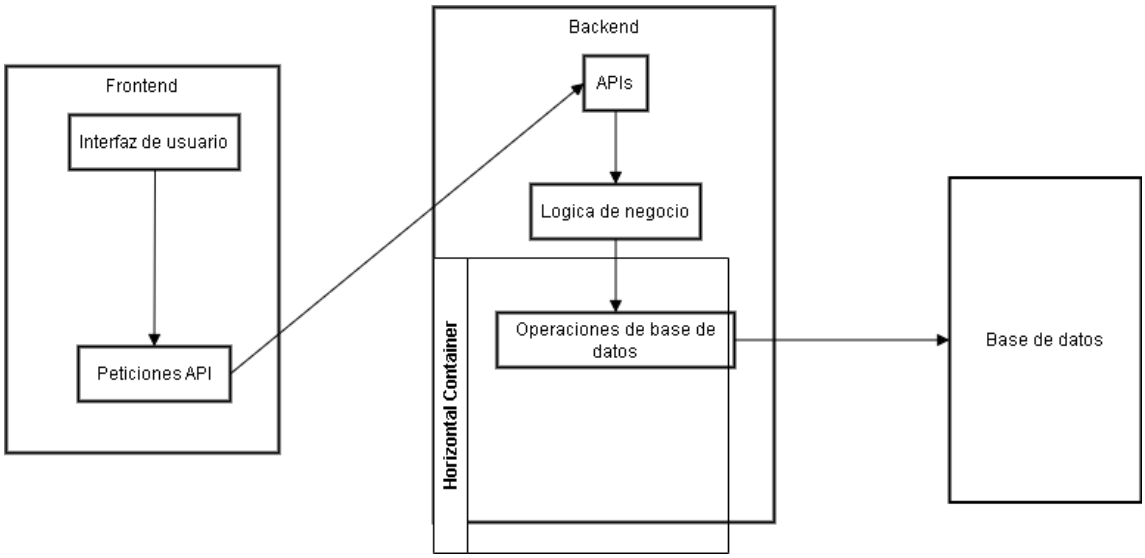
En función de lo anterior, este proyecto se aborda a partir de metodologías tradicionales de desarrollo de software codificado segmentado en capas. Estas capas podemos conocerlas o diferenciarlas como Frontend, Backend y base de datos. El frontend es todo aquello con lo que un usuario ve e interactúa cuando hace uso de una página web, plataforma móvil o aplicación de escritorio. Esta capa es conocida como la aplicación del lado del cliente.

Por otra parte, el *backend* es aquella capa de un sistema que se encarga de manipular la información o los datos que este recibe o provee; es donde se crea la lógica de los procesos que darán valor a la aplicación. Esta capa es conocida como la parte de la aplicación del lado del servidor. Todo este modelo lógico que contribuye a la construcción del software se ve apoyado en una arquitectura o un marco de trabajo en donde podemos organizar, modelar, diseñar y construir lo que hace falta para que la aplicación web funcione. En función de la arquitectura de software, el backend estará divido en capas, llamadas microservicios.

Un backend de microservicios es un estilo de arquitectura de software en donde se tiene una colección de servicios que son independientes en función del despliegue y que son débilmente acoplados, lo que significa que su responsabilidad con la aplicación debe ser única, particular y no muy extensa. En la figura #1 se puede observar cómo se relacionan la capa Frontend, Backend y la base de datos de una manera sencilla y, por otro lado, como deben ser las relaciones entre sí.

Figura 1.

Diagrama básico de una aplicación web con Frontend, Backend y una base de datos.



En la actualidad, el desarrollo de software en estas capas de frontend y backend han venido evolucionando, brindando marcos de trabajo o frameworks que simplifican la forma de construir software y hacerlo de una manera más eficiente y veloz. Un framework es una estructura de trabajo que provee un base para el proceso de desarrollo de una aplicación *(GeeksforGeeks, 2024).* Un framework da muchas ventajas a los desarrolladores en función del tiempo y esfuerzo requerido en el proceso de desarrollo de la aplicación. Esto se ve reflejado en la capacidad que dan de escribir código más limpio, más entendible y tener esquemas de archivos más organizados.

Una vez se comprenden las diversas capas en el desarrollo de software, se busca la forma de comunicarlas. En este caso, la forma de comunicación se hace a través de las API. La palabra API viene del acrónimo de “Application Programming Interface”, y su función es ser el intermediario que permite a dos aplicaciones comunicarse entre ellas *(What Is An API?*

(Application Programming Interface) | MuleSoft, s. f.)

De igual manera, la arquitectura de software REST (*Representational State Transfer*) es aquella que nos brinda las condiciones y estándares de cómo debe funcionar una API (Codecademy, s. f.). Esta arquitectura se ve apoyada en el protocolo HTTP, el cual predefine una serie de estándares o procedimientos que pueden ser utilizados para solicitar información. Los métodos GET, PUT, POST y DELETE son algunos de estos métodos y cada uno presenta una alternativa para realizar una acción sobre un recurso puntual *(HTTP Request Methods - HTTP | MDN, 2023).*

Apoyado en lo anterior, se hace necesario definir un patrón de diseño arquitectónico el cual le da forma y, como su nombre lo indica, un patrón al flujo y recuperación de la información dentro la aplicación. En este caso se hace uso del patrón *Modelo-Vista-Controlador* (MVC, por sus siglas en inglés: *Model View Controller*). El MVC es un patrón de diseño de software comúnmente utilizado para implementar interfaces de usuario, manejo de información y lógica de manejo de datos. Este patrón enfatiza en la separación de las capas de la lógica del negocio y la interfaz gráfica, dentro del software. Cada una de estas capas posee una responsabilidad única y debe mantenerse de esta manera. El modelo es aquel que representa a partir de un objeto una tabla o representación de la base de datos. La vista es aquel que controla el diseño y pantallas que tendrá a disposición el usuario y el controlador es el puente conector entre el modelo y la vista. *(MVC - MDN Web Docs Glossary: Definitions Of Web-related Terms | MDN, 2023b)*

Sumado a lo anteriormente mencionado, se hace uso de un patrón de diseño llamada Repository. El patrón de diseño Repository consiste en cómo funciona la lógica de persistir los datos de la aplicación hacia la base de datos. Este patrón agrega una capa al software que se encarga

de conservar externamente el acceso de los datos y, de esta manera, mantener el dominio agnóstico a sus fuentes de datos y sus implementaciones.

La capa *Repository* provee los métodos básicos de SQL, tales como inserción, actualización, lectura y eliminación de los datos. Se utiliza con la finalidad de evitar la duplicidad del código y mantener el dominio de acceso a datos de manera externa y oculta.

Un tema importante que considerar dentro del desarrollo de software viene a ser la forma de testear o probar que el código que se ha desarrollado cumple con su función basado en los requerimientos y/o reglas de negocio que busca responder. En este caso, el uso de pruebas unitarias permite tomar una parte pequeña del código y verificar que por sí sola funciona de la manera esperada. (What Is Unit Testing? - Unit Testing Explained - AWS, s. f.). El uso de estas se entiende como una buena práctica de programación al escribir código limpio e incluso, para algunas metodologías, estas deben guiar el proceso de desarrollo de código, tales como la metodología TDD (Test Driven Development).

3.2 Marco Tecnológico

3.2.1 Frontend.

En la actualidad la versatilidad y variabilidad de frameworks para el frontend es bastante amplia, sin embargo, uno de los frameworks más usados, robustos y con mayor comunidad de desarrolladores vigentes en Angular. Angular es un framework de desarrollo construido sobre TypeScript que facilita la creación de sistemas frontend a partir de su metodología basada en la orientación a componentes. Por otro lado, también proporciona una gama de librerías que cubren una amplia variedad de características, incluyendo el routing (Comunicación entre componentes), control de formularios (Formularios guiados por templates y formularios dinámicos), comunicación cliente-servidor y muchas más.

Entre las librerías más utilizadas, y con mayor estabilidad, es Angular Material. Esta es una librería de componentes estilizados a partir de los estándares de Google Material Design. Además de todos los componentes que ya vienen incorporados, dan la facilidad de crear componentes bajos sus estándares de clases, estilos y funcionalidades, lo cual permite al desarrollador construir una aplicación de manera mucho más rápida, sin la necesidad de preocuparse por funcionalidades transversales.

Por otro lado, es importante el uso de un administrador de paquetes como Yarn. Yarn, como se había dicho antes, es un administrador de paquete de código abierto alternativo a npm (Node Package Manager), el cual nos permite controlar las dependencias de los proyectos basados en JavaScript o TypeScript. Esta herramienta asiste en el proceso de instalación, actualización, configuración y eliminación de los paquetes dependientes en el proyecto. Además, optimiza los bundles que serán desplegados a los ambientes de pruebas, lo cual hace de nuestra aplicación más ligera, sin la necesidad de sobrecargar el servidor con miles de paquetes que no se utilizan.

3.2.2 Backend.

Como marco de trabajo para el backend, se utiliza el framework Spring Boot. Este framework brinda una gran facilidad al desarrollador para construir aplicaciones stand-alone, con un sistema de paquetes incorporados que hacen muy eficiente la construcción de API y con una comunidad inmensa que garantiza la estabilización de versiones del framework.

Al igual que el administrador de paquetes Yarn para el frontend, es necesario tener una herramienta para controlar el uso de dependencias que el proyecto requiere. Para este proyecto se decidió por usa Maven. Maven es una herramienta desarrollada por el grupo Apache para construir, publicar y desplegar aplicaciones para mejorar su gestión. A partir de sus ciclos de vida, Maven permite entender el proceso de desarrollo y documentación de las aplicaciones que se construye, dando así una mayor claridad al proceso de empaquetado y despliegue de estas.

Una de las librerías más importantes en este proceso fue Feign. La librería Feign nos permite reducir la complejidad de las interfaces que comunican nuestro backend con servicios REST externos; en otras palabras, otorga a nuestro backend la capacidad de ser intermediador con varios servicios, con la finalidad de recuperar información, manipularla y usar servicios transversales, algo de suma importancia para el proyecto de Renovación de los Sistemas de Información UIS.

Por otro lado, el proceso de recuperación de información de una fuente de datos es vital en este proyecto. Para esto, se hace uso de la librería Hibernate. Hibernate es un ORM (Object-Relational Mapping, por sus siglas en ingles) que le da la capacidad de Java de mapear o modelar dominios de datos a partir de tablas en bases de datos relacionales. (Kale, 2023).

3.3 Estado del Arte

Los sistemas institucionales se extiendan desde inicio del presente siglo como una extensión y mejoramiento a los procesos administrativos que se llevan a cabo dentro de la Universidad Industrial de Santander. El servicio de comedores UIS no escapa a esta regla, desde el año 2012 sale a producción el sistema “Comedores” asociado a la plataforma “Nuevas Versiones”, la cual fue una iniciativa institucional que centralizo los procesos y diversos subsistemas dentro de un mismo “ambiente” o plataforma.

Estos sistemas se enfocan en recopilar datos y a través de procesos internos realizados con el lenguaje de programación 4GL, el cual es provisto por el motor de base de datos INFORMIX. Estos sistemas al presentar estas condiciones resultan ser poco atractivos en términos de interfaces gráficas, muy difíciles de comprender por nuevos usuarios y código difícil de depurar y mantener por desarrolladores que no tengan un nivel avanzando en dicho lenguaje y proceso.

La mayoría de estos sistemas institucionales se construyeron en tecnologías vanguardistas, en ese momento, pero nunca se buscó una modernización de estas. En el sistema de comedores las tecnologías utilizadas fueron Java 5, Java Server Pages, Informix y Primefaces. Estas tecnologías presentan limitaciones y desventajas en función de su escasa documentación, errores persistentes no solucionados por versionamiento y carencia de acceso a nuevas funcionalidades para el hardware actual.

Por otro lado, en el proceso de renovación de los sistemas institucionales UIS, se han podido evidenciar muchas ventajas en el transcurso de 5 años de vigencia que lleva el proyecto. El uso de tecnologías modernas, frameworks robustos y estables y documentación accesible, entendible y completa, han brindado a la universidad la capacidad de desarrollar software de calidad a la medida (Algo que es un gran desafío) en tiempos óptimos y con una calidad muy

favorable.

La visión del proceso de renovación va ligado a la creación de sistemas institucionales con una mejor relación entre el proceso que se busca suplir, con los procesos que los sistemas antiguos han implementado. Este enfoque busca, además de renovar las herramientas de desarrollo, mejorar la capacidad de adaptabilidad, escalabilidad y convergencia de los procesos vigentes con los nuevos procesos que genera una entidad educativa de las cualidades que presenta la UIS.

3.3.1 Sistema oficial actual de comedores.

La Universidad Industrial de Santander actualmente cuenta con una plataforma, mencionada anteriormente, llamada “Nuevas Versiones”, esta plataforma institucional reúne todos los sistemas administrativos de la universidad y los da a disposición a través de los roles que puedan ser asociados a los usuarios vigentes. Por ejemplo, para acceder al sistema de comedores se debe tener primeramente un usuario dentro de la plataforma y contar con el rol “Auxiliar de comedores” o “Jefe de comedores”. Esto, solamente enfocado en la realización de las labores administrativas enfocadas en este servicio. Las siguientes figuras ilustran la interfaz del sistema de comedores, inicialmente, en el proceso de creación de semestres académicos.

Figura 2.

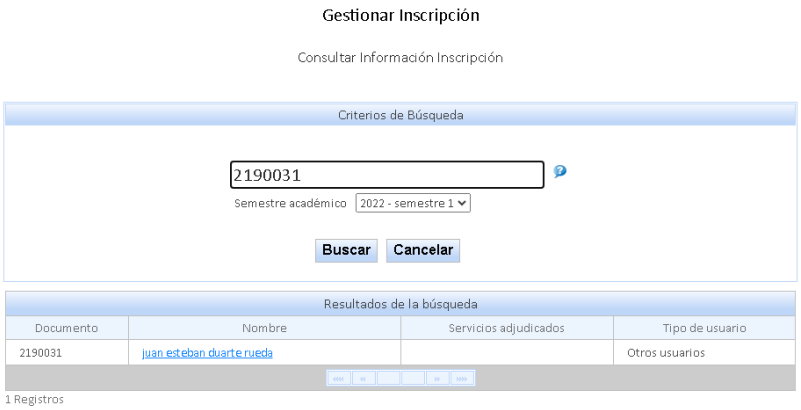
Vistas del sistema principal y la configuración de semestres académicos.

(a) Vista principal (b) Vista semestre académico

Nota. Sistema nuevas versiones, (2024). Captura de pantalla. Recuperado de www.uis.edu.co/comedores/modulos

Figura 3.

Vista de la pantalla de consulta para los estudiantes inscritos por semestre académico

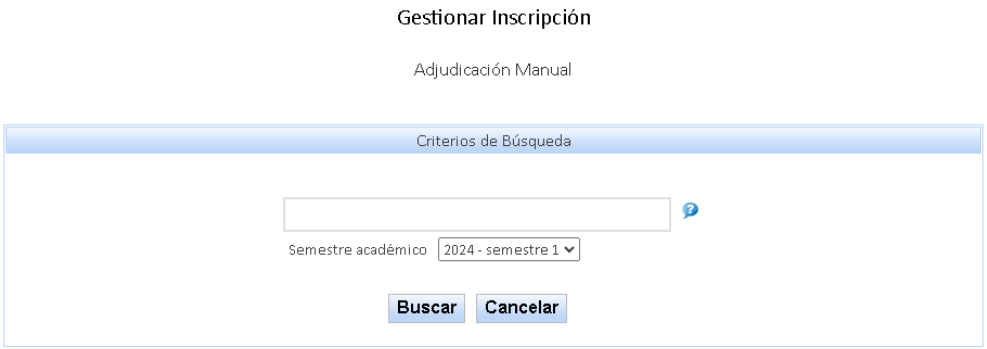


(a) Vista de estudiantes inscritos

Nota. Sistema nuevas versiones, (2024). Captura de pantalla. Recuperado de www.uis.edu.co/comedores/modulos

Figura 4.

Vista de la pantalla de adjudicación para los estudiantes inscritos a un semestre particular.



(a) Vista de estudiantes inscritos

Nota. Sistema nuevas versiones, (2024). Captura de pantalla. Recuperado de www.uis.edu.co/comedores/modulos

4. Requerimientos

Los requerimientos establecen y determinan las condiciones que el sistema debe cumplir y las capacidades que debe poseer. Dentro del sistema de comedores se definieron una serie de requerimientos agrupados por los módulos que se van a tratar y también, se categorizan como requerimientos funcionales o requerimientos no funcionalidades. Los requerimientos se presentarán en las categorías presentadas con anterioridad con la finalidad de tener una comprensión y claridad de estos.

4.1 Requerimientos Transversales

Para el desarrollo de los módulos se determinaron unos requerimientos que serán transversales para el sistema en general. Dentro de estos requerimientos se busca dotar al sistema con una mejor experiencia de usuario, mayor consistencia de los módulos y mejor integralidad entre los módulos que lo componen. Los requerimientos planteados se exponen a continuación.

4.1.1 Requerimientos Funcionales.

Los módulos del sistema deben contar con un estado inicial dependiente de los datos que se estén recompilando, a partir de esto, se muestra un estado vacío, pero con la capacidad de agregar información

Los módulos no deben permitir la alteración de la información a menos de que el usuario que se encuentre en ese momento tenga el rol definido para hacerlo. El rol es “Auxiliar de comedores”.

La información que se registre dentro de los módulos deberá dispersarse a su respectivo dominio de datos en Informix. Esto con la facilidad de poder realizar el desarrollo a futuro con los módulos del sistema primarios en producción.

4.1.2 Requerimientos No Funcionales.

Los módulos de la aplicación deben contar con un sistema de mensajes de internacionalización que se ajuste al idioma por defecto del navegador del usuario.

4.2 Inicio de sesión y navegación

4.2.1 Requerimientos Funcionales.

El módulo de inicio de sesión debe estar habilitado para estudiantes de pregrado con el respectivo rol de “Estudiante” que le permita acceder al sistema de comedores.

El módulo de inicio de sesión debe estar habilitado para personal administrativo adscrito a comedores con el respectivo rol de “Auxiliar de comedores” que le permita acceder al sistema.

Todo estudiante que no sea estudiante activo en el momento del login no podrá acceder al sistema y por ende al módulo de comedores.

4.2.2 Requerimientos No Funcionales

El sistema debe estar disponible para los usuarios en cualquier momento del día para poder acceder a consultar información actualizada sobre el servicio de comedores.

4.3 Registro de comedores

4.3.1 Requerimientos Funcionales.

El módulo de registro debe estar habilitado únicamente en las fechas y horas estipuladas de la convocatoria semestral de comedores para realizar un único registro en el sistema.

Si un estudiante ya ha realizado el registro de comedores para un semestre en particular debe ver un mensaje donde indique, que ya existe un registro vigente.

4.3.2 Requerimientos No Funcionales

El módulo de registro debe mostrar un breve resumen sobre la información del estudiante que está realizando el registro. Por ejemplo, nombre, código y semestre académico que está cursando.

4.4 Adjudicación

4.4.1 Requerimientos Funcionales.

El módulo de adjudicación debe permitir listar a todos los estudiantes inscritos para un semestre académico en particular.

El módulo de adjudicación debe permitir la adjudicación general al servicio de comedores semestral de los estudiantes inscritos.

El módulo de adjudicación debe permitir adjudicaciones individuales para los casos excepcionales de estudiantes inscritos.

El módulo de adjudicaciones debe permitir adjudicar a los estudiantes inscritos a partir de un numero de cupos disponibles suministrados por el Auxiliar de comedores.

El módulo de adjudicaciones debe categorizar a cada estudiante como un tipo de usuario dependiendo de si cumple con los requisitos establecidos.

El módulo de adjudicaciones debe encargarse de asignar los cupos disponibles de acuerdo con la prioridad de los tipos de usuario.

4.4.2 Requerimientos No Funcionales.

El módulo de adjudicación debe mostrar los estudiantes que ya han sido adjudicados para el semestre académico seleccionado.

El módulo de adjudicación deberá bloquearse una vez se cumpla el primer periodo de pago. En este punto, no se deberán realizar más adjudicaciones al sistema por lo restante del semestre académico.

4.5 Administración

4.5.1 Requerimientos Funcionales.

Los módulos administrativos deben poder manipular la información de tipos de subconvocatorias.

Los módulos administrativos deben poder manipular la información de requisitos generales y particulares que deben cumplir los inscritos

Los módulos administrativos deben poder manipular la información de las convocatorias, para crearlas y editarlas según sea la necesidad del usuario.

Los módulos administrativos deben poder manipular la información de los tipos de usuarios y ordenar la prioridad que tienen para el proceso de adjudicación.

4.5.2 Requerimientos No Funcionales.

Los módulos de administración deben tener un sistema de auditoria para validar los usuarios que realizan operaciones de base de datos sobre los mismos.

Los módulos de administración solamente pueden ser vistos por los Auxiliares de comedores. Al igual, que solo ellos, pueden realizar acciones sobre dichos módulos.

5. Diseño

A partir de la premisa de construir una aplicación robusta, integral y que pueda comunicarse con otras dependencias, sistemas y fuentes de datos, se definió seguir una arquitectura basada en servicios. Este enfoque de desarrollo de sistemas de software se basa en la idea de tener pequeños componentes de un sistema que puedan ser capaces de existir por si solos. Esta arquitectura da una guía para tener muchas partes autónomas dentro de un mismo sistema, haciendo que se relacionen entre si a partir de su definición por medio de interfaces. (What Is SOA? - SOA Architecture Explained - AWS, s. f.)

Dentro de esta arquitectura, se definirán una serie de servicios que tendrán la capacidad de comunicarse entre sí y cada servicio, de manera autónoma e independiente, realizar los procesos definidos dentro de sí. Este proceso y segmentación, dará una mayor facilidad al momento de mantener el software libre de errores, debido a que será mucho más sencillo evidenciar un error al localizar la capa o el proceso que lo genera.

Estos servicios definidos, serán la capa interna de la capa controladora. Esta capa estará expuesta y es aquella que define las posibles peticiones de información por medio del protocolo HTTP. Por medio de esto, nuestra aplicación frontend se comunicará con el backend por medio de los endpoints ofrecidos por esta API. Es así, como tendremos la información y toda la capacidad lógica del sistema a la mano del servicio de comedores y, para cualquier unidad adscrita que lo requiera.

5.1 Base de datos

5.1.1 Estructura lógica de datos.

A partir del análisis que se realizó sobre el sistema de “Nuevas Versiones” se pudieron

identificar estructuras de datos que generaban duplicidad en la información y, para efectos prácticos, resultaban insostenibles en función de la capacidad de escalar que debiese tener el servicio de comedores.

Actualmente, el servicio de comedores realiza un total de 3 convocatorias para que los estudiantes registren su inscripción y puedan ser parte del servicio de comedores. Cada una de estas convocatorias con información del semestre al cual se realiza la inscripción y en algunas, no se tiene en cuenta los servicios a los cuales el estudiante quiere que se le asignen.

En función de lo anterior, se definió el término "Subconvocatoria". Una subconvocatoria es un evento que ocurre dentro de una convocatoria general al servicio de comedores, que se caracteriza porque va direccionada a un tipo de usuario en específico. Estas subconvocatorias se definen a partir de los 3 eventos de inscripción que realizan los estudiantes en un semestre académico: *Inscripción por vulnerabilidad, inscripciones generales e inscripción para auxiliatura*.

Cada uno de estos eventos exige al usuario proveer al sistema diferente tipo de información y, a partir de esto, ser categorizado como un tipo de usuario que hace parte de comedores. Esto diverge con el sistema actual, debido a que, por ejemplo, realizar el proceso de verificación como estudiante vulnerable no tenía nada que ver con la solicitud del servicio de comedores; esto obligaba a dichos estudiantes a volver a entrar en un proceso de inscripción, para de esta manera solicitar los servicios de alimentación requeridos para el semestre académico. Estos casos generaban confusión debido a que muchos estudiantes que quedaban categorizados como vulnerables no obtenían el servicio por no volver a realizar su respectiva inscripción.

A partir de lo mencionado anteriormente, las subconvocatoria reducen sustancialmente el proceso de inscripción al eliminar los reprocesos y garantizan que el usuario

tenga la certeza de que realizar su registro en alguno de los 3 procesos de subconvocatorias le garantiza entrar ser considerado durante la adjudicación.

Para la consideración del lector, se omite el diseño de la base de datos por temas de protección de los datos sensibles a cargo de la universidad y confidencialidad de la información. Sin embargo, esta información será suministrada de manera parcial para términos de revisión y evaluación.

5.2 Frontend

5.2.1 Componentes del frontend.

El Frontend del proyecto se diseñó utilizando una arquitectura basada en módulos que se encuentra interconectada a partir de componentes transversales, tal como se ilustra en la figura 5. El primer módulo del frontend es el módulo App, este módulo es el encargado cargar todas las dependencias del sistema, tales como módulos de Angular, servicios para el llamado de peticiones HTTP, entre otros. Además, este módulo almacena el archivo de rutas primario que protege cada uno de los módulos y servicios que no deben ser visibles o usables para un usuario en particular que conozca la URL a la cual busca acceder. De igual manera, aquí se encuentran un módulo de componentes compartidos que pueden ser utilizados en diferentes módulos diferentes y que, no necesariamente tengan relación entre sí.

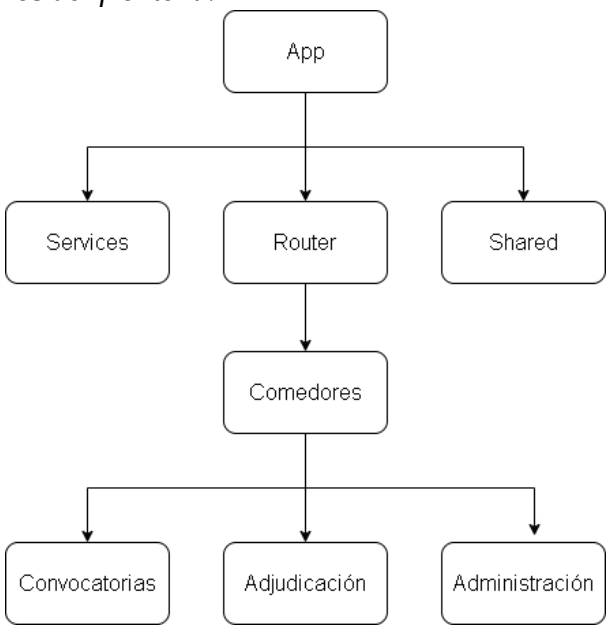
El módulo de convocatorias encontramos los diferentes componentes relacionados únicamente con los procesos relacionados con las mismas.

El módulo de adjudicación, al igual que el de convocatorias, se encarga de mostrar aquellos estudiantes inscritos para poder posteriormente realizar el proceso de adjudicación, basado en el semestre académico de la convocatoria en la cual realizo su inscripción.

Por último, encontramos el módulo administrativo, el cual internamente almacena los módulos de requisitos, tipos de usuarios, tipos de subconvocatorias, oferta de servicios y grupos especiales.

Figura 5.

Interconexión de los módulos del frontend.



5.3 Backend

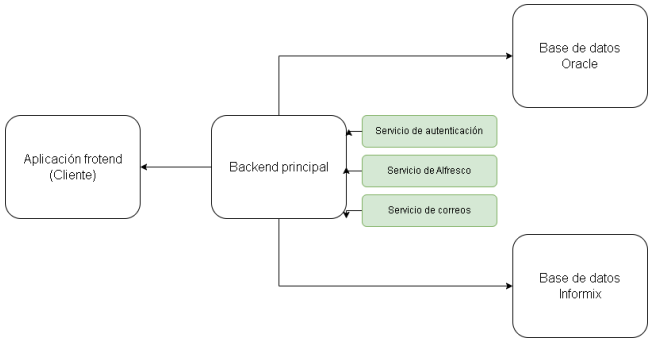
5.3.1 Componentes de la API.

La API principal de la aplicación, es aquella que va conectada con el frontend en cada servicio en particular que este requiera, está compuesta de un backend principal que se conecta a la base de datos ORACLE principal de la universidad y, por medio de un DBLink recupera información de la base de datos Informix para realizar los procesos de validación de registros y adjudicación.

Este diseño del backend se puede observar de manera más clara en la figura 6.

Figura 6.

Estructura lógica de la arquitectura de la API



5.3.2 Componentes del backend.

En función de la arquitectura del backend, se mantiene una relación entre el uso de componentes independientes y autónomos que están interconectados entre sí para darle la funcionalidad esperada a un servicio. Como primera estructura encontramos los controladores. Los controladores son aquellos que definen y reciben las peticiones REST por medio de la red. Estos controladores son fundamentales, porque por medio de este se encuentra la interfaz de comunicación entre el backend y el frontend.

Por otro lado, se crean los servicios, los cuales almacenan toda la lógica de las operaciones tanto transaccionales, como el control de la lógica del negocio que debe poseer la aplicación. Estos servicios, además, apoyan los procesos de validación, depuración, autorización y manejo de la información de maneja especifica y consistente, en función de la funcionalidad requerida.

Además de estos componentes, se crean los repositorios, que son los encargados de interactuar directamente con la base de datos mediante Hibernate. Los repositorios se construyen a partir de su conexión con la representación lógica de las tablas SQL: Los modelos. Estos modelos

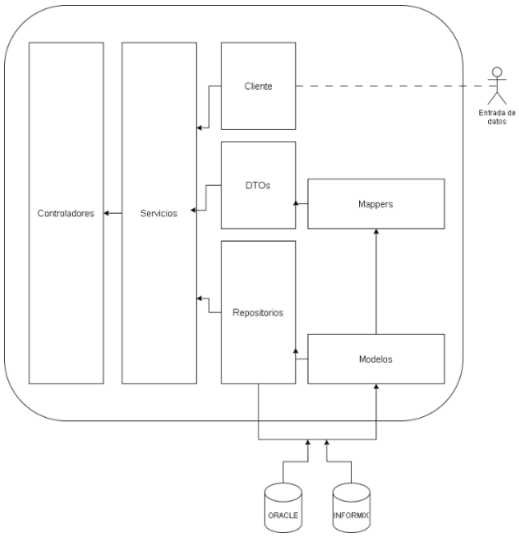
es la representación en forma de una clase Java de una tabla de base de datos, y se utilizan para modelar o mapear los registros de base de datos como objetos en nuestro backend.

Por último, se disponen de objetos auxiliares que cumplen una función similar a los modelos, pero estos funcionan para enviar y recibir información en un formato que sea entendible tanto para el usuario, como para el sistema. Los DTO (Data Transfer Object) son estructuras de datos que se usan para definir los datos que se reciben y se envían entre el cliente y el servidor.

En la figura 7 se visualiza lo mencionado anteriormente.

Figura 7.

Arquitectura general del backend.



6. Metodología

El proceso de análisis, diseño e implementación del sistema de comedores UIS se desarrolló bajo la metodología SCRUM. El equipo de desarrollo enfocado en este proyecto conto con componentes externos a los ya mencionados como autores de este proyecto de grado, sin embargo, estos recursos se añadieron con la finalidad de conservar el estándar de UX (Un diseñador) y de calidad de procesos y producto (QA y tester). Por medio de este enfoque, se fueron realizando entregas parciales que sumaran valor al producto entregable.

Dentro del marco de trabajo metodológico, se definieron unos tiempos de desarrollo basado en objetivos denominados *Sprint*. Estos sprints tenían una duración de 4 semanas cada uno, en donde a partir de backlog (que se entiende como todo el conglomerado de tareas pendientes por realizar), se definían las actividades a realizar y estas entraban en el sprint para realizar como historias de usuario.

En el transcurso del sprint, se llevaron a cabo una serie de reuniones entre el equipo; las cuales permitían analizar los avances, los bloqueantes y las oportunidades de mejora en el proceso de desarrollo. Diariamente, se llevaba a cabo una reunión conocida como Daily, la cual nos permitió identificar avances diarios, bloqueantes y, al mismo tiempo, poder priorizar mejor las tareas y encontrar un punto de equilibrio entre lo que se estaba desarrollando y lo que se está planeando. Además, semanalmente se llevó a cabo una reunión en donde se presentaba los avances de las actividades realizadas en ese periodo de tiempo. Por último, al finalizar cada sprint, se llevaba a cabo una reunión conocida como Retrospective (Retroalimentación, por su traducción del inglés) en donde se compartían las oportunidades de mejora y todo aquello que funciono durante el sprint.

Para garantizar una gestión centralizada de las actividades y, además, contar con un manejo

de tiempos y recursos utilizados en los objetivos planteados durante el sprint, se hizo uso de la herramienta Taiga; la cual nos permitió hacerles seguimientos a las actividades y tener la documentación de lo que se iba desarrollando.

6.1 Implementación

A continuación, se detalla todo el proceso de desarrollo que se llevó a cabo para crear los módulos administrativos, de registro y adjudicación del sistema de comedores. Se iniciará con los respectivos servicios creados desde el backend y, posteriormente, el frontend que se desarrolló para consumir dichos servicios, con la finalidad de cumplir con el alcance esperado.

6.1.1 Backend

El backend desarrollado para el sistema de comedores se construyó haciendo uso del lenguaje de programación Java, con el framework de Spring Boot haciendo uso de la arquitectura mencionada con anterioridad en el marco teórico de este documento. Esta elección de tecnologías se realizó por encontrar en el framework la robustez necesaria para el proyecto, la integración con paquetes y librerías necesarios y por el crecimiento y estabilización que se le da al framework debido a su gran comunidad activa.

A partir de esto, y teniendo en cuenta las necesidades expuestas para este proyecto, se procedió a construir 5 servicios primarios con las funcionalidades requeridas. El nombre y su función se mencionan a continuación.

6.1.1.1 Servicio de configuración transversal Este servicio es el encargado de proveer al sistema todos los datos y procesos necesarios para alcanzar los estados requeridos, tanto en el proceso de registro, como en el proceso de adjudicación. Dentro de él se maneja las funcionalidades orientadas a los estados de una subconvocatoria, los tipos de subconvocatorias, los parámetros del sistema, los tipos de usuarios, los tipos de servicios y la relación entre los datos obtenidos de los estudiantes y los requisitos solicitados para adjudicarlo.

6.1.1.2 Servicio de convocatorias Este servicio ofrece las funcionalidades requeridas para abrir las convocatorias al servicio de comedores semestralmente. Además, permite obtener datos relevantes sobre las convocatorias ya realizadas en semestres anteriores.

6.1.1.3 Servicio de requisitos Este servicio brinda el control sobre los requisitos necesarios, ya sean generales o particulares (Requisitos solo aplicables a un tipo de usuario), que se van a aplicar sobre una convocatoria de comedores. Además, al ser el servicio de comedores un sistema que requiere información transversal de la universidad requiere que sus requisitos estén direccionados hacia los esquemas de información requeridos.

6.1.1.4 Servicio de registro Este servicio es el encargo de realizar toda funcionalidad relacionada con el registro de los estudiantes, por medio de las convocatorias, al sistema de comedores universitarios.

7.1.5 Servicio de adjudicación. Este servicio es el encargado adjudicar el servicio de comedores de manera única o grupal a él o los estudiantes inscritos al servicio de comedores. Se encarga de validar los requisitos, asignarle un grupo especial al estudiante y, posteriormente, incluirlo en el servicio semestral de comedores.

6.1.2 Frontend

6.1.2.1 Generalidades. Este proyecto se desarrolló bajo una serie de estándares que deben

ser conservados en función de obtener el mejor de los resultados a partir de su desarrollo. Es por esto, que antes de explicar cada uno de los módulos desarrollados, se explicaron aspectos generales que cubren el sistema por completo

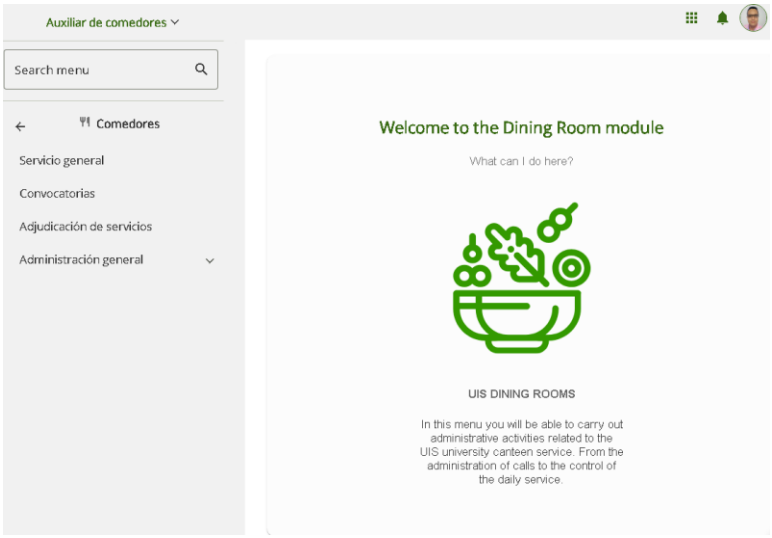
6.1.2.2 Internacionalización. Todos los módulos del proyecto deben contar con un sistema de internacionalización basado en el lenguaje por defecto del usuario. En este caso, todos los proyectos bajo la guía de la DTIC (División de Tecnologías de la Información y Comunicación), están cubiertos por traducciones al idioma inglés y español. Todos estos mensajes se almacenan en base de datos con un identificador único y, a partir de la configuración de idioma del cliente, muestra su mensaje indicado.

6.1.2.3 Guardian de rutas. Cada vez que se ingresa al sistema de comedores, se activa el Guardian de Rutas. Esta funcionalidad se encarga de determinar a qué componentes se pueden acceder dentro del sistema. A partir de esto, se busca restringir el acceso a usuarios que no cuentan con la autorización de hacerlo y desbloquear funcionalidades para aquellos usuarios que deban tener acceso a la misma.

6.1.2.4 Sistema de menús: Cada uno de los sistemas dentro del SIA, cuenta con su propio menú. Estos menús se definen transversalmente para los roles y usuarios que tengan dicho rol asignado. Para el caso del sistema de comedores, se tiene el menú administrativo, el cual es el que dispone el auxiliar de comedores. En la figura 8 se pueden observar los menús disponibles para el control administrativo de comedores

Figura 8.

Menú principal para el rol "Auxiliar de comedores"



Como se puede observar en la figura 8, se cuenta con 4 menús con el nombre de la funcionalidad a la que se puede acceder. Dentro del menú de servicio general se han venido adelantando funcionalidades que no se tienen en cuenta para el alcance de este proyecto. Además, al ser la vista principal, se cuenta con un mensaje de bienvenida y presentación de las funcionalidades que se presentan en el sistema.

Por otro lado, como se puede observar en el menú "Administración general", presenta la característica de controlar internamente los módulos administrativos mencionados anteriormente, los cuales permiten al sistema alcanzar el estado necesario para proceder con la funcionalidad de registro y adjudicación. En la figura 9 se observar estos submenús con sus respectivos nombres alusivos a su funcionalidad.

Figura 9.

Submenús del módulo "Administración general".

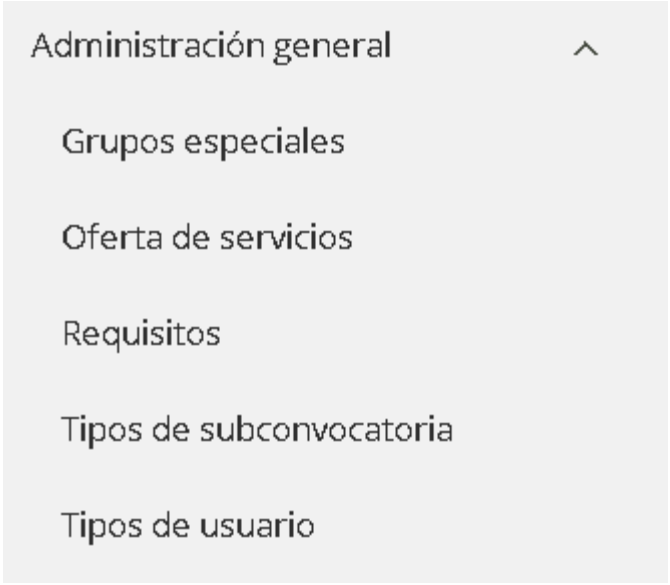
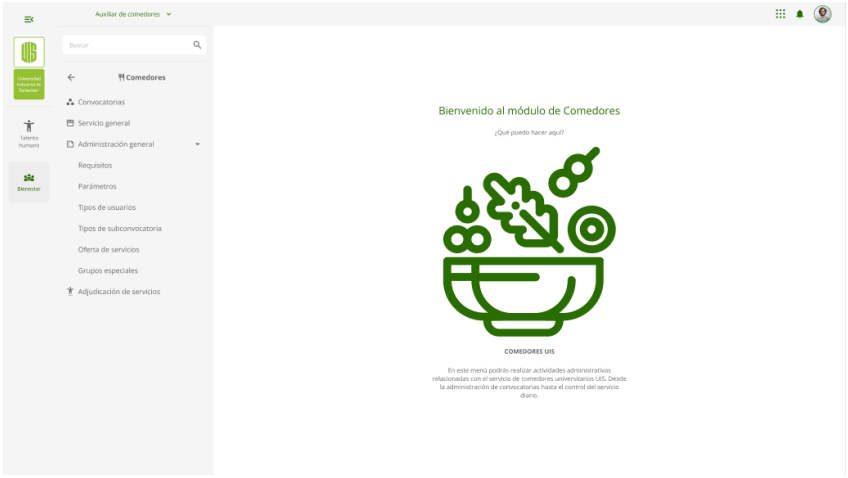
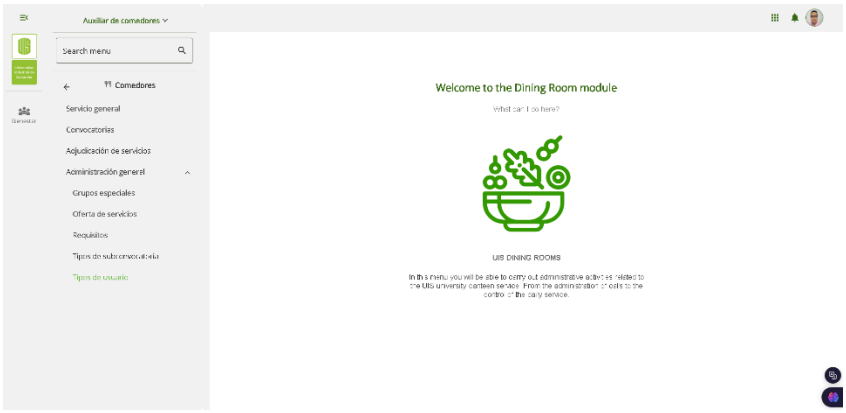


Figura 10.

Comparación entre el diseño (a) y la implementación (b)



(a) Diseño



(b) Implementación

Módulos Administrativos. Una vez el usuario "Auxiliar de comedores" ingresa por primera vez al sistema, debe conocer las necesidades primarias que se necesitan para abrir el proceso de convocatorias. Es por esto por lo que los módulos administrativos se tuvieron en cuenta en este sistema, dando la claridad, que resultan indispensables para cumplir plena y objetivamente con los objetivos de este proyecto. Cada uno de los módulos administrativos se presentarán de manera independiente a continuación.

6.1.3 Tipos de usuario:

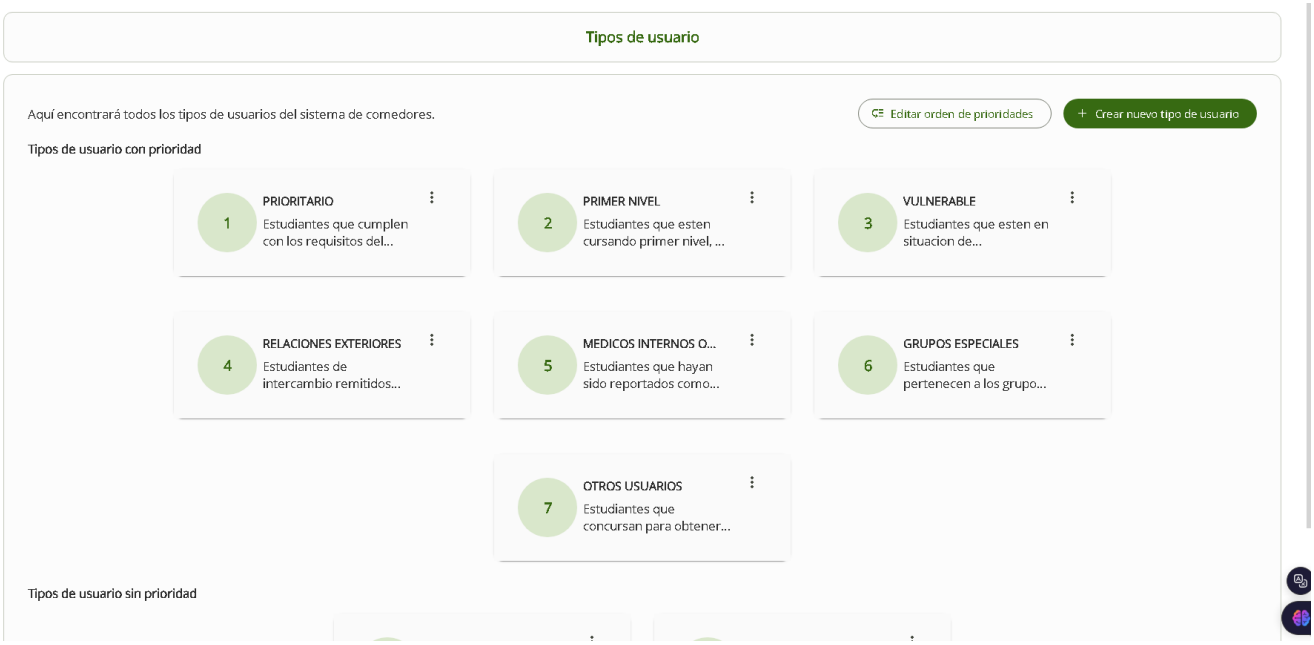
Los tipos de usuario se refieren a las diferentes categorías con las cuales un estudiante puede ser identificado al momento de su adjudicación. El reglamento del servicio de comedores estudiantil establece en el acuerdo 094 del 2016 los tipos de usuarios existentes y sus respectivos requisitos.

Es por esto, que se dispuso una interfaz para controlar este tipo de usuarios y, en el proceso de adjudicación, garantizar que todos los inscritos quede, por lo menos, categorizados en alguno de estos grupos.

En esta interfaz de usuario, se proveen las funcionalidades para la creación, edición, desactivación y priorización de los tipos de usuarios existentes. En la figura 11 se puede observar como el sistema de visualización va acompañado de la prioridad del tipo de usuario, tal como lo establece el reglamento interno de comedores.

Figura 11.

Interfaz administrativa para el control de los tipos de usuarios



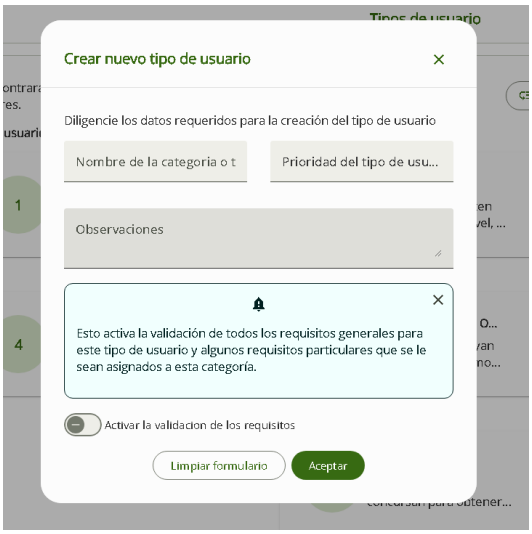
Por otro lado, resulto conveniente disponer en este sistema, una categoría interna dentro de los tipos de usuario, que garantizara la validación de los procesos de adjudicación en la evaluación del cumplimiento de los requisitos mínimos o, conocidos como, requisitos generales.

Es por esto, que en el momento de la creación del tipo de usuario resulta convenientes preguntar si para este usuario se validaran los requisitos. A partir de esto, se evaluarán los requisitos generales y, además, para dicho tipo de usuario será posible la creación de requisitos

particulares. En la figura 12 podemos observar el formulario de creación de tipos de usuarios.

Figura 12.

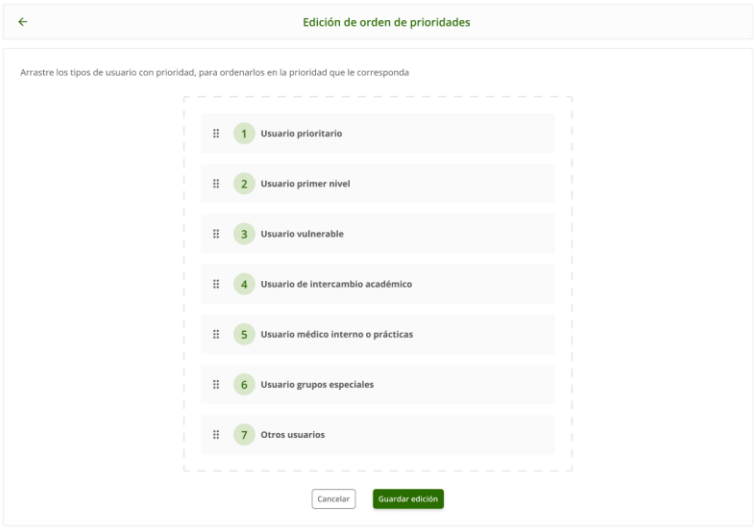
Formulario de creación de tipos de usuario.



Complementariamente, la prioridad de los tipos de usuario determina cuáles serán los estudiantes que, al ser categorizados como cierto tipo de usuario, cuenta con mayor probabilidad de obtener el servicio de comedores. La reorganización de prioridades, en el caso de que ya existan los tipos de usuarios creados, resulta un proceso complicado por la cantidad de validaciones que tiene asignar un cierto tipo de usuario a una prioridad ya ocupa. Por tal razón, se implementó un componente Draggable, el cual le permite al usuario realizar la reasignación de prioridades sin caer en lógicas complejas o reprocesos. En la figura 13, se puede observar la implementación de dicho componente.

Figura 13.

Sistema draggable de orden de prioridades para los tipos de usuarios



6.1.4 Requisitos.

Los requisitos de adjudicación son aquellos que permiten categorizar a un estudiante como cierto tipo de usuario y, además, garantizar que hace parte de la comunidad universitaria objetivo para adquirir este servicio. Como se había mencionado anteriormente, los requisitos se dividen en dos categorías puntuales, unos son los requisitos generales y los otros los requisitos particulares. Los requisitos generales son aquellos requisitos que aplican para todo usuario inscrito al sistema; mientras que los requisitos particulares, solamente aplican para cierto tipo de usuario y son la garantía del carácter diferenciador entre un estudiante y otro. Para la administración de requisitos se desarrolló una interfaz entendiendo dichas categorías y permitiendo optimizar al máximo su funcionalidad y escalabilidad. En la figura 14 se observa la interfaz de consulta de requisitos.

Figura 14.

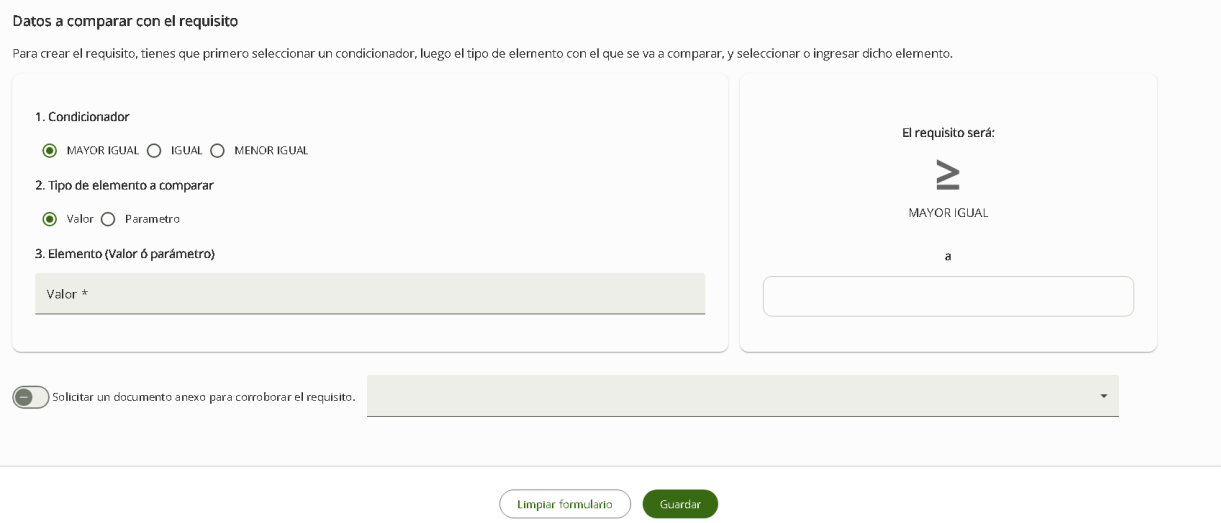
Interfaz de consulta de requisitos de adjudicación

En la figura 14 se observa la diferenciación de categorías y, por otro lado, cada una de ellas presenta su propio botón de creación para agregar un requisito en particular. Para el proceso de creación se solicitan datos básicos del requisito, como su nombre, descripción, esquema al que apunta, entre otros. Además, al entender la naturaleza cambiante y dependiente del sistema de comedores, el uso de parámetros es fundamental. Por consecuente, se hace uso de parámetros asociados, por ejemplo, al salario mínimo mensual legal vigente (SMMLV) para tomar referencias puntuales de, por ejemplo, la matricula máximo que puede pagar un estudiante perteneciente a comedores. En la figura 15 se puede observar la interfaz de creación de requisitos.

Figura 15.

Formulario de creación de requisitos

(a) Datos básicos del requisito



(b) Información adicional del requisito

Al momento de la creación de un requisito, este inmediatamente redirigirá a la interfaz de consulta y, dependiendo de si es un requisito general o particular, se visualizará en su respectiva sección. En la figura 16 se observa un requisito general cargado desde la interfaz de consulta.

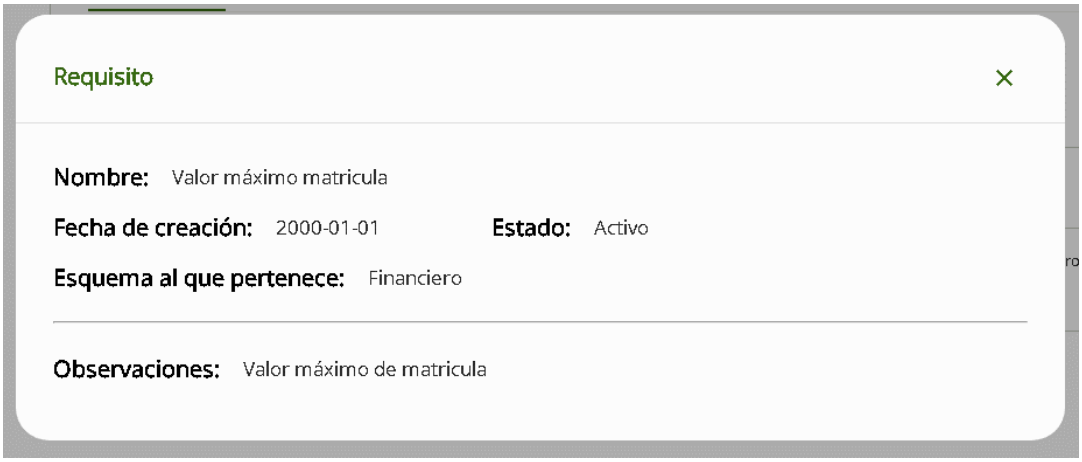
Figura 16.

Visualización de los requisitos generales

Desde esta pantalla, se brindan las acciones básicas de consulta, edición y eliminación. Además, con el fin de tener los detalles básicos del requisito, se dispone de una ventana modal para ver dicha información requerida. La figura 17 muestra el modal de visualización de requisitos.

Figura 17.

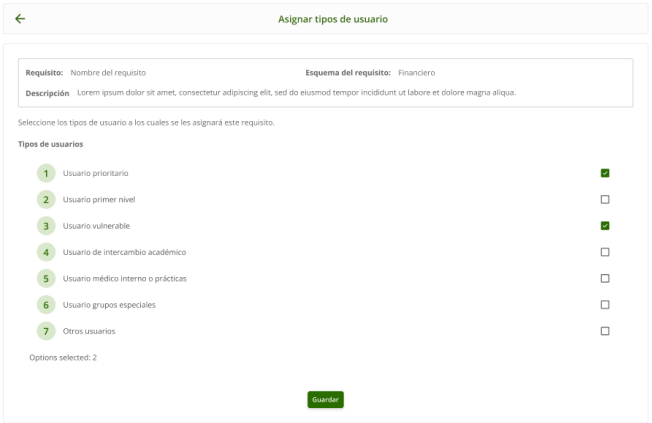
Consulta de datos básicos de un requisito



Adicionalmente, para los requisitos particulares se crea una opción adicional, la cual permite asignar los tipos de usuarios a los cuales se les aplicara este requisito en el momento de la adjudicación. En la figura 18 se muestra la pantalla de asignación de tipos de usuario a un requisito particular.

Figura 18.

Pantalla de asignación de tipos de usuario a un requisito particular



6.1.5 Oferta de servicios

Este módulo permite tener un control administrativo sobre los servicios que se prestan dentro de comedores. Actualmente, se establece la oferta de 3 servicios alimenticios diarios. El desayuno, el almuerzo y la cena son estos servicios que se prestan por comedores, y para cada uno de ellos, se establece un horario de servicio y un valor de ración; el cual, a futuro, permitirá generar los recibos de pagos para los estudiantes. En la figura 19 se puede observar la interfaz administrativa de la oferta de servicios.

Figura 19.

Interfaz administrativa de los servicios ofrecidos.

6.1.6 Tipos de subconvocatorias.

Este módulo presenta los tipos de subconvocatorias creados a partir del diseño de registro de inscripciones a los estudiantes. A partir de esta interfaz se podrán administrar los tipos de subconvocatorias posibles en un proceso de inscripciones semestral. Cada una de estas subconvocatoria cuenta con condiciones básicas que determinan el flujo de información de los estudiantes en sus respectivas inscripciones. En la figura 20 se puede observar la interfaz

administrativa de las subconvocatorias

Figura 20.

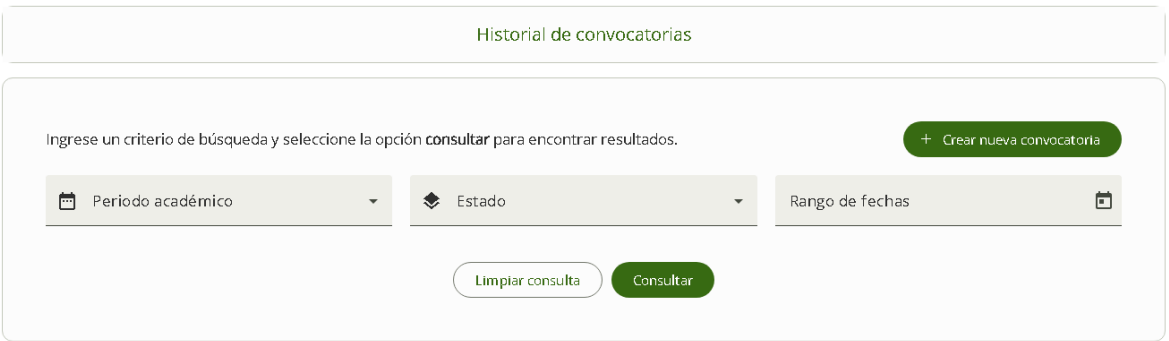
Interfaz administrativa de las subconvocatorias

6.1.7 Convocatorias

En este módulo, el auxiliar de comedores podrá tener información de las convocatorias anteriores y, además, abrir proceso de convocatorias semestrales si así lo requiere. Las convocatorias, las cuales son la puerta de entrada al sistema, son la base para el proceso de registro de los estudiantes al sistema de comedores. Es por esto, que el sistema genera una guía constante a los usuarios para realizar el proceso de creación y, requerido el caso, ver cómo fueron convocatorias pasadas. En la figura 21 se puede observar la interfaz de consulta de convocatorias para el sistema de comedores.

Figura 21.

Interfaz de consulta y administración de convocatorias al sistema de comedores.



Dentro del sistema, se da la capacidad de realizar la consulta a partir de diferentes criterios, tales como, el periodo académico, el estado de la convocatoria y un rango de fechas. Cada uno de estos criterios ajusta el filtrado de convocatorias y, además, permite sectorizar la búsqueda a convocatorias vigentes, cerradas o en un estado de espera. En la figura 22, se presenta una consulta para el periodo académico 2024-1 a partir del primer criterio de consulta.

Figura 22.

Consulta de la convocatoria del periodo académico 2024-1.

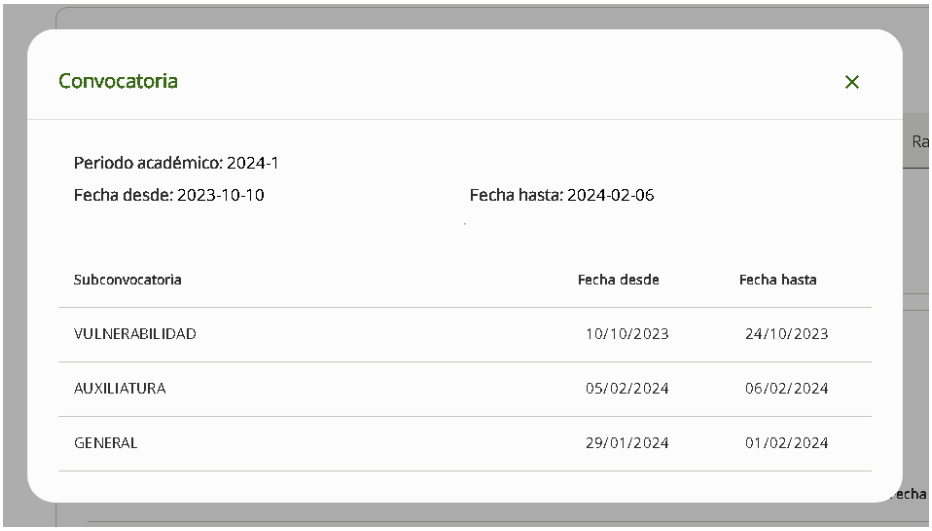
A partir de la figura 22, se observan las diferentes acciones que marcan una convocatoria puntual. Estas opciones van desde la visualización de datos específicos de la convocatoria deseada, hasta acciones de edición y eliminación. Estas opciones, como se evidencia en la figura 22, no siempre se encuentran habilitadas. La disponibilidad de estas acciones está determinada por el estado de la convocatoria. El estado de la convocatoria es la forma en el que el sistema conoce si una convocatoria está en su fase inicial, si está en proceso o ya finalizo. Todo estado diferente a

finalizado posee una serie de condicionantes adicionales para la edición y eliminación de las convocatorias. Por ejemplo, siempre y cuando una convocatoria no tenga ningún estudiante registrado y sus fechas de inscripción no hayan iniciado, podrá ser eliminada.

Con la finalidad de seguir con el ejemplo de la convocatoria para el semestre académico 2024-1, se hará uso de la primera acción para ver el detalle de dicha convocatoria. En la figura 23 se observan los detalles de la convocatoria, ajustados a la propuesta de subconvocatorias ya realizada.

Figura 23.

Detalle de la convocatoria para el semestre académico 2024-1



Esta opción de visualización brinda una vista particular de las subconvocatorias realizadas dentro de la convocatoria general. Este detalle permite conocer las fechas de registro de los estudiantes y brindar una mejor experiencia de usuario al, en un mismo lugar, poder consultar todas las “convocatorias” realizadas para un semestre académico. Esta implementación reduce sustancialmente los pasos que el usuario realiza al consultar dicha información en el sistema actual, debido a que, para acceder a la misma información, dispone de 4 menús diferentes para consultar

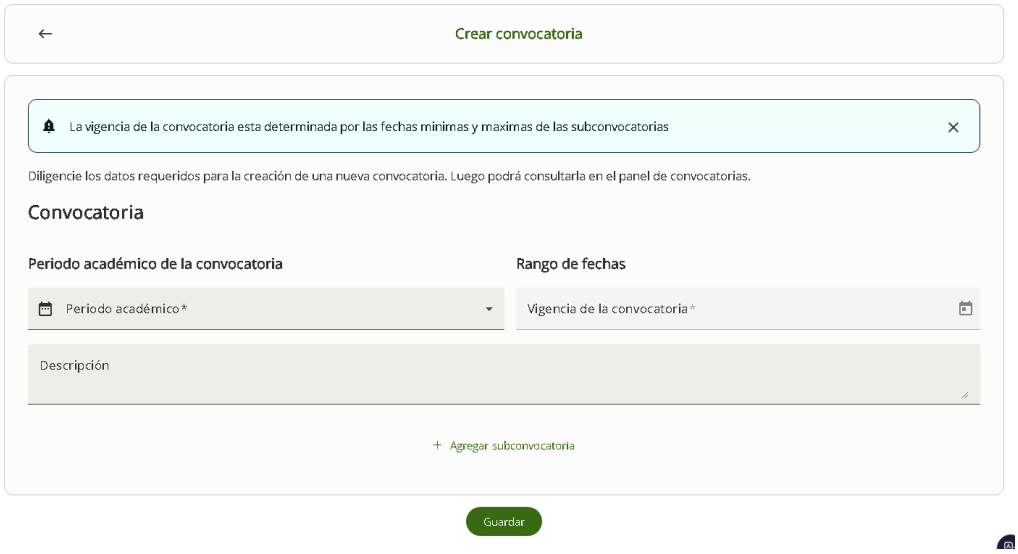
cada una de estas “convocatorias”.

Por otro lado, la funcionalidad de creación de convocatorias debe disponer una serie de validaciones y, además, permitir al usuario entender con facilidad el proceso de ahora crear una única convocatoria con sus respectivos periodos de tiempo para cada una, en lugar de 3 convocatorias con información duplicada y desde secciones diferentes de la plataforma.

En la figura 24 se puede observar el formulario de información básica para la creación de la convocatoria de comedores.

Figura 24.

Formulario de información básica de la convocatoria.

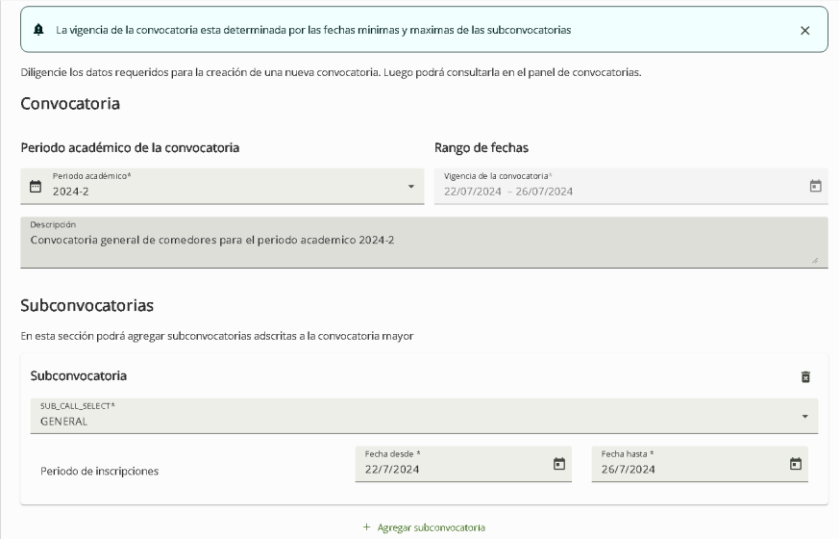


Como se puede observar en la figura 24, principalmente se piden datos básicos como el periodo académico y la descripción de la convocatoria. Sin embargo, se puede observar una casilla de nombre “Rango de fechas”, la cual la información se ira guardando automáticamente dependiendo de la vigencia de las subconvocatorias.

En la figura 25, se puede observar la funcionalidad de agregar subconvocatoria a la convocatoria y, a partir de esto, como se va actualizando la vigencia de esta.

Figura 25.

Formulario de creación de convocatoria con la subconvocatoria general agregada.



.

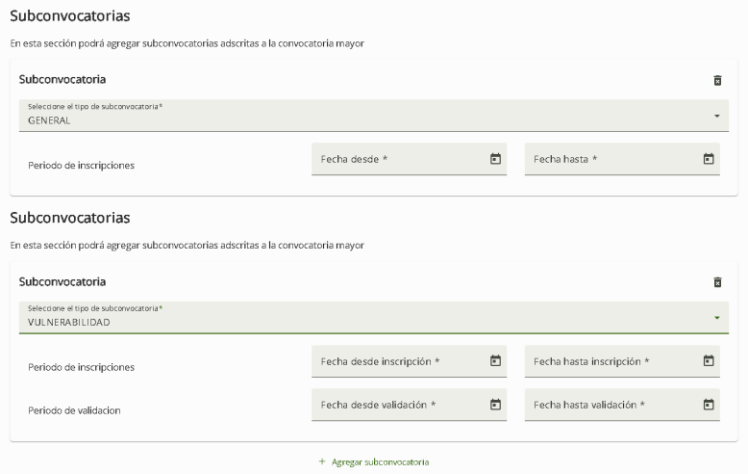
Por consecuencia, el usuario podrá manipular la información de vigencias para cada subconvocatoria según sean las necesidades y/o indicaciones dadas a partir de las fechas de los calendarios académicos. El formulario está dispuesto de tal manera que únicamente le permita asociar una subconvocatoria por cada subconvocatoria general, esto definido de esta manera a partir del conocimiento obtenido del acuerdo 094 del 2016, que establece el funcionamiento interno de comedores.

Por último, dentro del proceso de creación de convocatoria, existe un tipo de subconvocatoria que difiere con las demás, esta es la subconvocatoria por vulnerabilidad. Esta subconvocatoria no solamente requiere de las fechas de registro o inscripción de los estudiantes, sino que, además, requiere de un periodo de validación por parte de funcionarios de bienestar

estudiantil; esto con el fin de determinar si un estudiante registrado en esta subconvocatoria puede ser considerado, o no, como un estudiante en situación de vulnerabilidad académica. Es por esto, que cuando se va a agregar dicha subconvocatoria, el auxiliar de comedores debe establecer las fechas de inscripciones y las fechas de validación de documentos. En la figura 26 se puede observar y comparar las diferencias entre el formulario interno de una subconvocatoria general y una subconvocatoria de vulnerabilidad.

Figura 26.

Comparación del formulario de subconvocatoria general y subconvocatoria de vulnerabilidad.



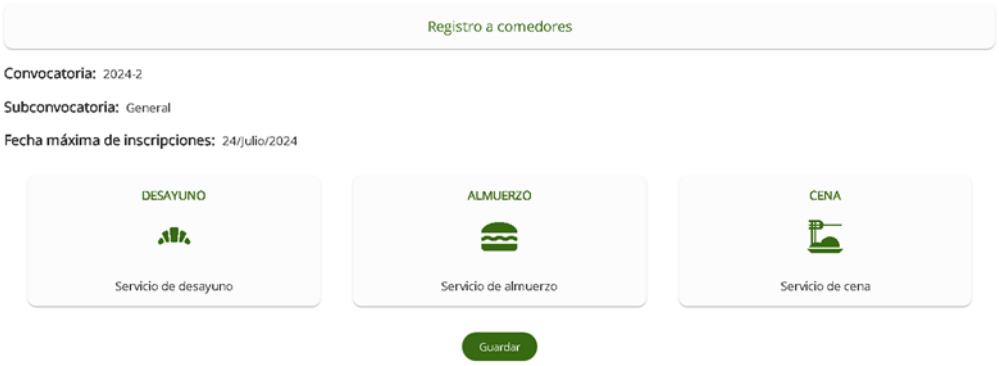
6.1.8 Registro.

Este módulo se encarga de obtener la información básica de la inscripción que un estudiante realiza al servicio de comedores. Principalmente consiste en guardar un registro único por estudiante y por convocatoria, el cual es el agrupador de los detalles del registro. Un detalle del registro es a cuál de los servicios ofrecidos el estudiante quiere participar, de esta manera, el estudiante selecciona los servicios a los que quiere participar y para cada uno de ellos se crea un detalle de inscripción. Este detalle es importante, debido a que a partir de esto se asociaran el tipo

de usuario respectivo y se hará el proceso de adjudicación. En la figura 27 se puede observar la interfaz de registro de estudiante, en donde se evidencian los datos básicos de la convocatoria, subconvocatoria y los servicios disponibles a adquirir.

Figura 27.

Pantalla de inscripción de los estudiantes a una convocatoria.



6.1.9 Adjudicación

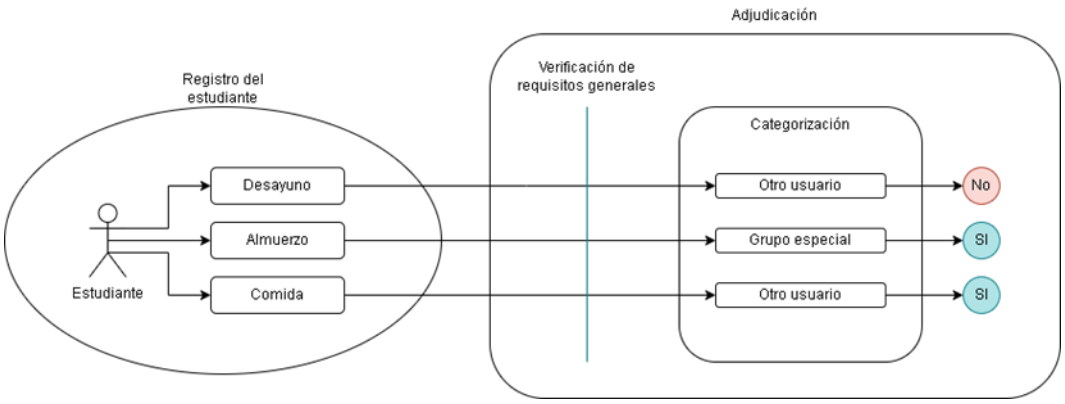
El módulo de adjudicación recupera a los estudiantes inscritos en una convocatoria en particular y evalúa puntualmente los requisitos necesarios para que puedan ser adjudicados como usuarios semestrales del servicio de comedores. Este proceso consiste en primeramente evaluar los requisitos generales del sistema para determinar si un estudiante debe ser categorizado. Estos requisitos generales son aquellos definidos con anterioridad desde el módulo de requisitos de adjudicación; y, a partir de esto, se procede a categorizar a los estudiantes en su respectivo grupo o tipo de usuario.

Por consecuente, vale la pena aclarar que para un mismo usuario puede pertenecer a grupos o tipos de usuario diferentes por cada servicio que haya solicitado. El proceso de selección y categorización genera la escala por cada detalle registrado por el estudiante, mas no de forma generalizada. En la figura 27 se muestra un gráfico que permite determinar el modelo lógico del

proceso de adjudicación por medio de un registro concreto.

Figura 28.

Modelo lógico del proceso de adjudicación.



A partir de la figura 27 se define el proceso que va a adjudicar a un estudiante para cierto servicio solicitado. Primeramente, se hace una validación de los requisitos mínimos y generales que debe cumplir; si por algún motivo el estudiante no cumple con alguno de los criterios, no podrá ser adjudicado para ninguno de los servicios solicitados.

Una vez se valide que el estudiante cumple con los requisitos generales, se interpretara al estudiante como un usuario (sin adjudicar) por cada servicio solicitado. Una vez se encuentre en este punto, se realiza un proceso de categorización a partir de un requisito eliminatorio. El requisito eliminatorio es aquel requisito único para cada tipo de usuario y que descarta de manera inmediata si un estudiante pertenece al mismo. De la misma forma vale aclarar, que para cada detalle solamente será posible ser categorizado en un único tipo de usuario; es por esto, que, si un usuario resulta categorizado en más de un tipo de usuario, se le asignara el que mayor prioridad tenga.

A partir de esto, una vez se categorice uno de los detalles, los otros heredaran la categoría de este; exceptuando únicamente al servicio de desayuno, el cual no puede ser adjudicado

para tipos de usuario que hagan parte de grupos culturales y selecciones deportivas. Sin embargo, si podrán ser adjudicados si poseen otra categoría diferente.

Posteriormente, se obtienen tres factores diferenciadores para realizar un ordenamiento dentro del sistema. Como primer factor se tendrá en cuenta el promedio acumulado, dando a entender que aquellos estudiantes con mayor promedio acumulado tendrán un mejor ranking dentro del grupo asignado y, por ende, mayor su posibilidad de ser adjudicado. Como segundo factor, se tiene en cuenta el número de créditos aprobados por el estudiante, y por último, el número de créditos matriculados para el semestre académico vigente.

Finalmente, durante el proceso de adjudicación se determinan un numero de cupos a asignar por servicio; los cuales serán repartidos respetando la prioridad de los tipos de usuarios y el orden interno que posean los estudiantes.

7. Validaciones

El proceso de validación de la calidad del producto a entregar se basó en, primeramente, realizar test unitarios, tanto a la capa backend como a la capa frontend, y las pruebas realizadas por el equipo de QA (Quality Assurance, por sus siglas en ingles). Estas pruebas se realizan con la finalidad de seguir el estándar de calidad de la DTIC en el software desarrollado.

Las pruebas unitarias desarrolladas en el backend se hicieron dando uso del framework JUnit y Mockito. Estas herramientas permitieron automatizar y establecer los escenarios posibles durante el proceso de desarrollo de los servicios backend implementados. A partir de esto, se buscó mantener un coverage (Cobertura de código testeado) en un nivel alto, esto como un indicador de buena calidad del producto a partir de la respuesta que el servicio da a ciertos casos de prueba.

Por otro lado, las pruebas unitarias desarrolladas desde el frontend se realizaron haciendo uso del framework Jasmine. Este framework, al igual que JUnit y Mockito, permitieron automatizar y realizar pruebas aisladas para cada uno de los componentes construidos en el sistema de comedores. Además, también se implementaron con la finalidad de testear el funcionamiento de los módulos y componentes propios de Angular, que se ajustaran de manera correcta a lo requerido por este equipo de desarrollo para llevar a cabo las implementaciones planteadas.

Finalmente, el equipo de QA se encargó de realizar escenarios de prueba, guiados por la perspectiva de un usuario totalmente nuevo, lo cual permitió encontrar mejoras dentro las distintas interfaces e implementaciones realizadas. A partir de esto, y haciendo uso de la metodología SCRUM, se realizaron unos reportes, conocidos como issues, que se tuvieron en cuenta dentro de la planeación de 4 semanas para mejorar el producto desarrollado.

A continuación, se da una explicación detallada las pruebas realizadas por el equipo de desarrollo en el proceso de implementación.

7.1 Test Unitarios

El desarrollo de pruebas unitarias fue una etapa crucial en la implementación de los módulos de Registro y Adjudicación del Servicio de Comedores Universitarios de la UIS. Estas pruebas se diseñaron para asegurar que cada unidad funcional del código cumpliera con los requisitos especificados y funcionara de manera correcta e independiente. Para lograr este objetivo, se empleó el framework JUnit, el cual es ampliamente reconocido en el entorno de desarrollo en Java por su eficiencia y facilidad de uso.

Adicionalmente, se utilizó Mockito para la creación de objetos simulados (mock objects), permitiendo así la realización de pruebas más aisladas y precisas al evitar dependencias externas. La combinación de estas tecnologías facilitó la detección temprana de errores y contribuyó a la mejora continua del código, asegurando una alta calidad en el software desarrollado.

En el desarrollo del frontend, se implementaron una serie de pruebas unitarias con el objetivo de garantizar la funcionalidad y estabilidad de los componentes de la aplicación. Para ello, se utilizaron las herramientas Karma y Jasmine, ampliamente reconocidas en el desarrollo de aplicaciones Angular. Jasmine, un framework de pruebas para JavaScript, se utilizó para escribir y estructurar las pruebas, permitiendo una sintaxis clara y fácil de entender. Karma, por su parte, actuó como un servidor de pruebas, ejecutando las pruebas en diversos navegadores y proporcionando un entorno de pruebas automatizado y continuo. La integración de estas tecnologías permitió validar el comportamiento de los componentes del frontend, asegurando que cada parte del código funcionara correctamente y cumpliese con los requisitos especificados, contribuyendo así a la calidad y robustez del sistema final.

7.1.1 Backend.

En el backend, se llevaron a cabo las pruebas unitarias para cada uno de los 5 servicios expuestos anteriormente. Dentro de cada servicio se desarrollaron las funcionalidades necesarias para que el sistema funcionara de manera adecuada. Por consiguiente, se realizaron un total de 35 test unitarios, enfocados en cada una de las funciones internas de los servicios, haciendo el análisis de los casos ideales, y posibles casos de fallo. En el anexo 1 se listan las pruebas unitarias para cada uno de los servicios implementados.

7.1.2 Frontend.

En el frontend, se llevaron a cabo las pruebas unitarias para los componentes y servicios desarrollados. Cada uno de los 22 componentes y servicios construidos cuenta con mínimamente una prueba unitaria para garantizar el correcto funcionamiento y uso, ya sea del componente o del servicio. En el anexo 2 se listan algunas de las pruebas realizadas sobre los principales componentes implementados.

8. Conclusiones

El equipo de desarrollo levanto requerimientos, diseño e implemento las funcionalidades primarias del sistema de comedores estudiantiles. En este momento, los usuarios de comedores tienen la capacidad de fundamentar el sistema, a partir de los componentes administrativos primarios, que servirán de base para migrar sus actividades administrativas al nuevo sistema.

En este sentido, se realizó un diseño escalable a partir de una visión general del sistema de comedores, el cual permitió implementar con éxito los módulos de registro y adjudicación del sistema de comedores. Además, se avanzó de gran manera en los módulos administrativos, los cuales son un factor fundamental dentro de los sistemas de bienestar estudiantil. Por otro lado, se realizaron las respectivas migraciones de convocatorias del sistema de “Nuevas Versiones” a la base de datos Oracle respetando el nuevo diseño y estructura desarrollado.

Por otro lado, el proceso de desarrollo de software se fundamentó en el seguimiento de la metodología, buenas prácticas y capacidad de escalabilidad del software, lo cual es uno de los factores fundamentales del proyecto de renovación de los sistemas de información. Además, el uso e implementación de pruebas unitarias, garantizo la realización de código funcional y desarrollos oportunos para atender las diversas problemáticas que se plantearon en el transcurso de este proyecto.

De igual manera, se realizó un acercamiento a la integración de los procesos y la información del sistema actual de comedores y este nuevo sistema. A partir de esto, se desarrolló el primer sistema asociado a bienestar estudiantil, lo cual abre un panorama nuevo para iniciar con el proceso de renovación e integración de los sistemas antiguos, a una visión novedosa, escalable y sostenible de los nuevos sistemas de la Universidad Industrial de Santander.

9. Trabajo futuro

Como primer foco de trabajo, se busca diseñar y desarrollar todos los procesos primarios faltantes al nuevo sistema, entre estos esta la realización del servicio diario de comedores para cada uno de sus servicios, con lo que se podrá tener la información diaria del funcionamiento del servicio de comedores en el nuevo sistema y generar procesos automatizados, tales como, la evaluación de requisitos diarios, el proceso de preinscritos (Estudiantes inscritos que no cumplían con los requisitos generales pero que ahora si los cumplen) y estadísticas de la población adjudicada.

En segundo lugar, se busca integrar el funcionamiento del servicio de "Combo estudiantil" dentro del sistema, dando la facilidad de comprar el combo por medio de la aplicación estudiantil en el apartado de bienestar estudiantil. De igual manera, la generación y pago de recibos desde el sistema, lo cual ayudara a reducir significativa los índices de estudiantes que consumen y no realizan el pago del servicio en las fechas estipuladas.

Por último, brindar a los estudiantes la capacidad de reportar excusas permanentes y esporádicas dentro del sistema. Lo cual centralizara el proceso de verificación de requisitos, fallas y otras causales de pérdida del servicio de comedores.

Referencias Bibliografícas

Altvater, A. (2024, 4 marzo). *Gradle vs. Maven: Performance, Compatibility, Builds, & More*. Stackify. https://stackify.com/gradle-vs-maven/#:~:text=Gradle%20and%20Maven%20fundamentally%20differ,%2C%20are%20the%20%E2%80%9Cworkhorses.%E2%80%9D

Angular. (s. f.). https://angular.io/guide/what-is-angular

Arunodi, N. (2024, 21 marzo). Top 10 Angular Component libraries. *Syncfusion*. https://www.syncfusion.com/blogs/post/top-angular-component-libraries

Atlassian. (s. f.-b). *Arquitectura de microservicios | Atlassian*. https://www.atlassian.com/es/microservices/microservices-architecture

Choksi, K. (2023, 5 junio). How to Write Unit Tests with Jasmine & Karma? | by Khushbu Choksi | Medium | Simform Engineering. *Medium*. https://medium.com/simform-engineering/how-to-write-unit-tests-with-jasmine-karma-f1908bdeb617

Codecademy. (s. f.-b). *What is REST?* Codecademy. https://www.codecademy.com/article/what-is-rest

Escalabilidad en proyectos de software: ¿Por qué es crucial para el éxito a largo plazo? (s. f.). Kodigo.org. https://kodigo.org/escalabilidad-en-proyectos-de-software-por-que-es-crucial-para-el-exito-a-largo-plazo/

Gaba, I. (2023, 13 enero). *What is Maven: Here's What You Need to Know*. Simplilearn.com. https://www.simplilearn.com/tutorials/maven-tutorial/what-is-maven

Getting Started | Building REST services with Spring. (s. f.). Getting Started | Building REST Services With Spring. https://spring.io/guides/tutorials/rest

Informix Servers 12.10. (s. f.). https://www.ibm.com/docs/es/informix-

servers/12.10?topic=tools-informix-4gl

Introduction | Yarn. (s. f.). https://yarnpkg.com/getting-started

JUnit 5. (s. f.). https://junit.org/junit5/

Kale, A. (2023b, septiembre 3). How to implement Hibernate in Spring Boot - Javarevisited - Medium. *Medium*. https://medium.com/javarevisited/how-to-implement-hibernate-in-spring-boot-69e4f10d0b80#:~:text=Hibernate%20is%20an%20ORM%20(Object,in%20a%20Spring%20Boot%20Project.

Lemonaki, D. (2022, 18 marzo). *Frontend VS Backend – What's the Difference?* freeCodeCamp.org. https://www.freecodecamp.org/news/frontend-vs-backend-whats-the-difference/#feintro

MVC - MDN Web Docs Glossary: Definitions of Web-related terms | MDN. (2023c, diciembre 20). MDN Web Docs. https://developer.mozilla.org/en-US/docs/Glossary/MVC

OpenFeign. (s. f.). *GitHub - OpenFeign/feign: Feign makes writing java http clients easier*. GitHub. https://github.com/OpenFeign/feign

Ramotion. (2023, 2 mayo). *MVC Architecture: Simplifying web application development*. Web Design, UI/UX, Branding, And App Development Blog. https://www.ramotion.com/blog/mvc-architecture-in-web-application/

Ricardo, & Ricardo. (2023, 2 noviembre). *PRUEBAS UNITARIAS con MOCKITO en JAVA*. Programando En Java. https://programandoenjava.com/mockito-pruebas-unitarias/

Spring boot. (s. f.). Spring Boot. https://spring.io/projects/spring-boot

Spring Framework Guru. (2019, 2 junio). *Decorator pattern - Spring Framework guru*. Spring Framework Guru - Become A Spring Framework Guru.

https://springframework.guru/gang-of-four-design-patterns/decorator-pattern/

Team, A. C. (s. f.). *Angular material*. Angular Material. https://material.angular.io/

UIS. (2024) *Acuerdo 094 de 2016: Reglamento para la prestación de los servicios ofrecidos a los estudiantes por la sección de Bienestar Universitario*. Consultado el 13 de abril del 2023, desde https://uis.edu.co/uis-comedores-es/

What are microservices? (s. f.). microservices.io. https://microservices.io/

What is an API? (Application Programming Interface) | MuleSoft. (s. f.-b). MuleSoft. https://www.mulesoft.com/resources/api/what-is-an-api

What is Unit Testing? - Unit Testing Explained - AWS. (s. f.-c). Amazon Web Services, Inc. https://aws.amazon.com/what-is/unit-testing/#:~:text=Unit%20testing%20is%20the%20process,test%20for%20each%20code%20unit.

Why do we need package manager for front end development? (s. f.). Quora. https://www.quora.com/Why-do-we-need-package-manager-for-front-end-development

Apéndices

Apéndice A. Tabla de pruebas unitarias en el backend

Tabla 2

Pruebas unitarias para el servicio de registro

RegistroServiceImpl	
Método	**Lista de pruebas unitarias**
findRegistrosByIdConvocatoria	• Verificar el funcionamiento de la capa Repository al proveer una convocatoria inexistente. • Verificar una respuesta adecuada cuando la convocatoria no tiene registros asociadas. • Verificar si el método retorna la información de registros. • Verificar que todos los registros realizar tuvieran mínimamente un detalle registrado
findRegistro	• Verificar que el método validara los parámetros enviados como actos o no. • Verificar que la información del usuario este vinculada a un estudiante activo • Verificar que se devuelva un único registro por estudiante y convocatoria. • Verificar que, si existe un registro, este tenga mínimamente un detalle registrado
create	• Verificar que se realice el registro para el estudiante en una convocatoria en particular.

Tabla 3-a

Pruebas unitarias para el servicio de convocatorias

ConvocatoriasServiceImpl	
Método	**Lista de pruebas unitarias**
findConvocatoriasVigentes	• Verificar el estado existente “Activo” como un parámetro de la convocatoria • Verificar un comportamiento adecuado del método para el caso donde no haya ninguna convocatoria vigente • Verificar que el método solamente retorne una convocatoria vigente.
findConvocatoriasEntreFechas	Verificar que el orden de las fechas no sea opuesto o genere inconsistencias. Verificar que el método responda adecuadamente a la sobreposición de fechas Verificar que el método retorne todas las convocatorias en la vigencia de las fechas
create	Verificar el adecuado funcionamiento del método cuando ya existe una convocatoria en el semestre académico. Verificar el adecuado funcionamiento del método cuando se registra más de un mismo tipo de subconvocatoria. Verificar el adecuado funcionamiento cuando no crean subconvocatorias Verificar el funcionamiento cuando se crea una convocatoria

Tabla 3-b

Continuación de prueba unitarias para el servicio de convocatorias

ConvocatoriasServiceImpl	
Método	**Lista de pruebas unitarias**
findAll	• Verificar que el método encuentre todas las convocatorias existentes.
search	• Verificar los parámetros de entrada de la consulta • Verificar que a partir de los parámetros se obtenga una respuesta esperada.
validatePeriodoAcademico	• Verificar que exista una única convocatoria por periodo académico. • Validar que el servicio general este inmerso en las fechas del periodo académico.
createSubconvocatoria	• Verificar que se cree la subconvocatoria una vez exista la convocatoria principal. • Verificar que la subconvocatoria no se cree si a existe una del mismo tipo.

Tabla 4

Pruebas unitarias para el servicio de configuración

ConfigComedoresServiceImpl	
Método	**Lista de pruebas unitarias**
getAllEstados	• Verificar que existan los estados de la convocatoria. • Verificar que los estados estén vigentes.
getAllComparadares	• Verificar que existan los comparadores de los requisitos • Verificar que los comparadores estén vigentes
getAllParametros	• Verificar que existan los parámetros asociados a bienestar. • Verificar que los parámetros estén vigentes. • Verificar que los parámetros tengan un valor asociado.
GetAllTipos Subconvocatoria	• Verificar que existan los tipos de subconvocatorias. • Verificar que los tipos de subconvocatoria estén vigentes.

getAllTiposUsuarios	• Verificar que existan los tipos de usuario de comedores. • Verificar que los tipos de usuario estén vigentes.
getAllOfertaServicios	• Verificar que existan los servicios de comedores. • Verificar que los servicios estén vigentes.

Tabla 5-a

Pruebas unitarias para el servicio de requisitos

RequisitosServiceImpl	
Método	**Lista de pruebas unitarias**
findRequisitoById	• Verificar la existencia de un requisito a partir de su id. • Verificar que un requisito no existe a un identificador provisto.
findAll	• Verificar la obtención de todos los requisitos de adjudicación. • Verificar que todos los requisitos sean vigentes.
findRequisitosGenerales	• Verificar la obtención de los requisitos generales. • Verificar que los requisitos sean vigentes. • Verificar que sean transversales a todos los tipos de usuarios.
findRequisitosParticulares	• Verificar la obtención de los requisitos particulares. • Verificar que los requisitos sean vigentes.

	• Verificar que cada uno tenga sus respectivos tipos de usuario asignados.
crearRequisito	• Validar que el requisito no tenga mismo nombre que uno existente. • Validar que el requisito tenga un valor o parámetro. • Validar que el requisito se cree correctamente.

Tabla 5-b

Continuación de pruebas unitarias al servicio de requisitos

RequisitosServiceImpl	
Método	**Lista de pruebas unitarias**
updateRequisito	• Verificar que el requisito a actualizar exista. • Verificar que no se actualice a un nombre existente. • Verificar la actualización correcta del requisito
eliminarRequisito	• Validar que no sea un requisito utilizado en alguna adjudicación. • Verificar que se haya eliminado correctamente.

Tabla 6

Pruebas unitarias para el servicio de adjudicación

AdjudicacionServiceImpl	
Método	**Lista de pruebas unitarias**
adjudicarEstudiante	• Verificar que no se adjudique el estudiante si no cumple con los requisitos generales. • Verificar que no se adjudique el estudiante el número de cupos indicados ya fueron asignados. • Verificar que el método responda correctamente si el usuario no marco cupos por asignar. • Verificar que el método adjudica correctamente al estudiante. • Verificar que no se pueda realizar la adjudicación si aún están abiertas las inscripciones.
findAdjudicados	• Verificar si se han hecho adjudicaciones para una convocatoria particular. • Validar la respuesta correcta de los adjudicados al servicio.

Apéndice B. Tabla pruebas unitarias para el frontend

Tabla 7

Pruebas unitarias al componente de subconvocatorias

FormSubconvocatoriaComponent	
Método	**Lista pruebas unitarias**
buildForm	• Verificar que el formulario se creó a partir del tipo de subconvocatoria seleccionada • Verificar que se crea un subformulario con los datos de las fechas de inscripciones
validateChanges	• Verificar que ante un cambio en el tipo de subconvocatoria se crea el subformulario de fechas nuevamente. • Verificar que para el tipo de subconvocatoria de vulnerabilidad se soliciten fechas de validación • Verificar que las fechas de inicio no sean mayores a las fechas de cierre.

Tabla 8

Pruebas unitarias para el componente de creación de convocatorias

createConvocatoriaComponent	
Método	**Lista de pruebas unitarias**
uploadDependencies	• Verificar que los semestres académicos se estén cargando. • Verificar que los estados de convocatorias se estén cargando.
addSubcall	• Verificar que se cree el formulario primario de subconvocatorias. • Verificar que la cantidad de subconvocatorias agregadas no exceda el total de subconvocatorias que existen.
deleteSubcall	• Verificar que se elimine únicamente la subconvocatoria seleccionada. • Verificar que el formulario interno se limpie. • Verificar que esté disponible el tipo de subconvocatoria

	eliminada para agregarla nuevamente.
validateChanges	• Verificar que al actualizar las fechas de una subconvocatoria se actualice las fechas de vigencia de la convocatoria. • Verificar que al eliminar una subconvocatoria, se actualice o elimine la vigencia de la convocatoria general.
saveConvocatoria	• Verificar que el formulario este completo al momento de guardar la información. • Verificar que los campos no diligenciados se marquen con un error. • Verificar que, al guardar la información de la convocatoria, redireccione al componente de consulta

Tabla 9

Pruebas unitarias para el componente de creación de requisitos

createOrEditRequisitoComponent	
uploadRequisito	• Verificar que el modo de configuración del componente sea edición. • Verificar que en la URL venga el id del requisito a editar • Verificar el cargue de información al requisito que no se provee desde el servicio REST.
uploadDataDidntCome	• Verificar si el requisito es general o particular
uploadDependencies	• Verificar que los objetos se carguen al inicializar el componente. • Verificar que para cada comparador se establezca una imagen predeterminada. • Verificar la salida adecuada dependiendo de los datos recibidos.
setForm	• Verificar que el formulario cuente con todos los campos necesarios • Verificar que los campos requeridos se estén validando en el momento de la creación
validateChanges	• Verificar que si el comparador cambia se actualice la interfaz

	• Verificar que si el parámetro cambia se actualice la interfaz • Verificar que si el valor cambia se actualice la interfaz • Verificar que si se selecciona valor se ajusten los campos requeridos del formulario.
createRequisito	• Verificar que el estado del formulario sea valido • Verificar que al guardar se redireccione a la interfaz de consulta • Verificar que a la falta de campos requeridos se marquen en la interfaz.

Tabla 10

Pruebas unitarias para el componente de consulta de requisitos

ConsultaRequisitosComponent	
Método	**Lista de pruebas unitarias**
ngOnInit	• Verificar que la vista se crea correctamente. • Verificar que se inicializan las tablas de requisitos generales y requisitos particulares. • Verificar que los datos se cargan o se muestra el emptyState.

Tabla 11

Pruebas unitarias para el componente de consultar convocatorias

ConsultarConvocatoriasComponent	
Método	**Lista de pruebas unitarias**
ngOnInit	• Verificar que la vista se crea correctamente. • Verificar que se crea el formulario de consulta correctamente. • Verificar que se cargan los datos necesarios correctamente.
consultar	• Verificar los criterios de consulta para el método. • Verificar el cargue de información para ciertos criterios • Verificar que no lleguen datos con el emptyState.

Tabla 12

Pruebas unitarias para el componente de registro

registroComponent	
Método	**Lista de pruebas unitarias**
ngOnInit	• Verificar el cargue de la información de servicios correctamente. • Verificar el cargue de la información de la convocatoria vigente. • Verificar el cargue de la información de la subconvocatoria vigente.
saveRegistro	• Verificar que exista al menos un servicio solicitado. • Verificar que el registro se guarde correctamente. • Verificar que, al guardar, muestre el resumen del registro del estudiante.

Tabla 13

Pruebas unitarias para el componente de servicios

serviciosComponent	
ngOnInit	• Verificar que la tabla se cargue correctamente. • Verificar que los datos llenen la tabla correctamente. • Verificar que, si no hay datos recuperados, muestre el emptyState.
	• Verificar el despliegue de una ventana modal con el formulario de creación.

addService	
createService	• Verificar que los campos requeridos estén diligenciados • Verificar la creación correcta del nuevo servicio. • Verificar el cierre de la ventana modal.

Tabla 14

Pruebas unitarias para el componente de tipos de subconvocatorias

tiposSubconvocatoriaComponent	
Método	**Lista de pruebas unitarias**
ngOnInit	• Verificar que la tabla se cargue correctamente. • Verificar que los datos llenen la tabla correctamente. • Verificar que, si no hay datos recuperados, muestre el emptyState.
addService	• Verificar el despliegue de una ventana modal con el formulario de creación.
createTipoSubcall	• Verificar que los campos requeridos estén diligenciados • Verificar la creación correcta del nuevo tipo de usuario. • Verificar el cierre de la ventana modal.

Tabla 15

Pruebas unitarias para el componente de tipos de usuarios

consultaTiposUsuariosComponent	
Método	**Lista de pruebas unitarias**
ngOnInit	• Verificar que la información se carga correctamente.
getTiposUsuario	• Verificar que los tipos de usuarios llegan correctamente. • Verificar que se establezcan la prioridad a partir de los datos recuperados. • Verificar que la lista de tipos de usuario sin prioridad se llene a partir de los datos recibidos. • Verificar que los tipos de usuario sin prioridad muestren la sigla de su nombre.
addUserType	• Verificar el despliegue de la ventana modal con el formulario de creación.

Tabla 16

Pruebas unitarias para el servicio de creación de tipos de usuarios

createTipoUsuarioComponent	
Método	**Lista de pruebas unitarias**
ngOnInit	• Verificar el cargue de los datos requeridos para el formulario.

setForm	• Verificar la creación del formulario con los campos requeridos.
saveTipoUsuario	• Verificar que el estado del formulario sea válido para guardar la información. • Verificar los campos no diligenciados que sean obligatorios. • Verificar que al guardar la información se cierre la ventana modal.

Tabla 17

Pruebas unitarias para el servicio de configuracion

configComedoresService	
Método	**Lista de pruebas unitarias**
getPeriodosAcademicos	• Verificar que llegue la información de los periodos académicos existentes en el SIA. • Verificar que si no hay periodos académicos el estado HTTP sea 204.
getEstados	• Verificar que llegue la información de los estados de convocatoria existentes.
getParametros	• Verificar que llegue la información de los parámetros existentes.
getTiposSubconvcatorias	• Verificar que llegue la información de los tipos de subconvocatorias existentes.
getTiposUsuarios	• Verificar que llegue la información de los tipos de usuarios existentes.
getServices	• Verificar que llegue la información de los tipos de servicios disponibles.

Tabla 18

Pruebas unitarias para el servicio de convocatorias

convocatoriasService	
Método	**Lista de pruebas unitarias**
search	• Verificar que los parámetros enviados vayan con el tipo de dato Pageable. • Verificar la información obtenida se ajuste a los parámetros. • Verificar que el código HTTP sea 200, sin importar si la página de respuesta viene sin elementos.
create	• Verificar que se envíen excepciones si ya existe una convocatoria para el periodo académico. • Verificar que se envié una excepción si ya cuenta la convocatoria con más de un mismo tipo de subconvocatoria.

Tabla 19

Pruebas unitarias para el servicio de requisitos

requisitosService	
Método	**Lista de pruebas unitarias**
getRequisitosByGeneral.	• Verificar que la respuesta del método sea un código HTTP 200, siempre que se recuperen datos. • Verificar que la información obtenida sea vigente. • Verificar que la respuesta del método sea un código HTTP 204, siempre que no se recuperen datos. • Verificar que la respuesta este condicionada por el parámetro enviado.
createRequisito	• Verificar que se envié una excepción si existe un requisito con el mismo nombre. • Verificar que el requisito se guarde correctamente. • Verificar que la respuesta del método sea el JSON del requisito creado.
updateRequisito	• Verificar que se envié una excepción si existe un requisito con el mismo nombre. • Verificar que la respuesta del método sea el JSON del requisito creado.
deleteRequisito	• Verificar que se envíe una excepción si el requisito ha sido usado en alguna adjudicación. • Verificar que la respuesta dada por el método sea un TRUE.

Tabla 20

Pruebas unitarias para el servicio de registro

registroService	
Método	**Lista de pruebas unitarias**
getRegistro	• Verificar que para el usuario que consulta sea un estudiante activo. • Verificar la información a partir de si para esa convocatoria el usuario realizo el registro.
createRegistro	• Verificar una excepción si el usuario ya realizo el registro para esa convocatoria. • Verificar una excepción si el usuario no envió detalles de registro. • Verificar una excepción si la fecha de registro es por fuera de las fechas de la convocatoria. • Verificar que la respuesta del método sea el registro con sus respectivos detalles.