Estadísticos Examen de Admisión UdeA (AnalySAGE)

David Nicolás Sánchez Sendoya
Duván Giraldo Avendaño

Informe de trabajo de grado como requisito para optar al título de:
Ingeniero Electrónico.

Asesor Interno
Augusto Enrique Salazar Jimenez
Doctor en Ingeniería

Universidad de Antioquia
Facultad de Ingeniería
Ingeniería Electrónica
Medellín
2023

Cita	(Giraldo Avendaño & Sanchez Sendoya, 2023)
Referencia	Giraldo Avendaño, D., & Sanchez Sendoya, D. N. (2023). *Estadísticos Examen de Admisión UdeA (AnalySAGE)* [Trabajo de grado]. Universidad de Antioquia, Medellín.

Asesor de trabajo de grado: Augusto Enrique Salazar Jimenez

Resumen

Este proyecto de grado se hizo con el objetivo de solucionar el problema que tiene el área de Admisiones y Registro de la Universidad de Antioquia, el cual consiste en que al día de hoy después de que se realiza el examen de admisión esta área hace un análisis de los datos de los aspirantes y de sus resultados en el examen; este análisis es realizado de manera "manual" usando archivos de excel que tienen la información respectiva, lo cual es poco práctico porque sólo se pueden observar cantidades de acuerdo a los filtros usados en los documentos de excel. Como trabajo de grado se implementó una aplicación web realizada en las tecnologías Laravel y Angular, la cual permite almacenar la información y generar una amplia gama de estadísticos gráficos sobre los aspirantes del examen y sobre sus resultados en este.

Introducción

La Universidad de Antioquia es una institución de educación superior pública por lo que para regular las admisiones es necesario implementar una prueba en la que se evalúan las capacidades cognitivas del aspirante. El examen de admisión de la Universidad de Antioquia para los programas de pregrado sea virtuales, presenciales o de regionalización, está dividido en dos partes o componentes principales los cuales son Competencia Lectora y Razonamiento Lógico. Cada uno de los componentes tiene 40 preguntas, para un total de 80 preguntas.

Cada semestre se presentan aproximadamente entre 25000 y 40000 personas, provenientes de varios departamentos del país. Entre los aspirantes hay de estratos 1, 2 y 3 en su mayoría y un pequeño porcentaje pertenece a los estratos 4, 5 y 6. En promedio, cada semestre la universidad tiene la capacidad de recibir alrededor 5000 estudiantes.

Para llevar a cabo el proceso de admisión el departamento de Admisiones y Registro de la universidad tiene una logística bien definida. Una vez el examen de admisión se realiza y se califica, este departamento se toma el trabajo de hacer varios análisis en torno al examen como tal. El análisis se hace de forma estadística para medir por ejemplo las distribuciones de admitidos y no admitidos por departamento, por estrato, por género, por colegio del que obtuvieron el grado de bachiller, entre otros. También se analiza los resultados tanto en el componente de Razonamiento Lógico como en el componente de Competencia Lectora, discriminando por tipo de registro, estrato, género, naturalidad del colegio del que se graduaron, el programa al que se presentaron, entre otros.

El problema es que actualmente el área de Admisiones hace estos análisis de manera "manual" usando filtros de los archivos excel que se generan desde ciertos aplicativos de la universidad que tienen la información de los aspirantes y de sus resultados en el examen. Esta forma de filtrar la información no permite desplegar de forma gráfica y sencilla las

estadísticas que se desean, además representa mucha manualidad y operatividad repetitiva para los encargados de realizar los análisis.

El proceso de desarrollo de la aplicación AnalySAGE, que se implementó como proyecto de grado, mitigó el problema ya que se implementó una aplicación web en la cual el departamento de Admisiones y Registro puede subir la información sobre los aspirantes y sobre sus resultados en el examen de admisión. Una vez la información queda almacenada en la base de datos, en la misma aplicación se creó un módulo que genera gráficas dinámicas sobre el examen de admisión. Se implementaron 3 submódulos de análisis:

1. Estadísticos discriminados por pregunta
2. Estadísticos discriminados por estado de admisión
3. Estadísticos discriminados por componente del examen de admisión

Además de que se crearon los módulos de almacenamiento de información y de generación de gráficos, la aplicación web tiene varios módulos administrativos en los que se pueden crear usuarios y asignarles los permisos que se necesiten, se puede también parametrizar datos de configuración como lo son los colegios, los continentes, países, departamentos, municipios, y programas.

En este documento se consolida todo el trabajo de grado, de tal forma que se describirán los objetivos tanto general como específico, luego se expondrán una serie de conceptos que son esenciales para la comprensión de la terminología que se utiliza en todo el documento. Posteriormente se describirá la metodología implementada para lograr cada objetivo. Finalmente se mostrarán los resultados discriminados por objetivo y se hará un listado de conclusiones relacionadas al desarrollo del proyecto.

Objetivos

General

Desarrollar una aplicación web usando arquitectura orientada a microservicios con plataforma frontend y backend, para la asistencia en el análisis de los resultados del examen de admisión de la Universidad de Antioquia.

Específicos:

1. Realizar levantamiento de requerimientos asociados a la necesidad del departamento de admisiones y registro, teniendo en cuenta directrices de seguridad y buenas prácticas de desarrollo.

2. Diseñar el módulo de almacenamiento y gestión de la información en frontend, backend y bases de datos.

3. Diseñar el módulo de generación de estadísticas y de asistencia al análisis, usando consultas dinámicas a la Base de Datos teniendo en cuenta parametrizaciones de seguridad de base de datos.

4. Implementar el módulo de almacenamiento y gestión de la información usando buenas prácticas de codificación y documentación.

5. Implementar el módulo de generación de estadísticas y de asistencia al análisis, teniendo en cuenta la eficiencia en las consultas al momento de desplegar los datos en los dashboards.

6. Realizar despliegue en ambiente local para verificar el correcto funcionamiento de cada uno de los componentes y detectar posibles errores antes de escalar al área de TI.

Marco Teórico

Desarrollo web: Desarrollo web significa construir y mantener sitios web; es el trabajo que tiene lugar en un segundo plano y que permite que una web tenga una apariencia impecable, un funcionamiento rápido y un buen desempeño para permitir la mejor experiencia de usuario [1].

Backend: El backend es la parte del desarrollo web que se encarga de que toda la lógica de una página web funcione. Se trata del conjunto de acciones que pasan en una web pero que no vemos como, por ejemplo, la comunicación con el servidor [2].

Frontend: El frontend es la parte del desarrollo web que se dedica a la parte frontal de un sitio web, en pocas palabras del diseño de un sitio web, desde la estructura del sitio hasta los estilos como colores, fondos, tamaños hasta llegar a las animaciones y efectos [3].

Angular: es un Framework de JavaScript de código abierto escrito en TypeScript. Su objetivo principal es desarrollar aplicaciones de una sola página. Google se encarga del mantenimiento y constantes actualizaciones de mejoras para este framework [4].

Laravel: Laravel es un framework PHP gratis y de código abierto que brinda un conjunto de herramientas y recursos para crear aplicaciones modernas. Posee un ecosistema integral que combina funciones integradas y una variedad de paquetes y extensiones compatibles.

Este framework de PHP creció en popularidad rápidamente en los últimos años, y muchos desarrolladores lo adoptaron como su framework de trabajo favorito para lograr un proceso de desarrollo optimizado [5].

Chart.js: es una biblioteca JavaScript gratuita de código abierto para la visualización de datos, que admite 8 tipos de gráficos:

1. Barra

2. Línea

3. Área

4. Circular

5. Burbuja

6. Radar

7. Polar

8. Dispersión.

Creado por el desarrollador web con sede en Londres Nick Downie en 2013, ahora es mantenido por la comunidad y es la segunda biblioteca de gráficos JS más popular en GitHub por la cantidad de estrellas después de D3.js, considerada significativamente más fácil de usar aunque menos personalizable que esta. Chart.js se representa en HTML5 Canvas y está ampliamente considerado como una de las mejores bibliotecas de visualización de datos [6].

Base de datos: Una base de datos es una colección de información (datos), generalmente relacionada, que se almacena electrónicamente y en la que se puede buscar. La función de una base de datos no es únicamente la de almacenar datos, sino que también debe hacerlo en un formato que permita una búsqueda eficiente y una recuperación rápida de la información, además de garantizar su seguridad [7].

SPA: Una web SPA (Single Page Application) es una página web la cual está todo el contenido en una sola página, es decir, carga tan solo un archivo HTML y todo se produce dentro de este único archivo. De esta manera se puede ofrecer una experiencia más fluida, más rápida [8].

MVC: Significa modelo (model) vista (view) controlador (controller). Esto es lo que significan cada uno de esos componentes.

Modelo: El backend que contiene toda la lógica de datos.

Vista: El frontend o interfaz gráfica de usuario (GUI)

Controlador: El cerebro de la aplicación que controla como se muestran los datos [9].

APIs: Las Interfaces de Programación de Aplicaciones (APIs por sus siglas en inglés) son construcciones disponibles en los lenguajes de programación que permiten a los desarrolladores crear funcionalidades complejas de una manera simple. Estas abstraen el código más complejo para proveer una sintaxis más fácil de usar en su lugar [10].

Arquitectura monolítica:

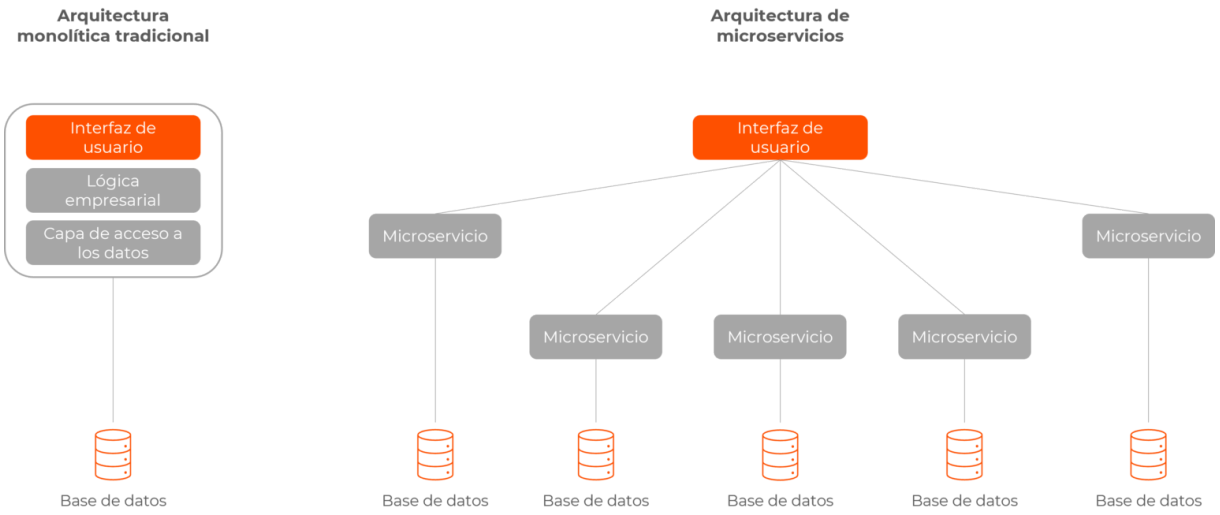
Los primeros programas informáticos utilizaban una arquitectura monolítica, agrupando todo lo relacionado con el sistema dentro del mismo proyecto. Este tipo de arquitectura se caracteriza por:

- Los programas son fáciles de desarrollar.
- El despliegue y la ejecución del software son muy sencillos.
- El costo de desarrollo es bajo en comparación con otras arquitecturas.

Los problemas de este tipo de arquitectura, como la escalabilidad o la dificultad para los desarrolladores (necesitan entender todo el código de la aplicación) han hecho que este tipo de desarrollo de software deje de ser utilizado en muchos proyectos (aunque su sencillez y bajo coste hace que siga siendo interesante para ciertos proyectos con bajos requerimientos) [11].

Arquitectura de microservicios:

En el esquema 1 se puede observar la comparativa de arquitecturas monolíticas y de microservicios.



Esquema 1. Comparativa de arquitectura monolítica y arquitectura de microservicios.

La arquitectura de microservicios es un método de desarrollo de aplicaciones software que funciona como un conjunto de pequeños servicios que se ejecutan de manera independiente y autónoma, proporcionando una funcionalidad de negocio completa. En ella, cada microservicio es un código que puede estar en un lenguaje de programación diferente, y que desempeña una función específica. Los microservicios se comunican entre sí a través de APIs, y cuentan con sistemas de almacenamiento propios, lo que evita la sobrecarga y caída de la aplicación.

Los microservicios han creado infraestructuras IT más adaptables y flexibles. Porque si se quiere modificar solamente un servicio, no es necesario alterar el resto de la infraestructura. Cada uno de los servicios se puede desplegar y modificar sin que ello afecte a otros servicios o aspectos funcionales de la aplicación.

Veamos qué ventajas y desventajas tiene la aplicación de una arquitectura de microservicios frente a otros tipos de arquitectura. Hay que tener en cuenta estos puntos a la hora de identificar qué tipo de arquitectura software será la mejor para un determinado proyecto u organización.

Ventajas:

- **Modularidad:** al tratarse de servicios autónomos, se pueden desarrollar y desplegar de forma independiente. Además un error en un servicio no debería afectar la capacidad de otros servicios para seguir trabajando según lo previsto.
- **Escalabilidad:** como es una aplicación modular, se puede escalar horizontalmente cada parte según sea necesario, aumentando el escalado de los módulos que tengan un procesamiento más intensivo.
- **Versatilidad:** se pueden usar diferentes tecnologías y lenguajes de programación. Lo que permite adaptar cada funcionalidad a la tecnología más adecuada y rentable.
- **Rapidez de actuación:** el reducido tamaño de los microservicios permite un desarrollo menos costoso, así como el uso de "contenedores de software" permite que el despliegue de la aplicación se pueda llevar a cabo rápidamente.
- **Mantenimiento simple y barato:** al poder hacerse mejoras de un solo módulo y no tener que intervenir en toda la estructura, el mantenimiento es más sencillo y barato que en otras arquitecturas.
- **Agilidad:** se pueden utilizar funcionalidades típicas (autenticación, trazabilidad, etc.) que ya han sido desarrolladas por terceros, no hace falta que el desarrollador las cree de nuevo [12].

Metodología

Para la implementación del proyecto se realizaron varios pasos que se pueden enumerar como sigue:

1. Inicialmente se hizo levantamiento de requerimientos asociados a la necesidad del departamento de admisiones y registro, teniendo en cuenta directrices de seguridad y buenas prácticas de desarrollo.

Este levantamiento consistió esencialmente en poner en un listado las necesidades que se querían suplir con el desarrollo de la aplicación. Estas necesidades se redactaron en una terminología funcional, de tal manera que se tenía una mejor visión del producto requerido.

Después de redactar estas necesidades, se hizo un análisis de su viabilidad técnica y se terminaron de consolidar las mismas pero desde un punto de vista más técnico y orientado a las tecnologías con las que se iba a implementar (Frontend: Angular, Backend: Laravel).

2. Se diseñó el módulo de almacenamiento y gestión de la información en frontend, backend y bases de datos.

Este diseño consistió en determinar para el frontend, las fuentes, colores, tamaños de los textos que se iban a utilizar en la aplicación. También se determinaron los colores y la maquetación HTML que se iba a implementar. Todo el diseño del frontend se hizo siguiendo los estándares de colores y estilos que tiene el portal web de la universidad.

Por parte del backend, se determinó la estructura de 3 capas que cada API o servicio iba a tener: controlador, BL y AO. También se determinó que las cargas masivas asociadas a la generación de la información se debía optimizar al máximo ya que la cantidad de datos es representativa (teniendo en cuenta que la cantidad de aspirantes va aproximadamente de 25000 a 40000).

Por parte de base de datos, se identificaron algunos índices a campos de las tablas que intervienen en las cargas masivas (tabla de información personal, tabla de ruta de respuestas, y tabla de respuestas de los aspirantes).

3. Se diseñó el módulo de generación de estadísticas y de asistencia al análisis, usando consultas dinámicas a la Base de Datos teniendo en cuenta parametrizaciones de seguridad de base de datos.

Como en el diseño del módulo de almacenamiento de información se determinó para el frontend, las fuentes, colores, tamaños de los textos que se iban a utilizar en la aplicación. También se determinaron los colores y la maquetación HTML que se iba a implementar. Además se consultó la mejor opción en cuanto a librería para generar las gráficas estadísticas y se eligió Chart Js, porque es open source y porque ofrece una amplia gama de posibilidades a nivel estadístico.

Para el backend de igual manera se determinó el uso de 3 capas para las APIs y se establecieron las buenas prácticas para evitar el Cross Site Scripting, el Sql Injection y la duplicidad de código.

4. Se implementó el módulo de almacenamiento y gestión de la información usando buenas prácticas de codificación y documentación. Este módulo consiste en 3 cargas de configuración correspondiente a programas, colegios y municipios, y de 3 cargas de archivos asociados a la presentación del examen.

Este paso fue uno de los más importantes y de los que tomó más tiempo en realizarse ya que la metodología implementada en el frontend fue ir creando componentes que tenían 3

archivos principales: el de la maquetación HTML, el de la lógica, y el de los estilos. El objetivo de hacer componentes es poder reutilizar código y así evitar la duplicidad innecesaria del mismo. A medida que se iba desarrollando cada componente se iba realizando el escaneo de código con el software Sonar, el cual informaba si había riesgos de seguridad, malas prácticas como variables sin usar o funciones con alta complejidad cognitiva, entre otras cosas. Cada que se ejecutaba el escaneo con Sonar y se encontraban posibilidades de mejora se intervenía el código en el lugar que indicaba el resultado del escaneo.

Por su lado en el backend la metodología para la creación de cada uno de los servicios consistió en inicialmente crear una ruta para la API, luego crear una función en el controlador respectivo para capturar la información que el frontend le enviaba y retornar el resultado del procesamiento que se realizaba en el servidor. La función del controlador invoca la función del BL en el que se programa la lógica de negocio (por sus siglas en inglés Business logic). La función del BL hace invocación a las funciones del AO que tienen la responsabilidad única de conectarse a la base de datos para hacer consultas o para hacer transacciones.

De igual manera a la par que se iba desarrollando el backend se iba haciendo escaneo con la herramienta Sonar y se iban haciendo correcciones si fuera necesario.

5. Se implementó el módulo de generación de estadísticas y de asistencia al análisis, teniendo en cuenta la eficiencia en las consultas al momento de desplegar los datos en los dashboards. Para el frontend se implementó la librería open source llamada Chart Js, la cual proporciona una variedad amplia de posibilidades en lo que respecta a la renderización de información de manera gráfica.

La metodología de desarrollo en el paso 5 fue la misma implementada en el paso 4, de tal manera que se implementaron varios componentes en el frontend, cada uno con su archivo HTML, su archivo de lógica y su archivo de estilos. Por su parte en el backend a las APIs se les construyó su ruta, su controlador, su BL y su AO.

6. Se realizó despliegue en ambiente local para verificar el correcto funcionamiento de cada uno de los componentes y detectar posibles errores antes de escalar al área de TI.

Una vez se culminó el desarrollo en lo que respecta a la codificación, se hicieron pruebas funcionales corriendo el backend y el frontend en local, lo que significa que el computador personal en el que se hizo el desarrollo era que el que estaba sirviendo como cliente y como servidor al mismo tiempo.

Estas pruebas funcionales dieron como resultado algunos errores mínimos los cuales se corrigieron.

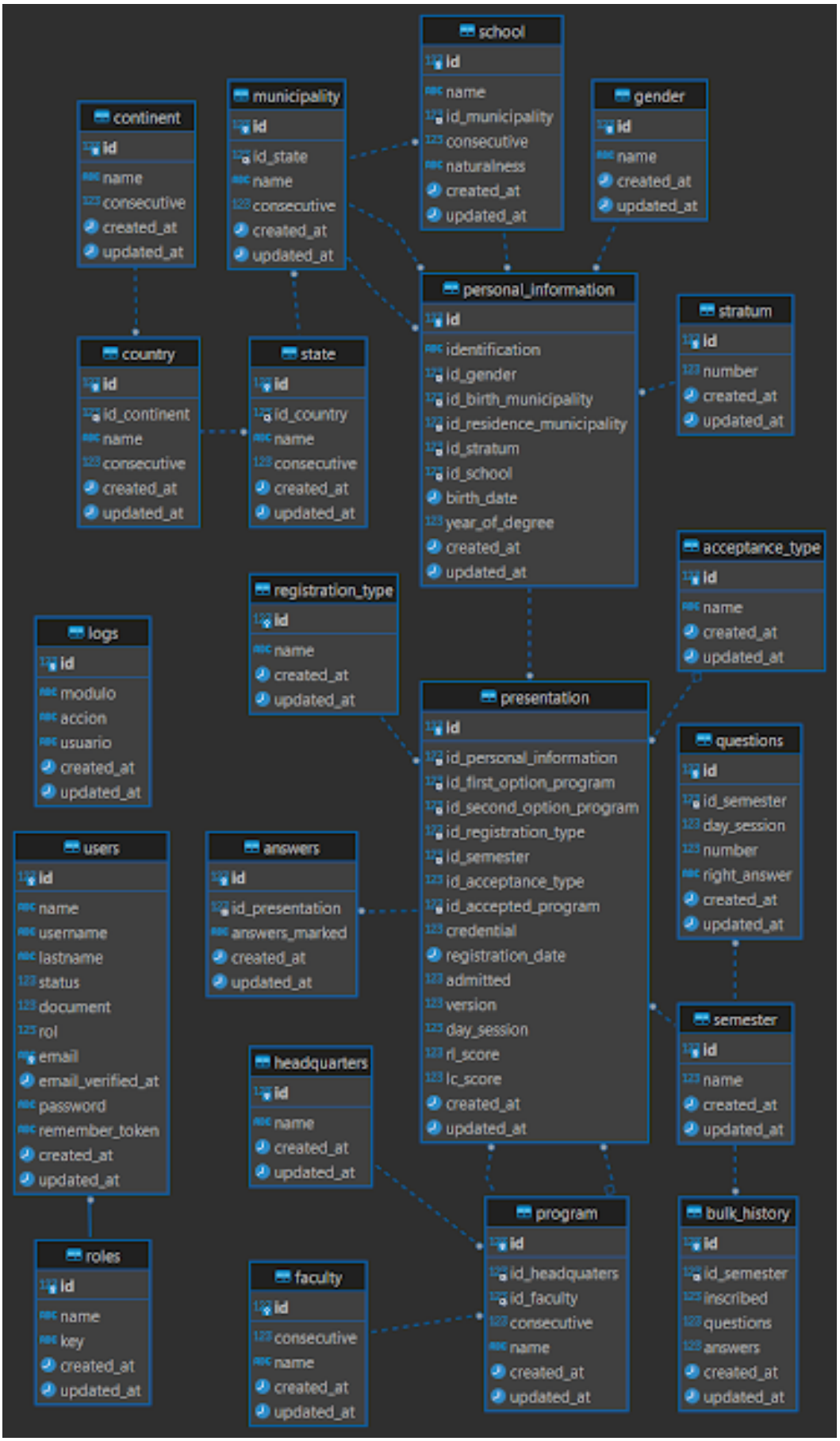
Finalmente se terminó de subir todo el código tanto del backend como del frontend a los repositorios de la universidad de tal forma que el código queda almacenado y protegido en la nube en la plataforma de Gitlab.

Resultados y análisis

Para cumplir el ***objetivo específico 1***, se redactó el listado de requisitos funcionales:

1. Se necesita un software que permita asistir en el análisis del examen de admisión de la Universidad de Antioquia.
2. El software requerido debe ser de tipo sitio web para que pueda ser accedido por internet.
3. Se necesita que el software tenga la capacidad de almacenar la información referente a los aspirantes del examen y respecto a sus resultados en el mismo.
4. Se necesita que el software tenga la capacidad de generar diversos estadísticos sobre la información almacenada. Estos estadísticos deben ser en lo posible muy visuales, por ejemplo gráficas de barra y de torta con colores distintivos y cifras representativas.
5. Se necesita que el software permita crear varios usuarios y que se le puedan asignar un rol.
6. Se requiere que el software tenga dos roles, el rol administrativo que puede acceder a todos los módulos dentro de la aplicación, y el rol de tipo Usuario genérico que sólo tenga acceso a las estadísticas generadas a partir de la información almacenada (esto con el objetivo de restringir las personas que tienen acceso a la información del resultado del examen de admisión).
7. Se requiere que el software tenga principios de seguridad como el uso de sesión, de tal manera que cada usuario tenga un nombre de usuario y una contraseña única.

Como parte del proceso de diseño (***objetivos específicos 2 y 3***) se planteó el modelo de base de datos (ver gráfica 1).



Gráfica 1. Modelo Entidad Relación de la base de datos implementada.

Como se puede observar en la gráfica 1, la base de datos es relacional por lo tanto algunas de las tablas tienen llaves foráneas que permiten generar restricciones respecto a las relaciones establecidas y que se representan por las líneas punteadas. Todas las tablas tienen el estándar de nombramiento separado por guión bajo y escrito en inglés, de la misma manera están nombrados cada uno de los campos de las tablas. Cada una de las tablas tienen los campos created_at y updated_at los cuales almacenan la fecha de creación de los registros y su fecha de actualización, respectivamente.

Las tablas principales son presentation y personal_information, puesto que en estas se almacena la información relacionada a la presentación del examen de todos los aspirantes y la información personal de los mismos, respectivamente. A su vez la tabla presentation se relaciona con las tablas registration_type, answers, program, semester, acceptance_type y personal_information las cuales almacenan respectivamente: el tipo de registro del aspirante, las respuestas que tuvo el aspirante en el examen de admisión, los programas a los que aplicó tanto de primera como de segunda opción, el semestre en el que presentó el examen, el tipo de aceptación en caso tal de haber sido aceptado y la información personal de cada uno de los aspirantes.

Por su parte la tabla personal_information se relaciona con las tablas gender, stratum, school, municipality, las cuales almacenan a su vez respectivamente: el género, el estrato socioeconómico, el colegio, y el municipio tanto de nacimiento como de residencia del aspirante.

Las tablas continent, state y country almacenan el continente, el departamento y el país, ya que en el archivo de carga de la información personal están estos junto con el municipio para indicar el lugar de residencia o el lugar de nacimiento.

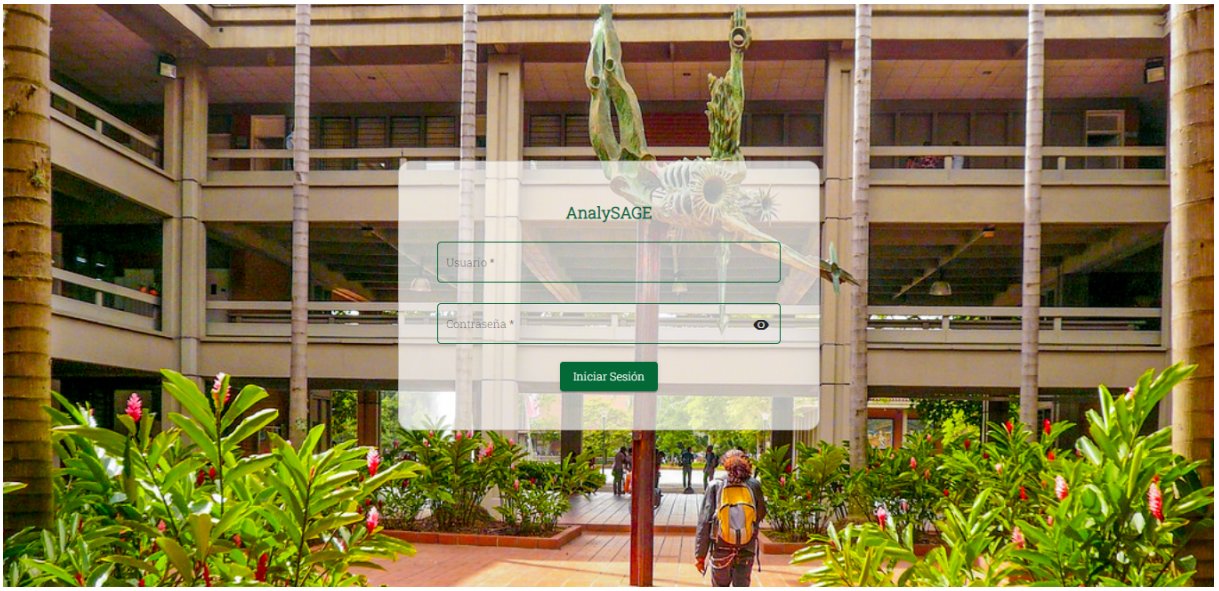
En la tabla logs se almacenan algunas de las acciones que se pueden realizar dentro de la aplicación, de tal manera que se almacena el módulo, la acción y el usuario que la realizó. La tabla de users y roles almacenan respectivamente los usuarios y los roles de la aplicación. En la tabla questions se almacenan las respuestas correctas por jornada para un semestre, basados en la versión 1 del examen. En la tabla bulk_history se almacena el historial de las cargas masivas que se realizan dentro de la aplicación, esto con el objetivo de realizar validaciones como que no se pueda subir el archivo de respuestas de los aspirantes para un semestre determinado sin antes haber subido el archivo de inscritos para ese semestre.

Las tablas headquarters y faculty almacenan respectivamente las sedes y las facultades de cada uno de los programas de pregrado que se almacenan a través de las cargas de configuración.

Para cumplir con el ***objetivo específico 6*** se corrió el proyecto en local, y se capturaron la serie de evidencias que se encuentran a continuación:

Para cumplir con el ***objetivo específico 5*** se realizó lo siguiente:

Inicialmente la aplicación muestra el formulario de inicio de sesión (ver gráfica 2), el cual pide usuario y contraseña que deben estar registrados previamente.

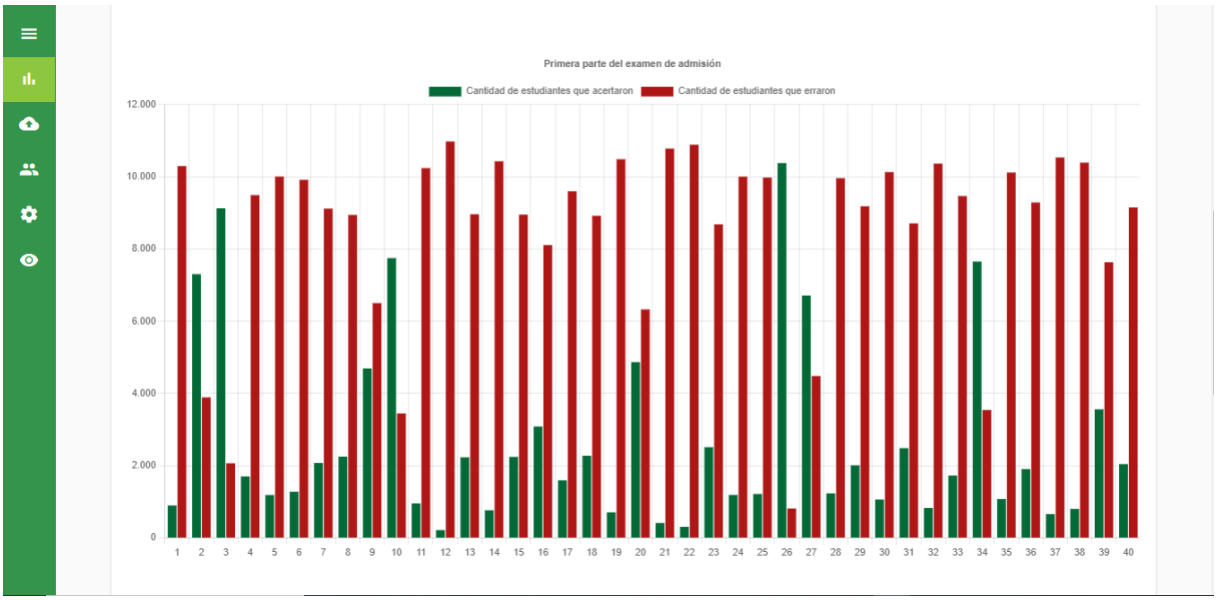


Gráfica 2. Inicio de sesión de la aplicación.

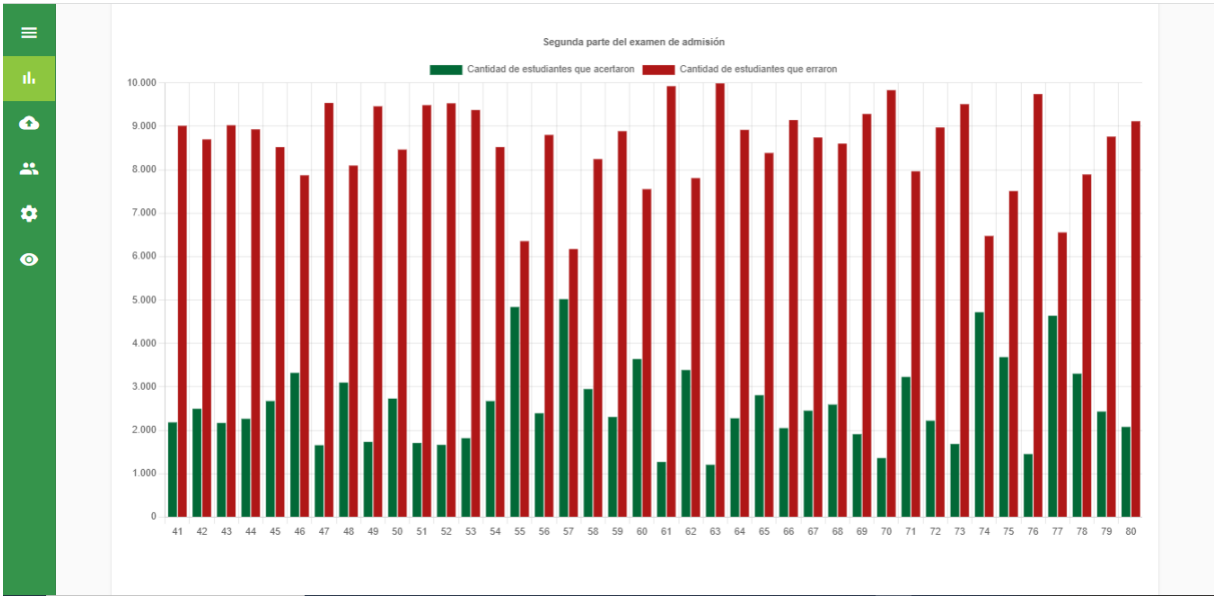
Una vez el usuario inicia sesión, la aplicación lo redirige al módulo de estadísticas dinámicas en los que se pueden apreciar los filtros de la gráfica 3.

Gráfica 3. Vista inicial de la aplicación una vez se inicia sesión.

En el módulo de estadísticas dinámicas hay 3 submódulos. El primero corresponde a las estadísticas discriminadas por pregunta, en el cual se presentan dos gráficas, una asociada al resultado por pregunta de la primera parte del examen y otra el resultado por pregunta de la segunda parte (ver gráficas 4 y 5).

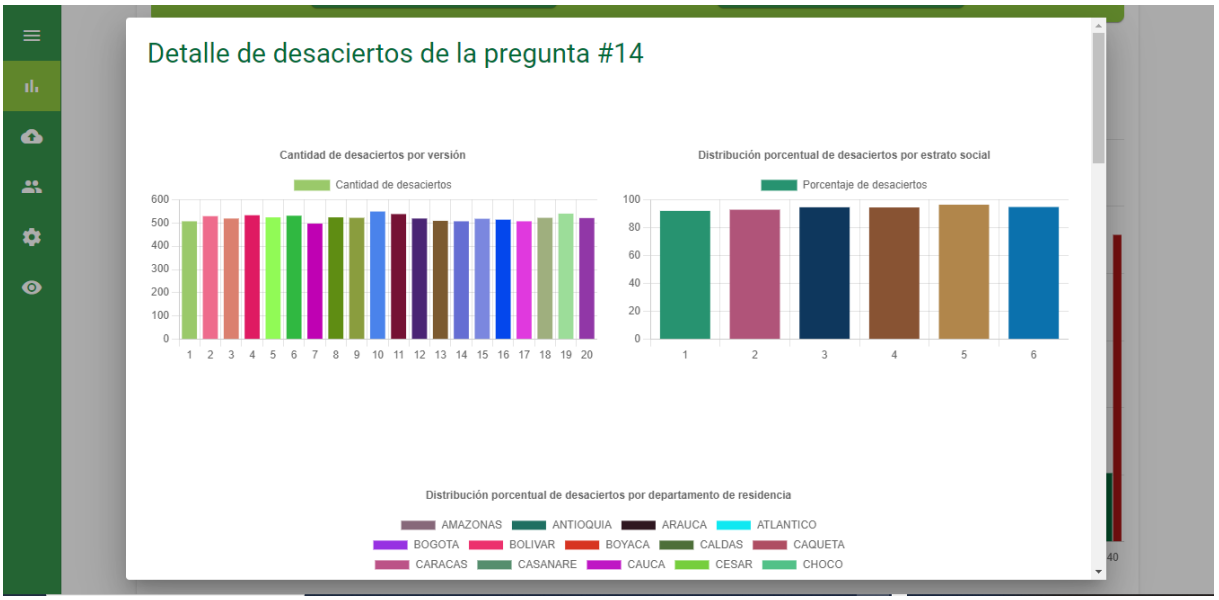


Gráfica 4. Estadísticas por pregunta de la primera parte del examen.

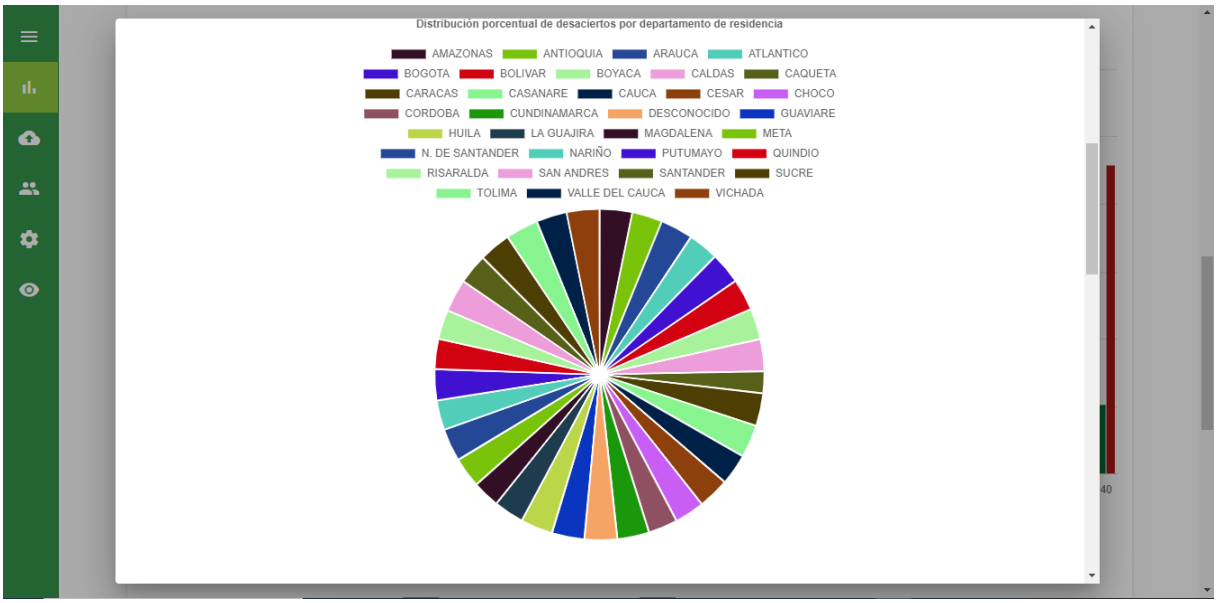


Gráfica 5. Estadísticas por pregunta de la segunda parte del examen.

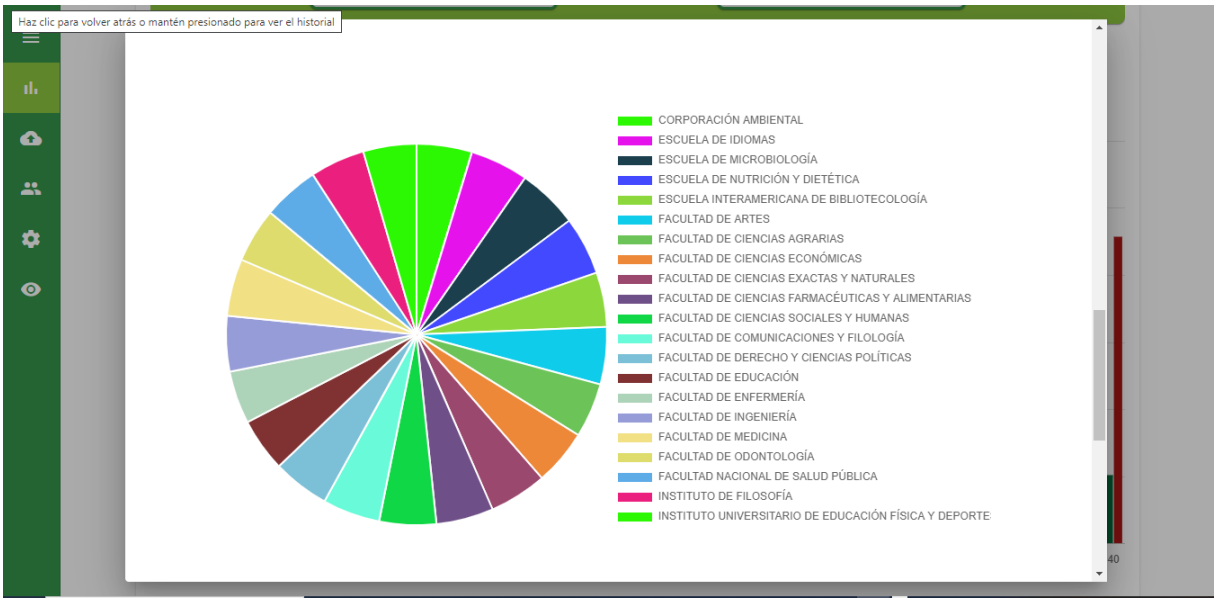
Al dar click en el cualquier barra de las gráficas 4 o 5, se abre una ventana en la que se muestran los detalles de la pregunta. En las figuras 6, 7, 8 y 9 se muestran las gráficas que se generan al dar click en la barra de desaciertos para la pregunta 14.



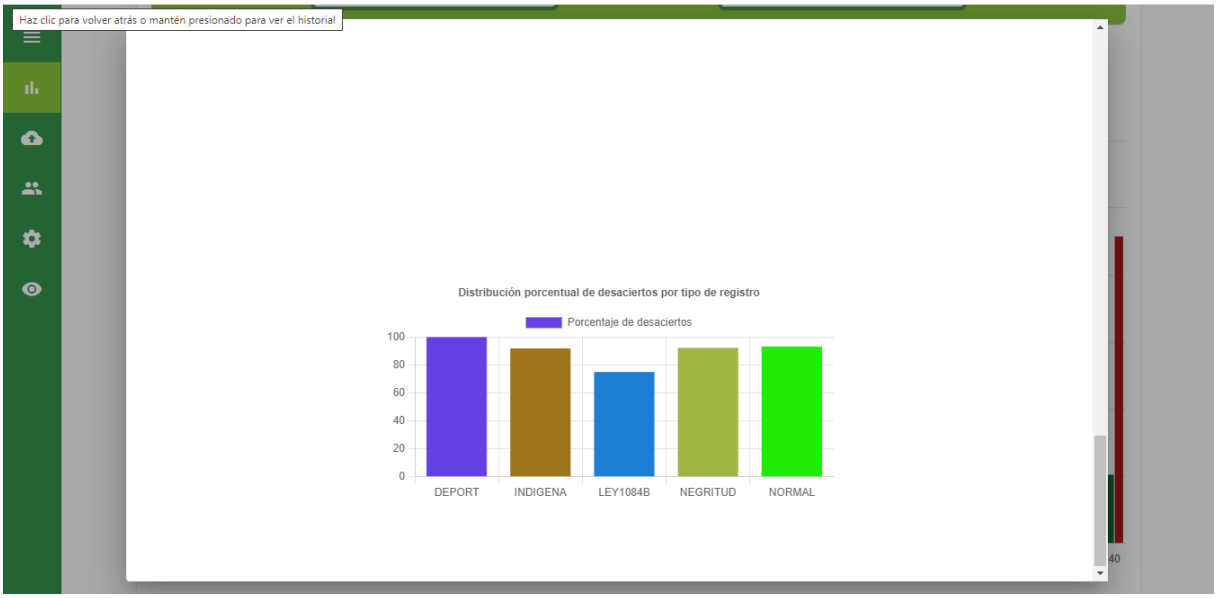
Gráfica 6. Gráficas de cantidad de desaciertos por versión y distribución porcentual de desaciertos por estrato social.



Gráfica 7. Distribución porcentual de desaciertos por departamento de residencia.

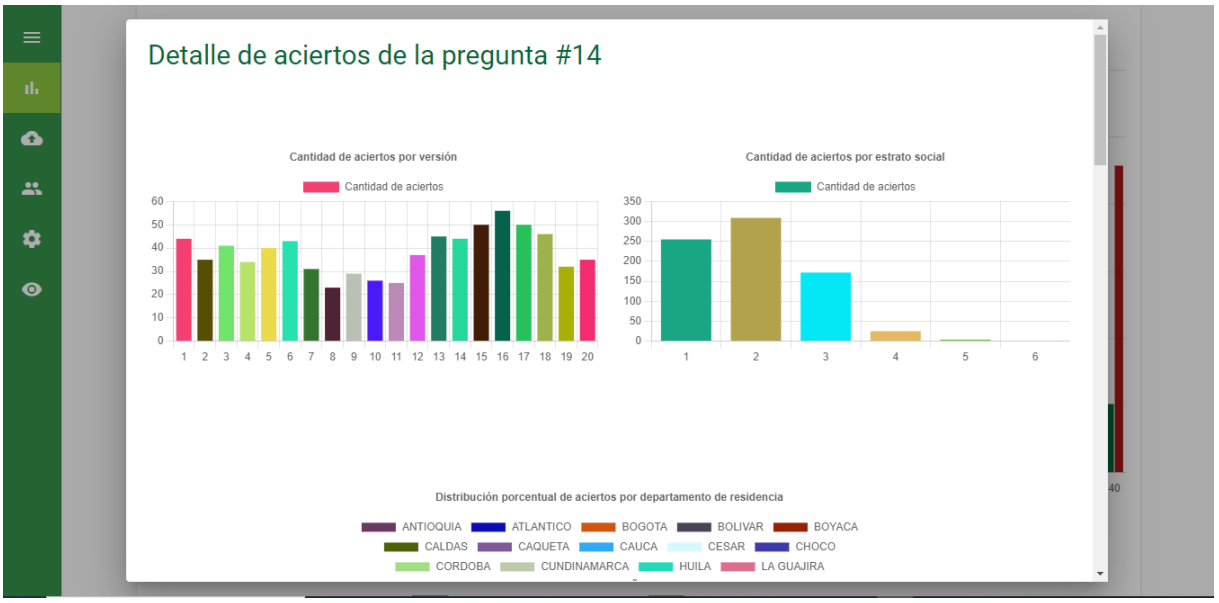


Gráfica 8. Distribución porcentual de desaciertos por facultad de primera opción.

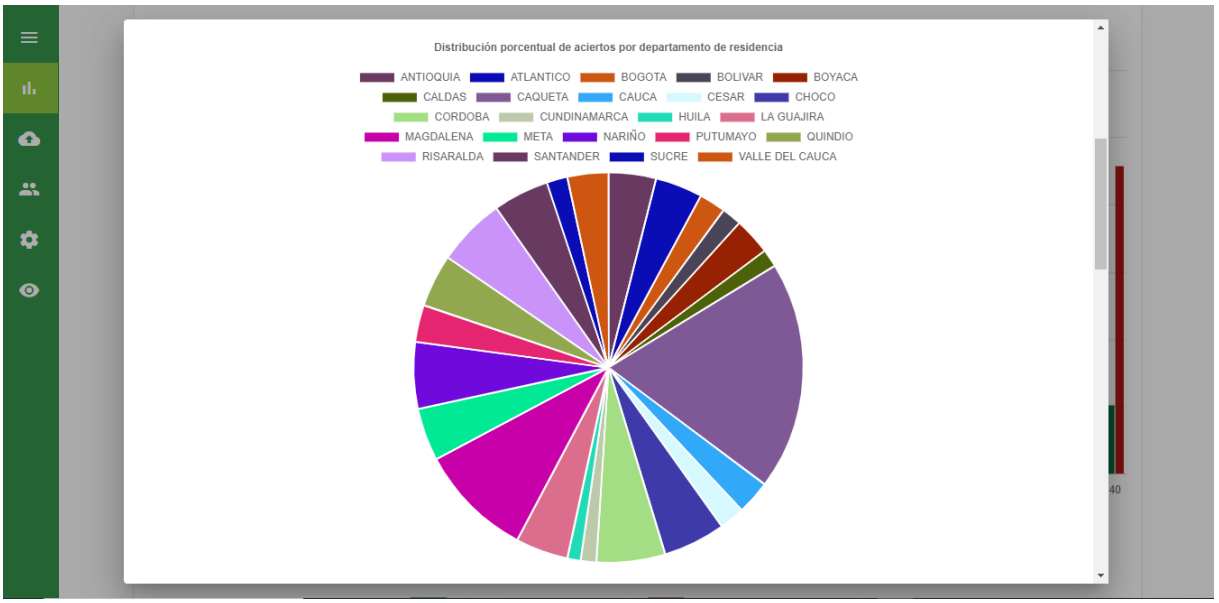


Gráfica 9. Distribución porcentual de desaciertos por tipo de registro.

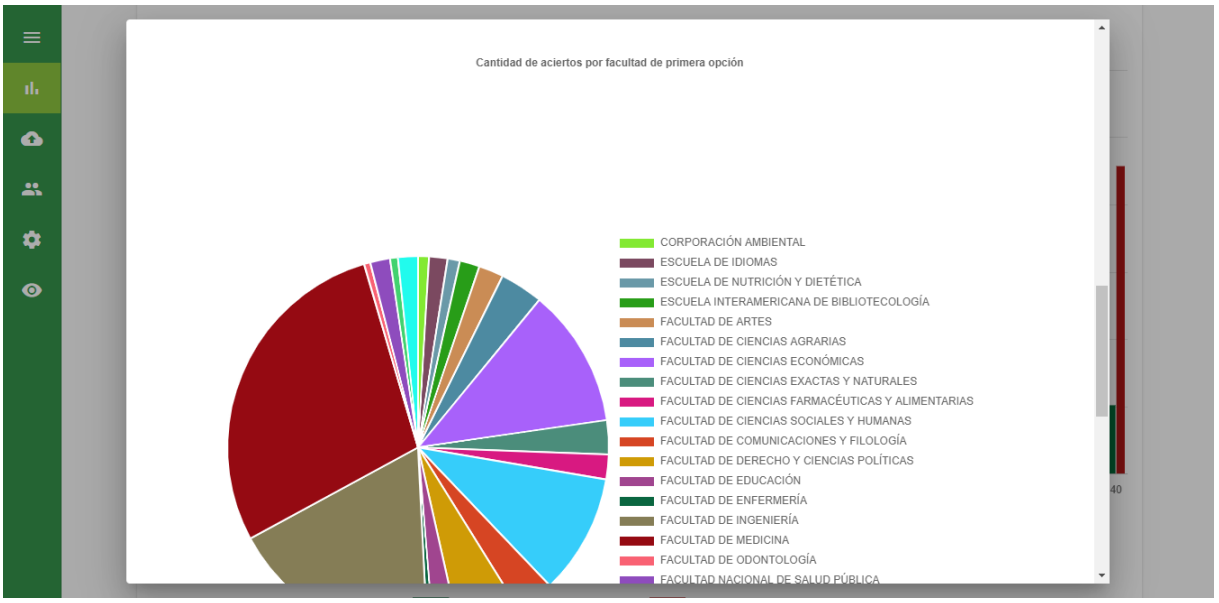
En las gráficas 10, 11, 12 y 13 se muestran las gráficas que se generan al dar click en la barra de aciertos para la pregunta 14.



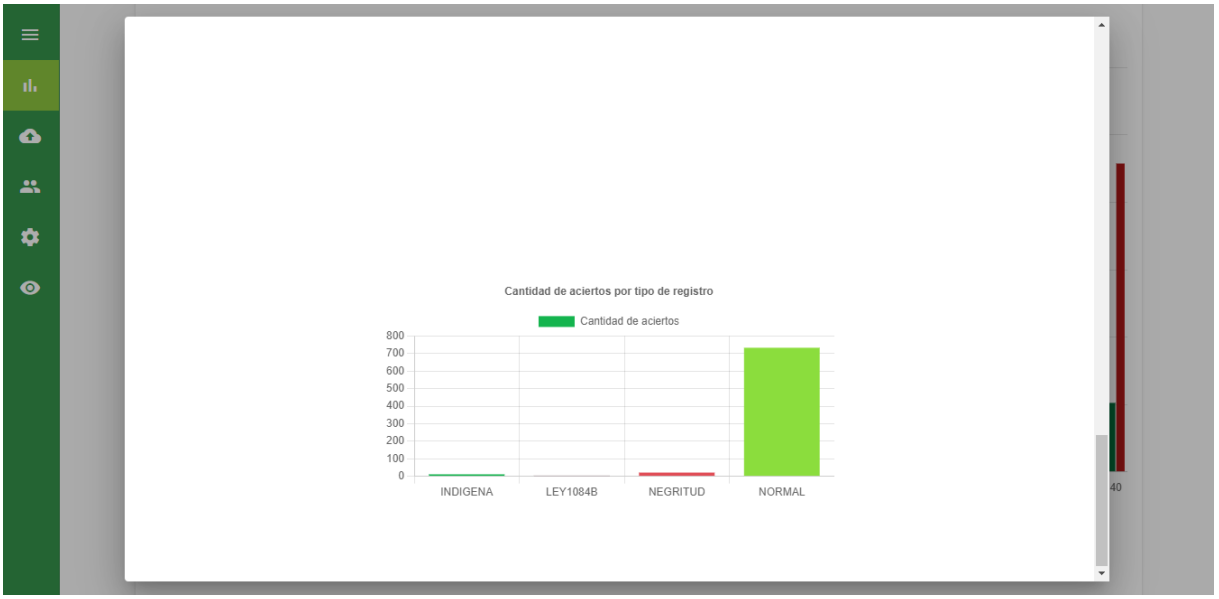
Gráfica 10. Gráficas de cantidad de aciertos por versión y por estrato social.



Gráfica 11. Distribución porcentual de aciertos por departamento de residencia

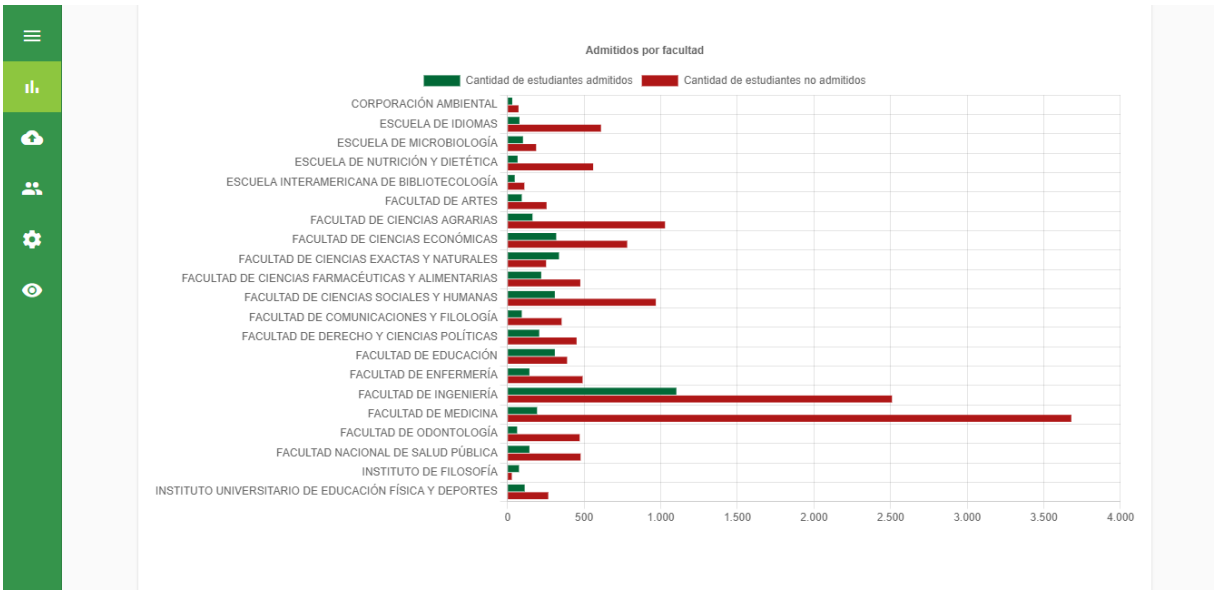


Gráfica 12. Cantidad de aciertos por facultad de primera opción.



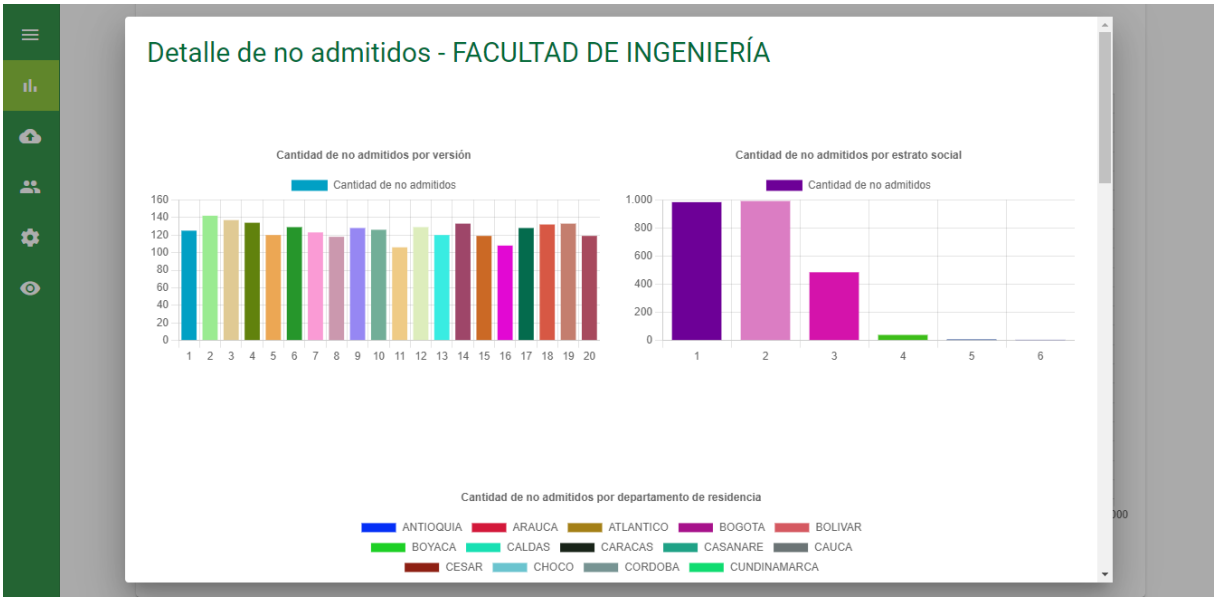
Gráfica 13. Cantidad de aciertos por tipo de registro.

El segundo submódulo corresponde a las estadísticas por estado de admisión, en el que se pueden validar la cantidad de admitidos y no admitidos por facultad (ver gráfica 14).

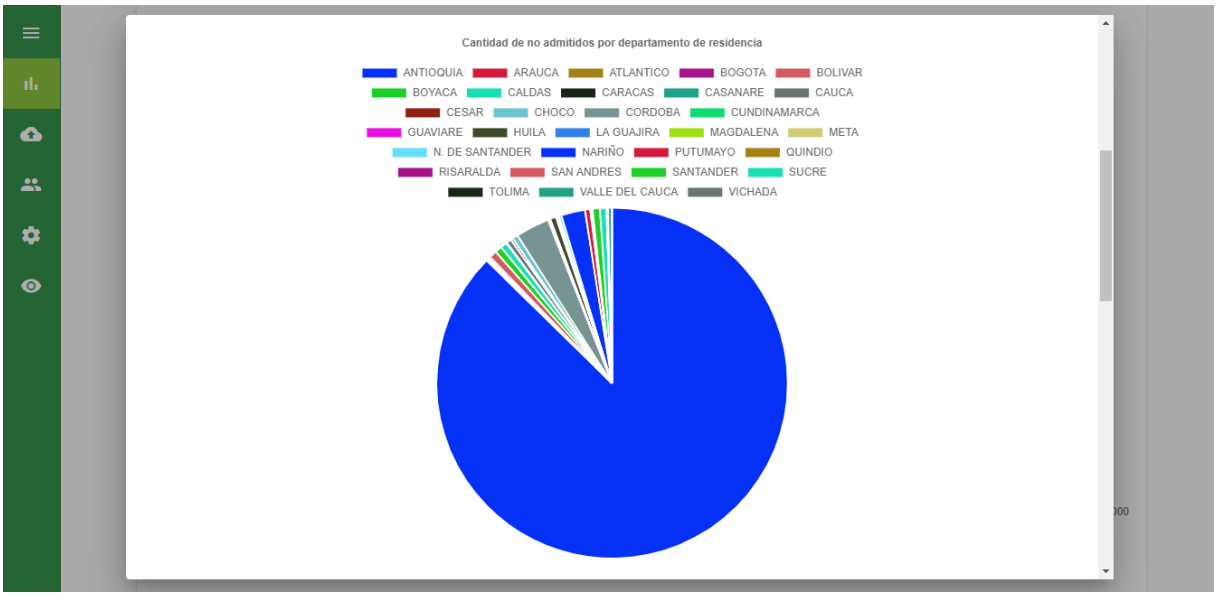


Gráfica 14. Cantidad de admitidos y no admitidos por facultad.

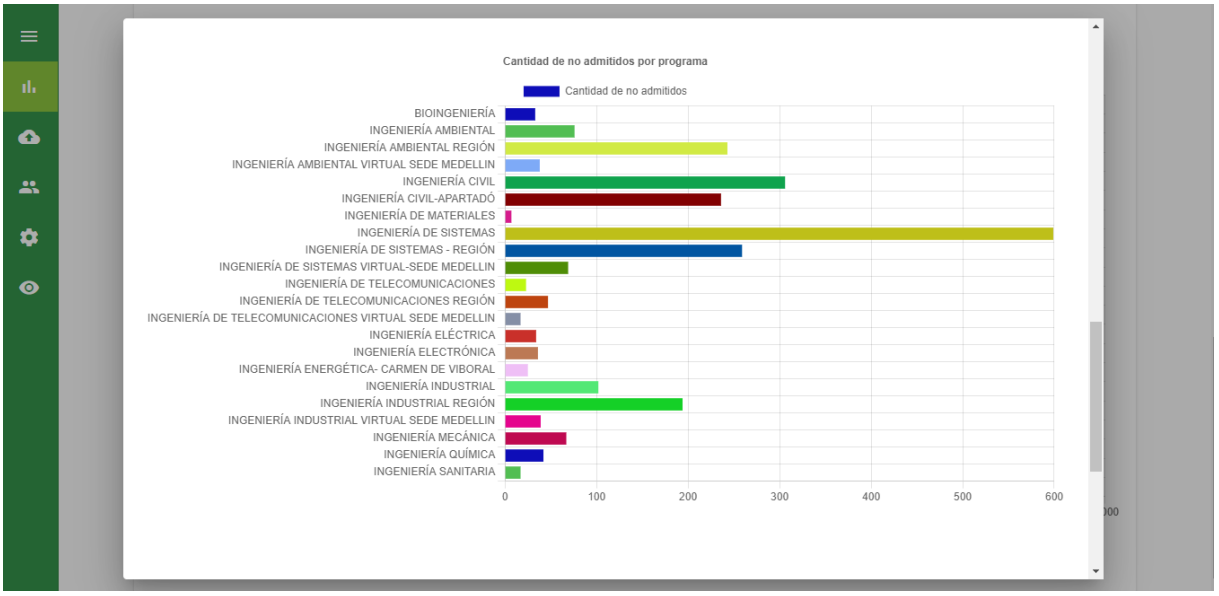
Al dar click en el cualquier barra de las gráfica 14, se abre una ventana en la que se muestran los detalles. En las gráficas 15, 16, 17 y 18 se muestra la ventana que emerge al dar click en la barra de no admitidos a la facultad de ingeniería.



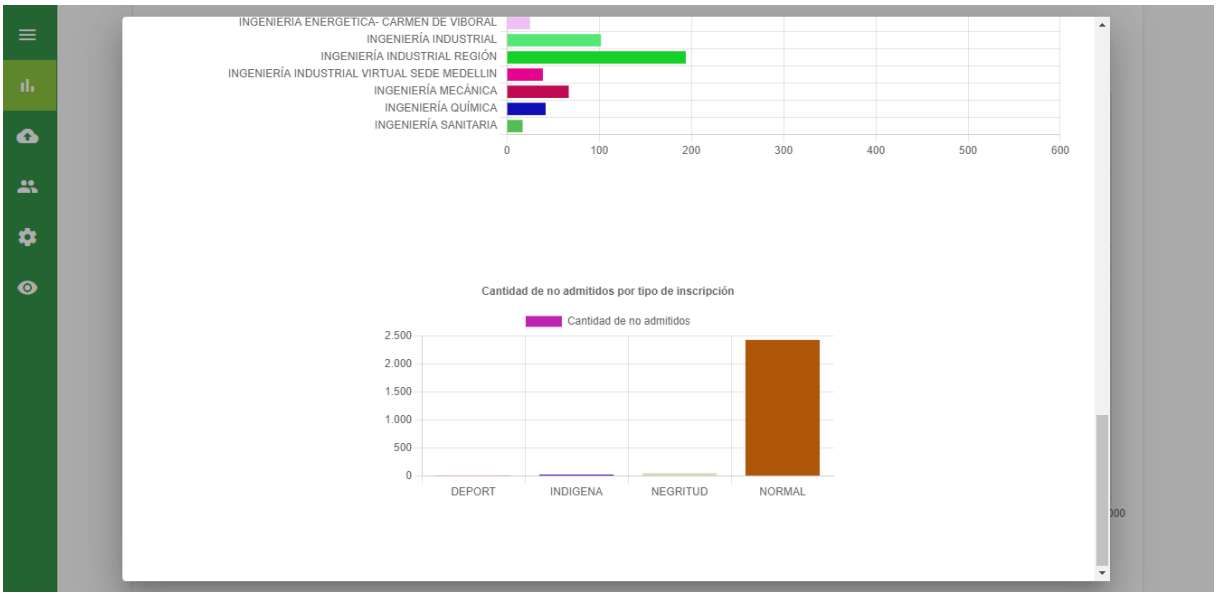
Gráfica 15. Detalle de cantidad de no admitidos por versión y por estrato social.



Gráfica 16. Cantidad de no admitidos por departamento de residencia.

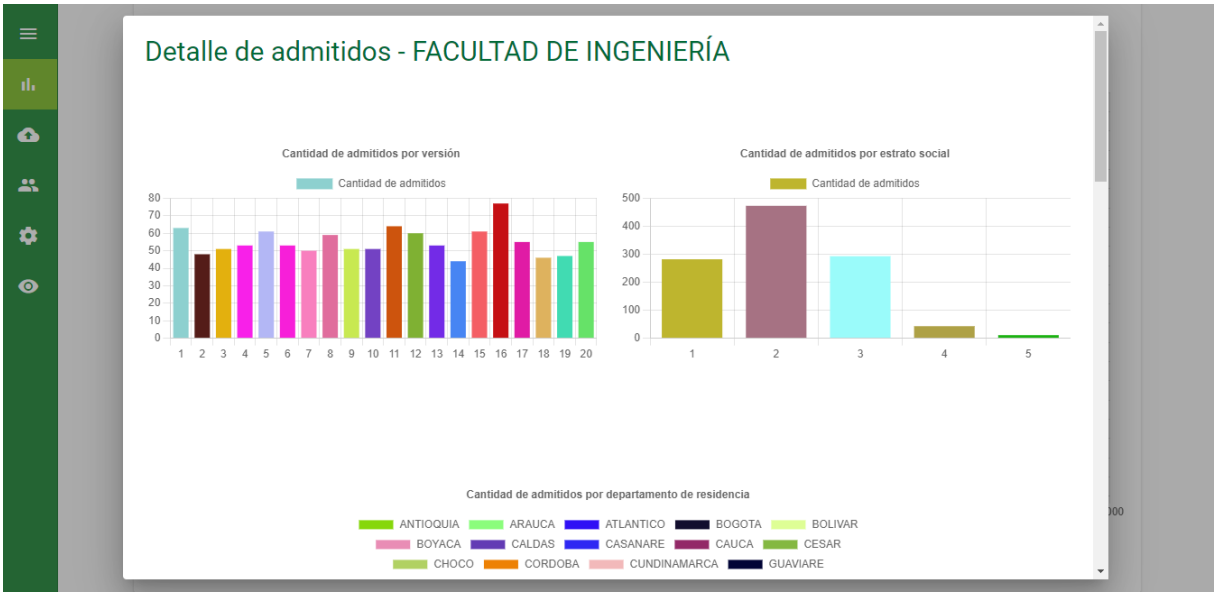


Gráfica 17. Cantidad de no admitidos por programa.

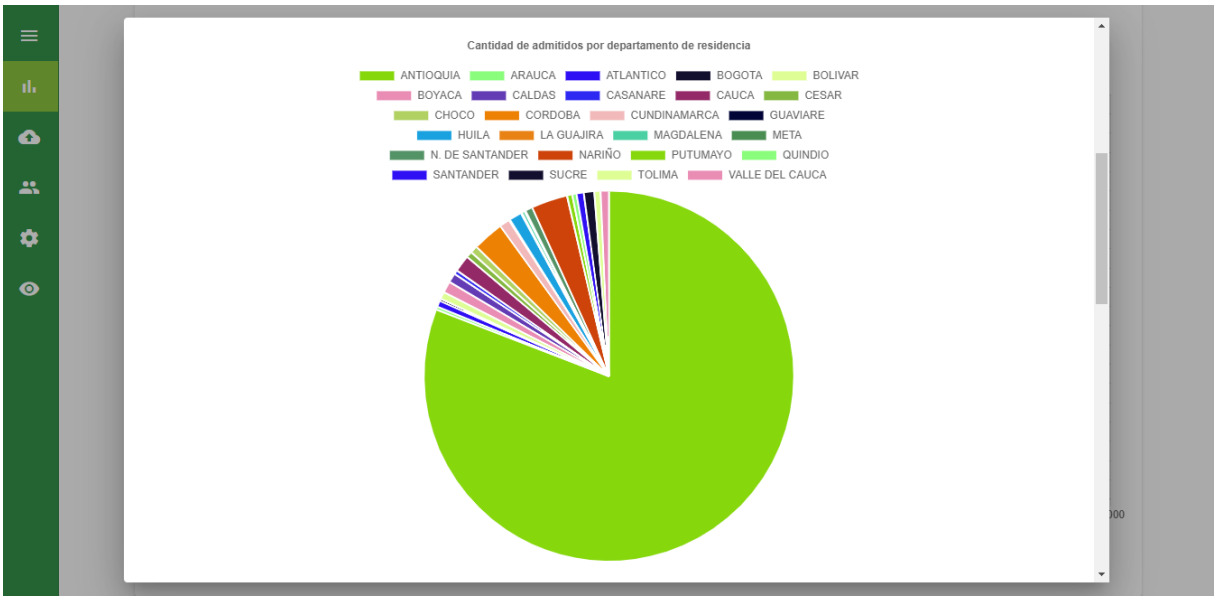


Gráfica 18. Cantidad de no admitidos tipo de inscripción.

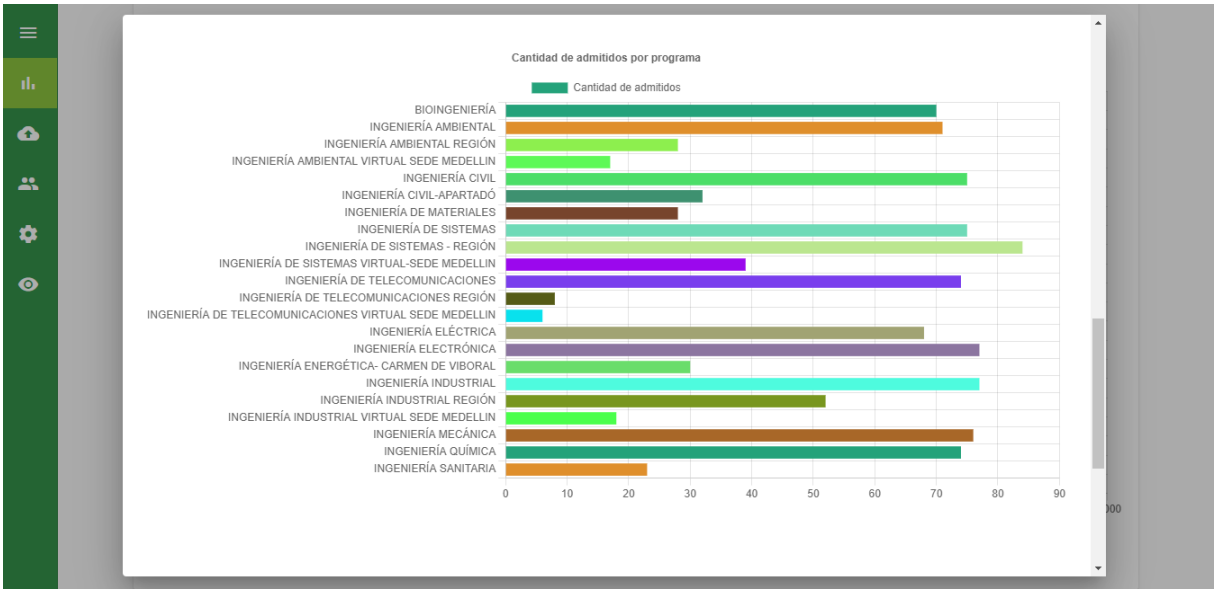
En las gráficas 19, 20, 21 y 22 se muestra la ventana que emerge al dar click en la barra de admitidos a la facultad de ingeniería.



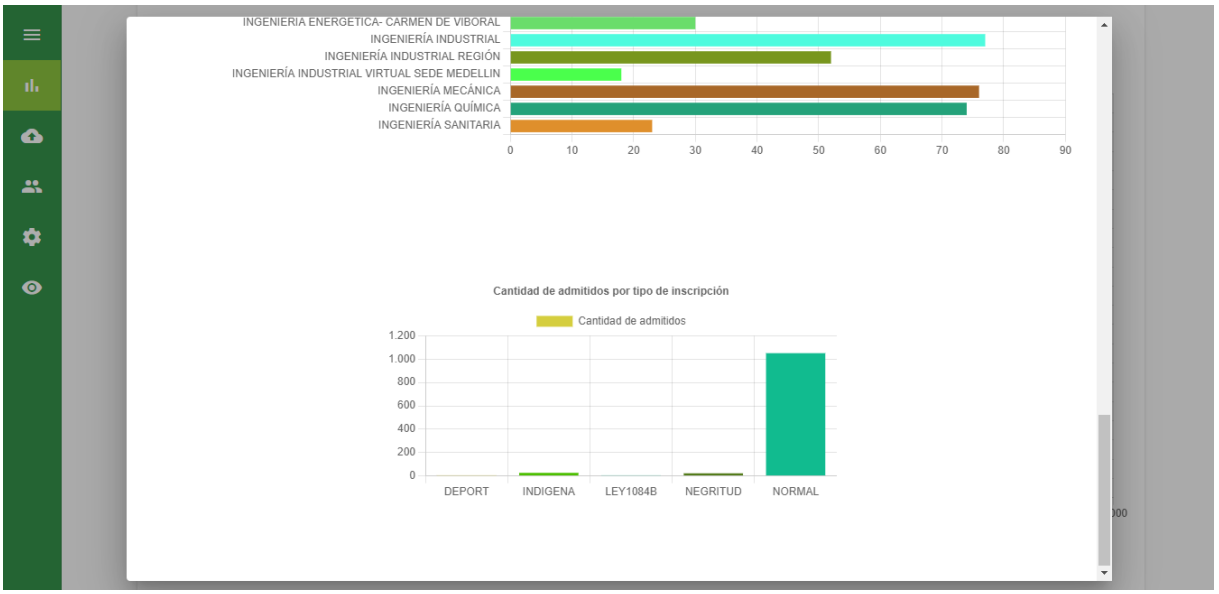
Gráfica 19. Cantidad de admitidos por versión y por estrato social.



Gráfica 20. Cantidad de admitidos por departamento de residencia.

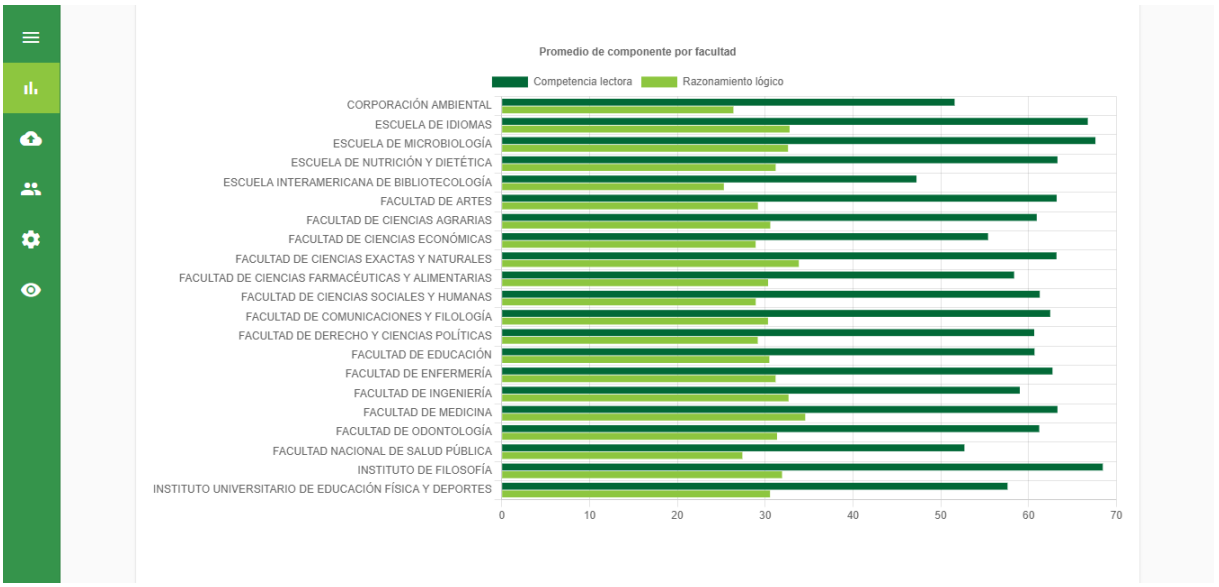


Gráfica 21. Cantidad de admitidos por programa.



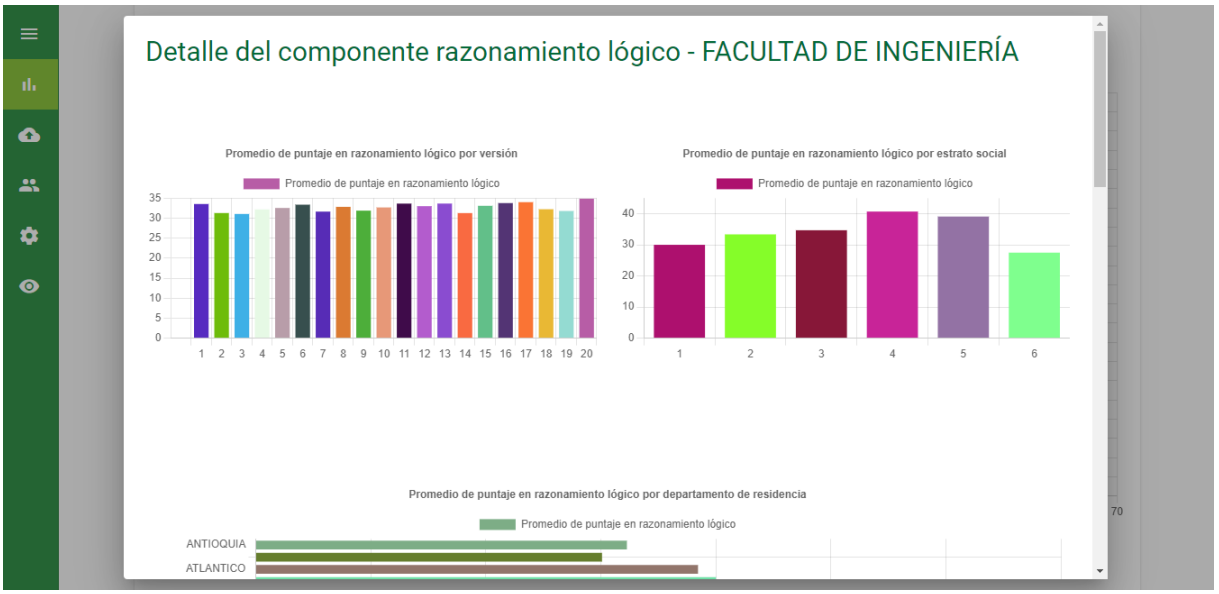
Gráfica 22. Cantidad de admitidos por tipo de inscripción.

El tercer submódulo corresponde a las estadísticas por componente del examen de admisión, en el que se pueden validar los resultados promedio en Razonamiento Lógico y en Competencia Lectora por facultad (ver gráfica 23).

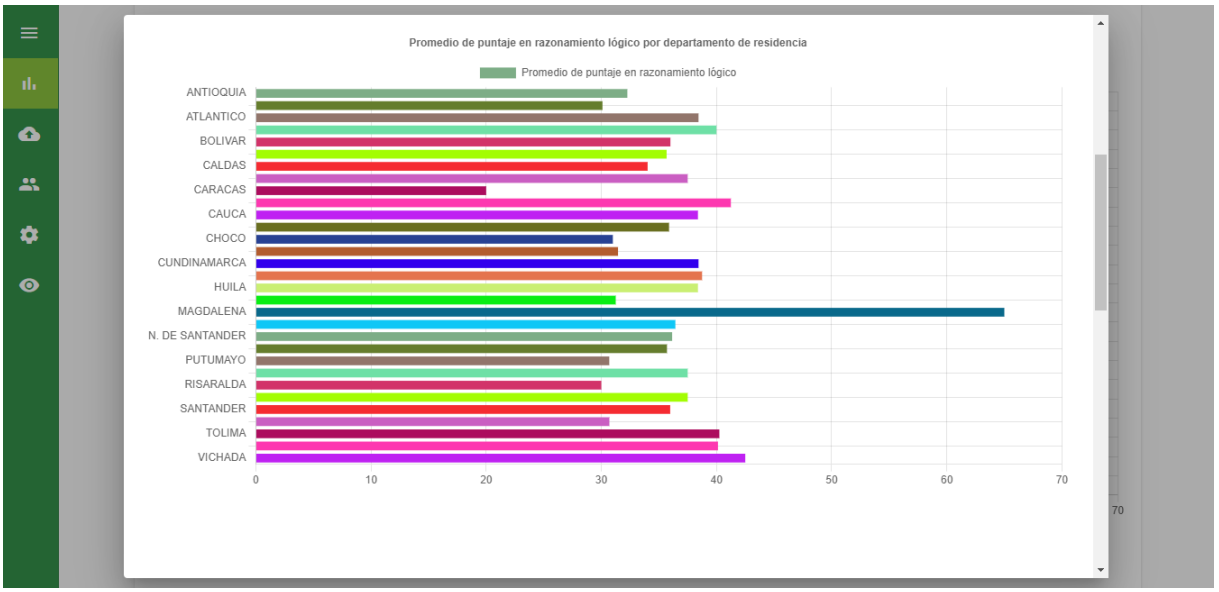


Gráfica 23. Promedio en Razonamiento Lógico y en Competencia Lectora por facultad.

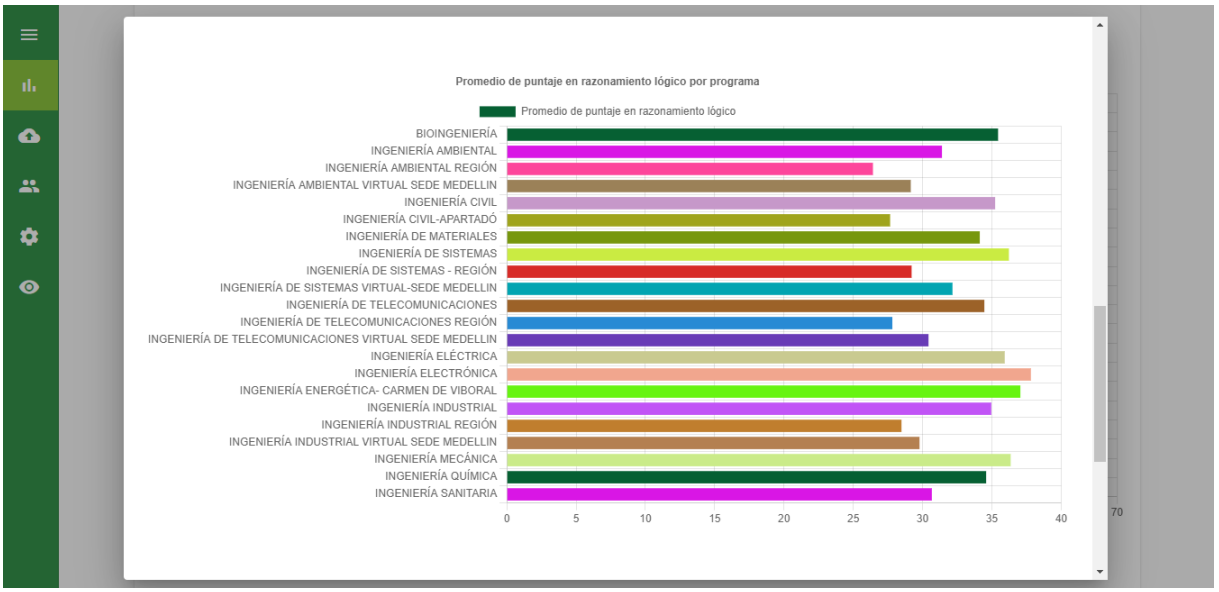
Al dar click en el cualquier barra de la gráfica 23, se abre una ventana en la que se muestran los detalles. En las gráficas 24, 25, 26 y 27 se muestra la ventana que emerge al dar click en la barra de Razonamiento Lógico para la facultad de ingeniería.



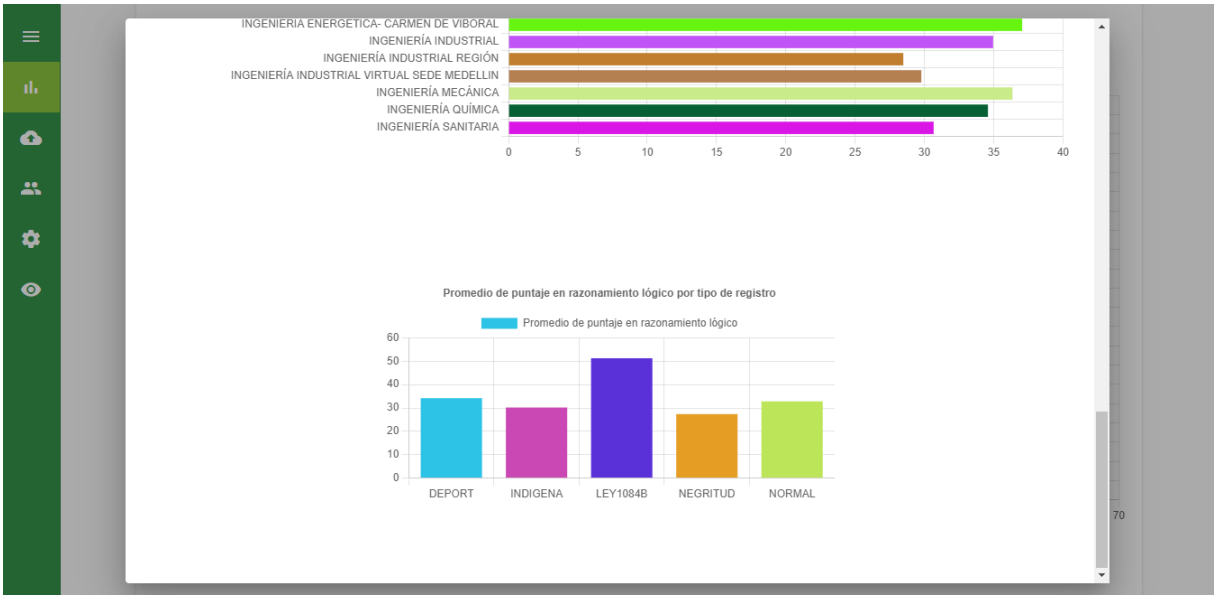
Gráfica 24. Promedio en Razonamiento Lógico por versión y por estrato social.



Gráfica 25. Promedio en Razonamiento Lógico por departamento de residencia.

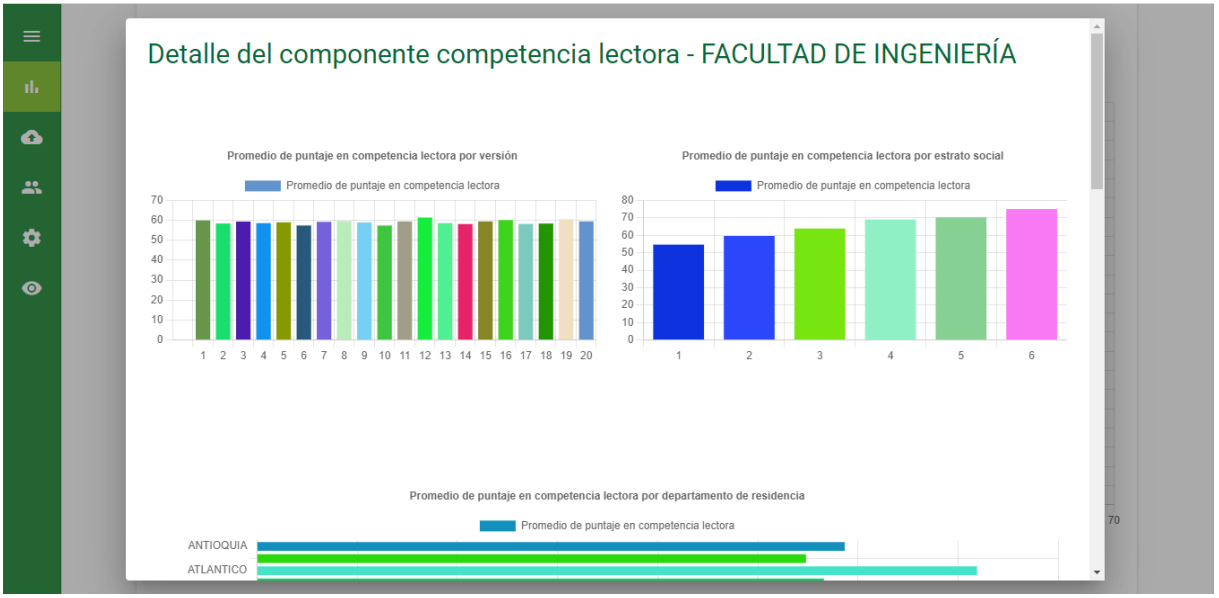


Gráfica 26. Promedio en Razonamiento Lógico por programa.

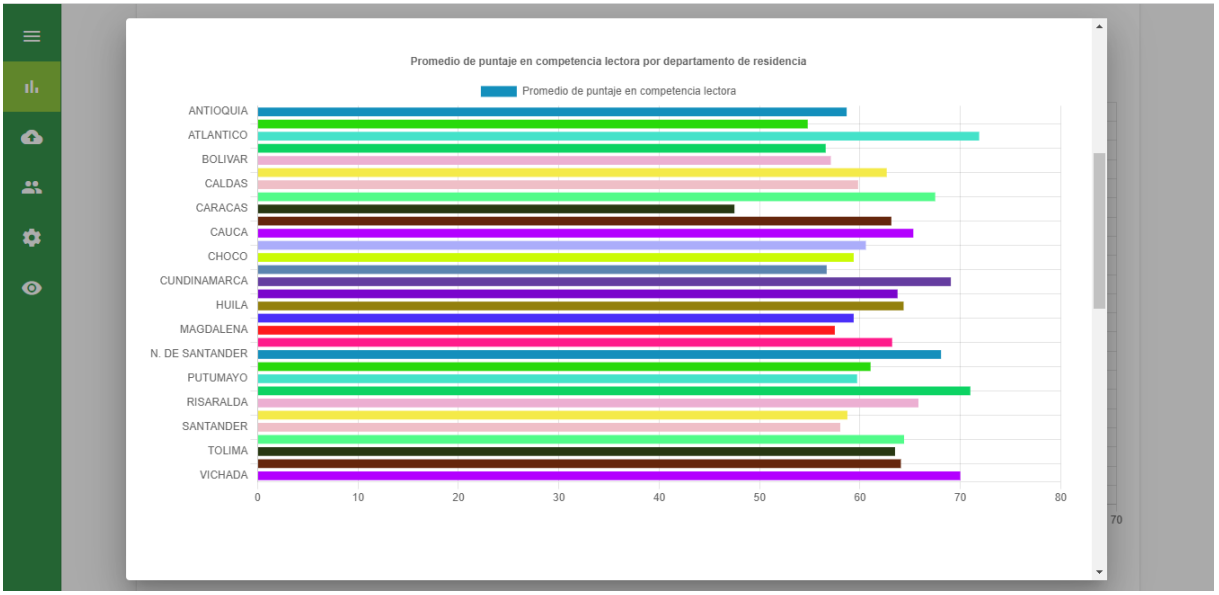


Gráfica 27. Promedio en Razonamiento Lógico por tipo de registro.

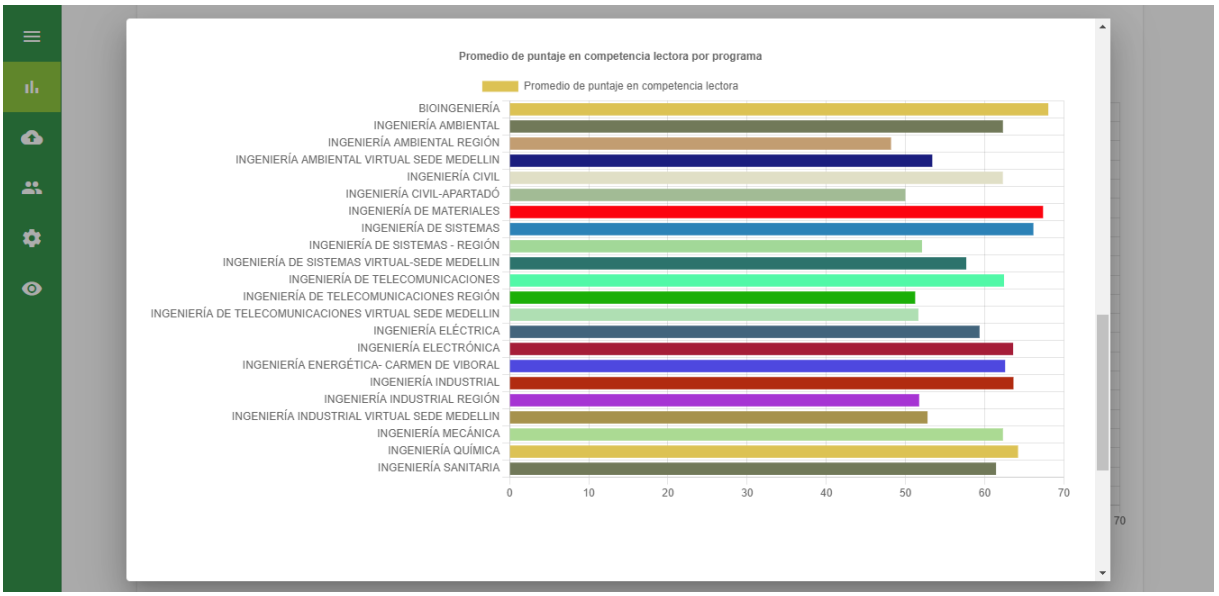
En las gráficas 28, 29, 30 y 31 se muestra la ventana que emerge al dar click en la barra de Competencia Lectora para la facultad de ingeniería.



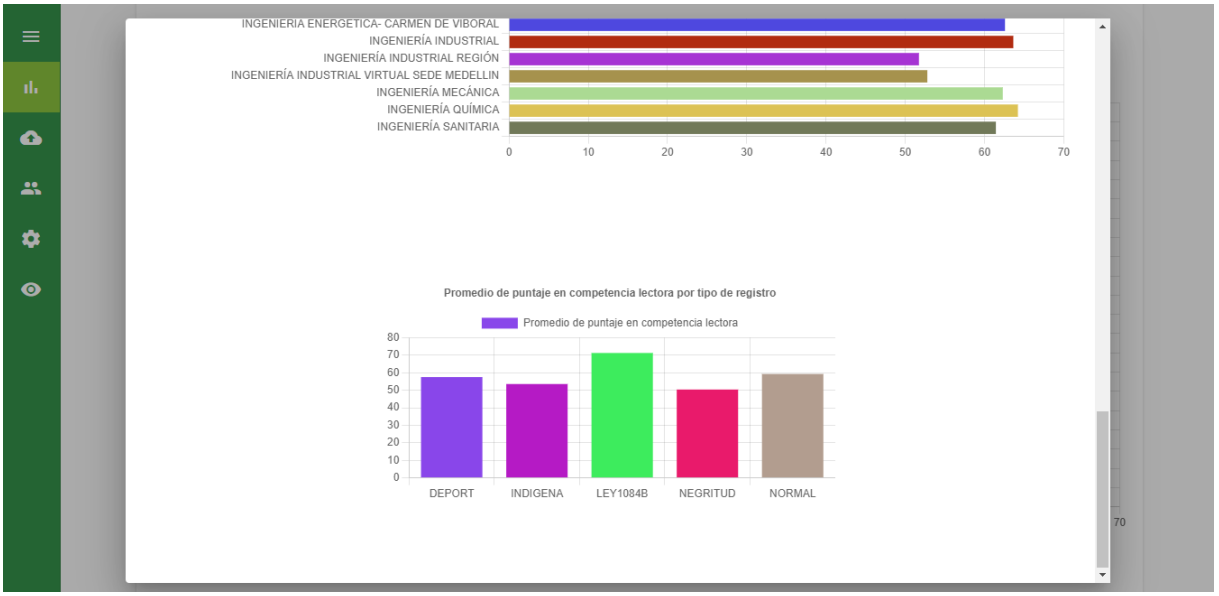
Gráfica 28. Promedio en competencia lectora por versión y por estrato social.



Gráfica 29. Promedio en competencia lectora por departamento de residencia.

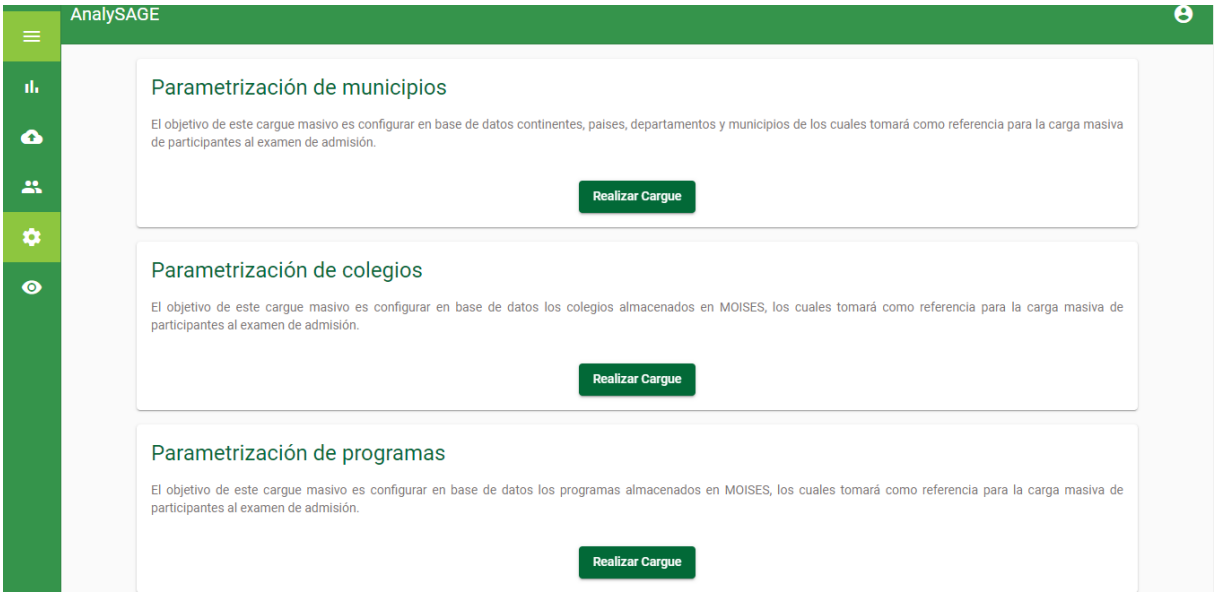


Gráfica 30. Promedio en competencia lectora por programa.

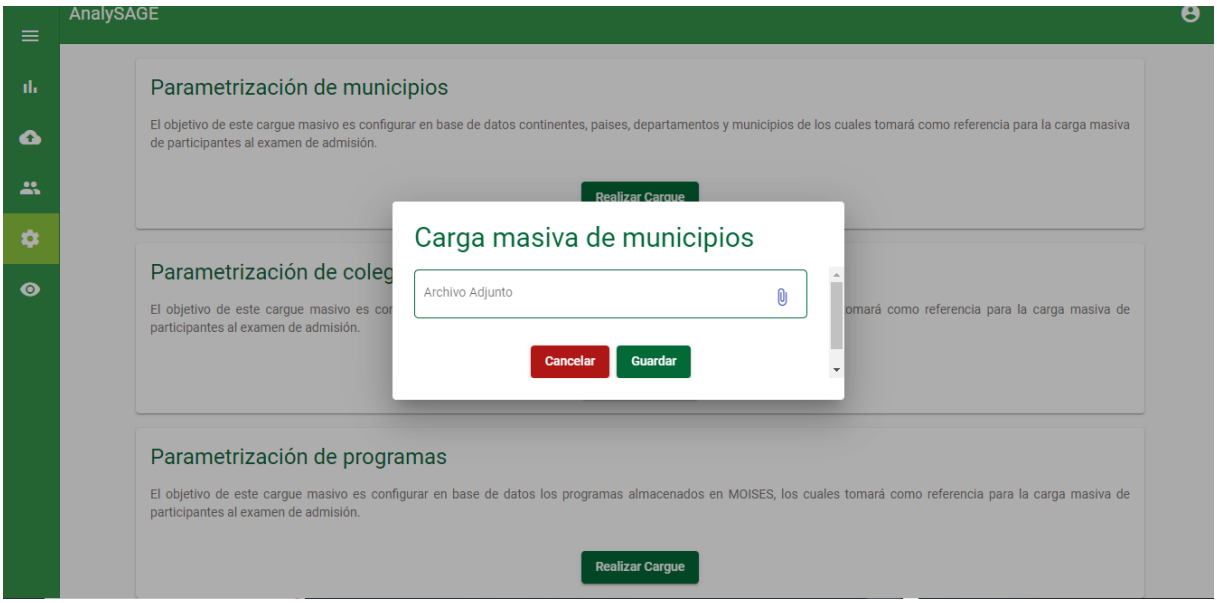


Gráfica 31. Promedio en competencia lectora por tipo de registro.

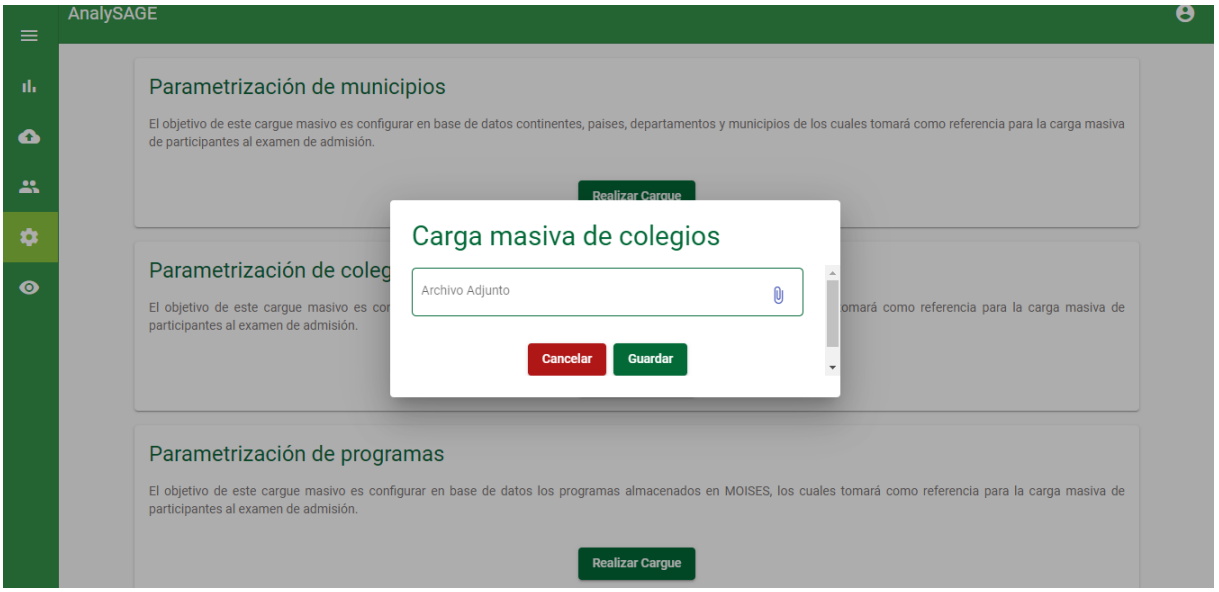
Para cumplir con el ***objetivo específico 4*** en la aplicación está el módulo para la parametrización de los municipios, colegios y programas. Este módulo permite hacer carga masiva de archivos tipo excel en los que se hace validación de contenido para validar que la información que se almacena tiene el formato correcto (ver gráficas 32, 33, 34 y 35).



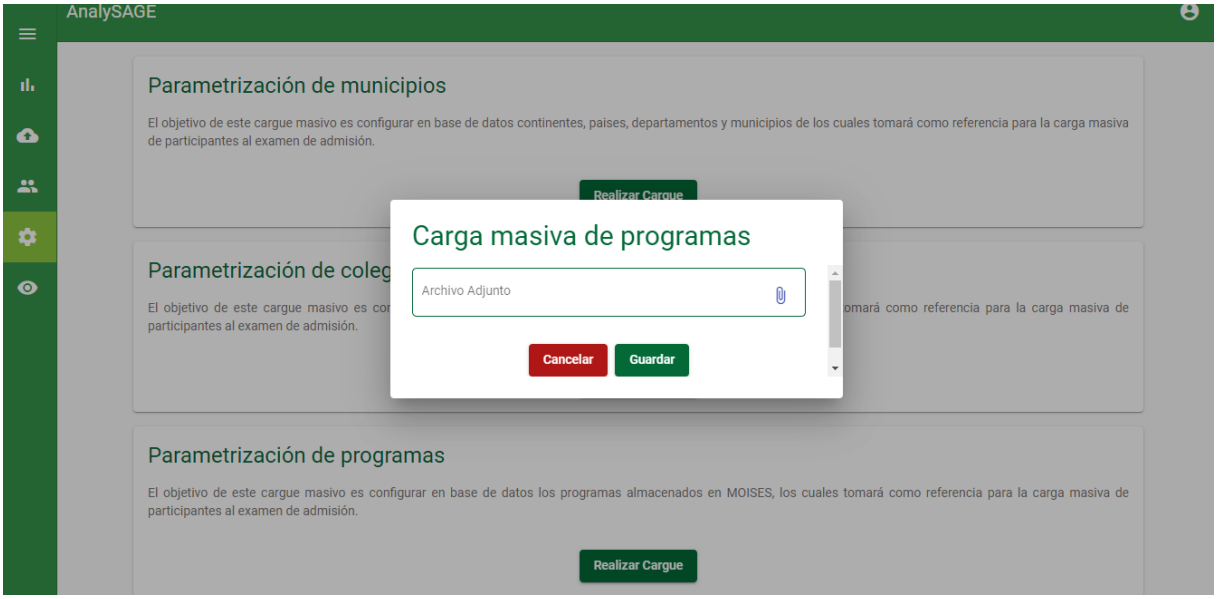
Gráfica 32. Módulo de configuración.



Gráfica 33. Modal emergente para hacer carga masiva de municipios.

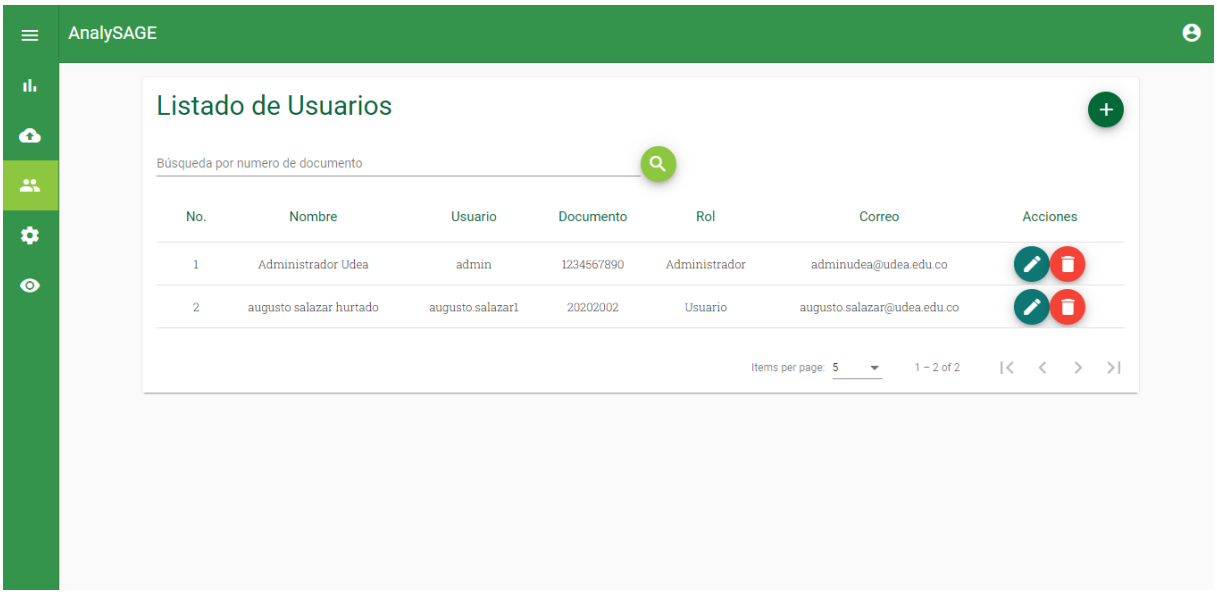


Gráfica 34. Modal emergente para hacer carga masiva de colegios.

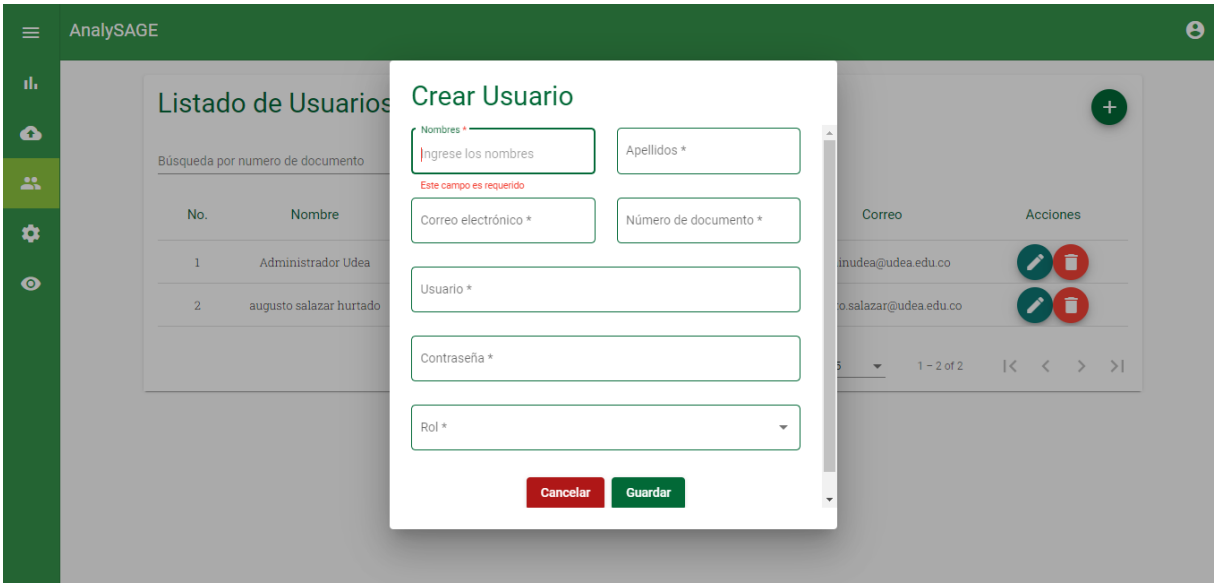


Gráfica 35. Modal emerge para hacer carga masiva de programas.

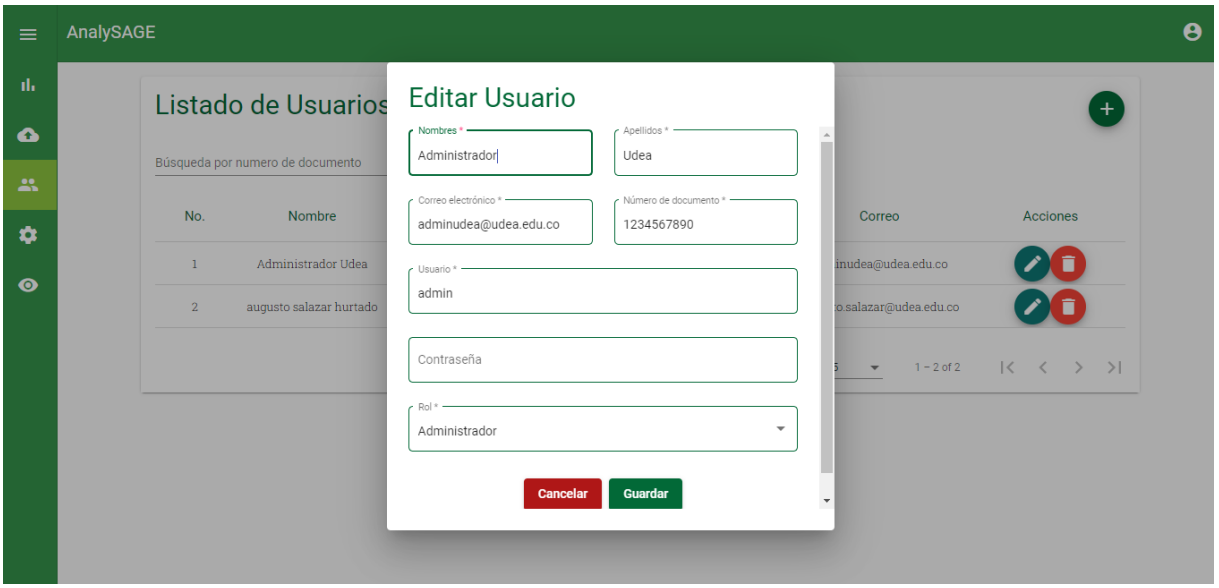
El módulo de usuarios permite crear usuarios, buscar, listar, eliminar y editar los que existen (ver gráficas 36, 37, 38 y 39).



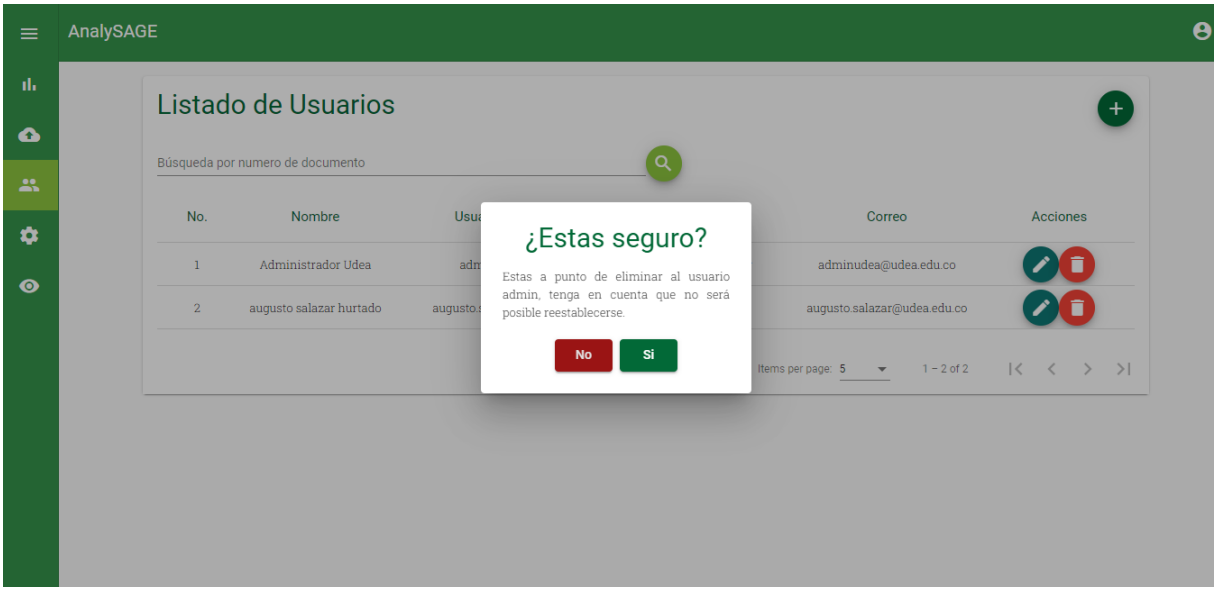
Gráfica 36. Módulo de usuarios.



Gráfica 37. Formulario de creación de usuarios.

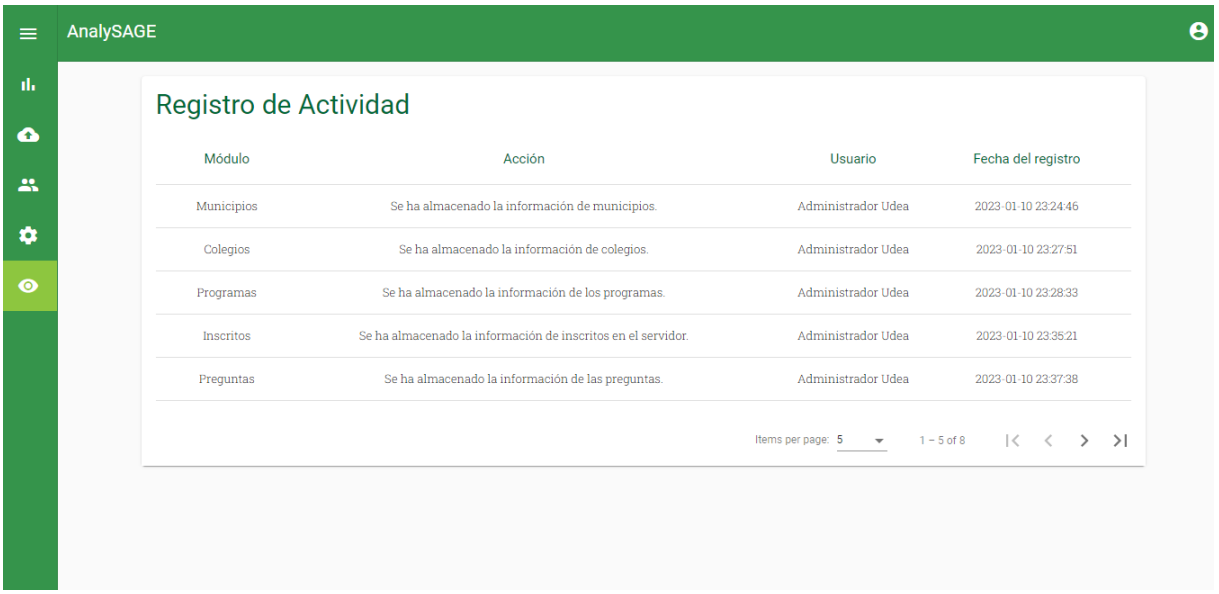


Gráfica 38. Edición de un usuario existente.



Gráfica 39. Eliminación de un usuario existente.

En la gráfica 40 se puede observar el módulo de Registro de Actividad en el que se podrá observar las acciones realizadas por los usuarios.

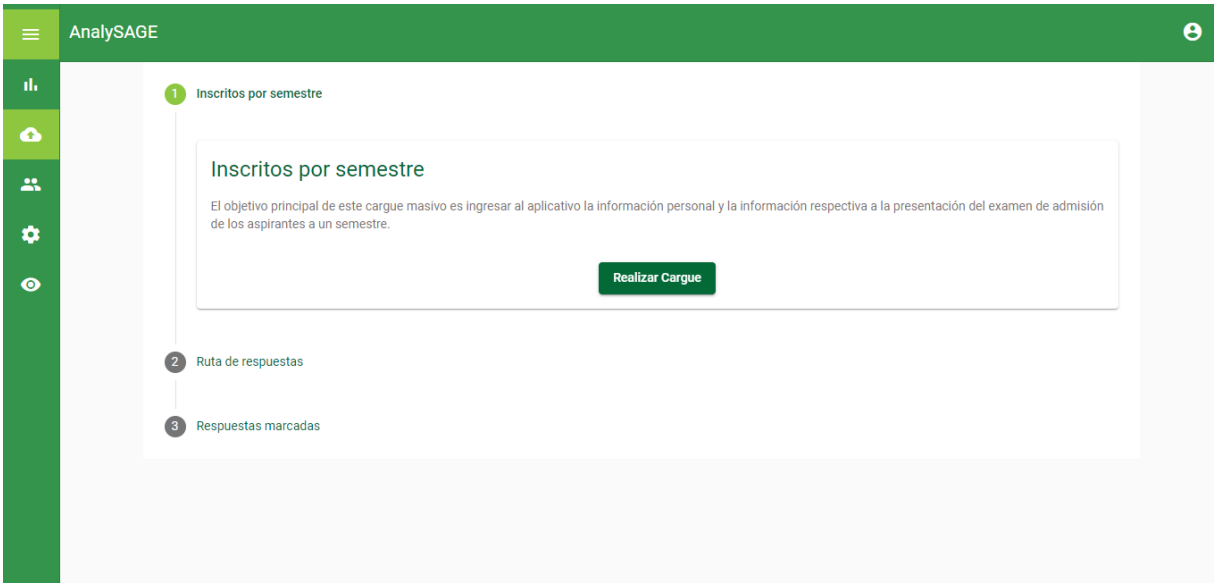


Gráfica 40. Módulo de registro de actividad.

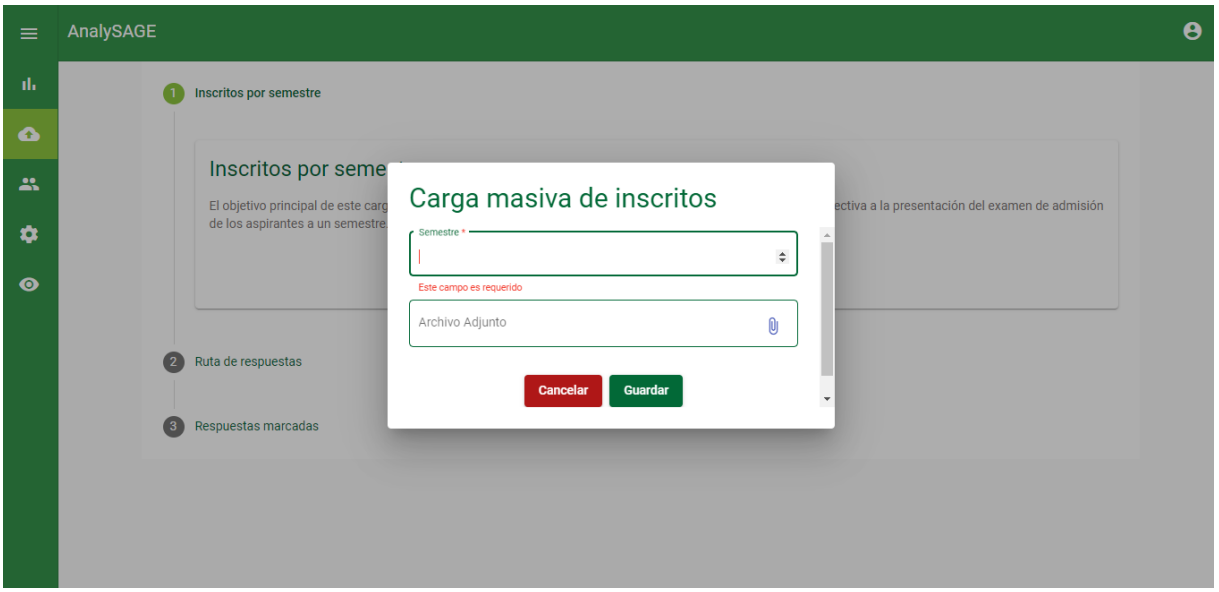
El módulo de cargas masivas permite hacer 3 cargas que tienen un orden establecido. Primero se debe subir la información de inscritos por semestre cuyo objetivo principal es ingresar al aplicativo la información personal y la información respectiva a la presentación del examen de admisión de los aspirantes a un semestre (ver gráficas 41 y 42).

La segunda carga tiene el objetivo de ingresar al aplicativo las respuestas correctas de la versión 1 del examen para las diferentes jornadas de un semestre (ver gráficas 43 y 44).

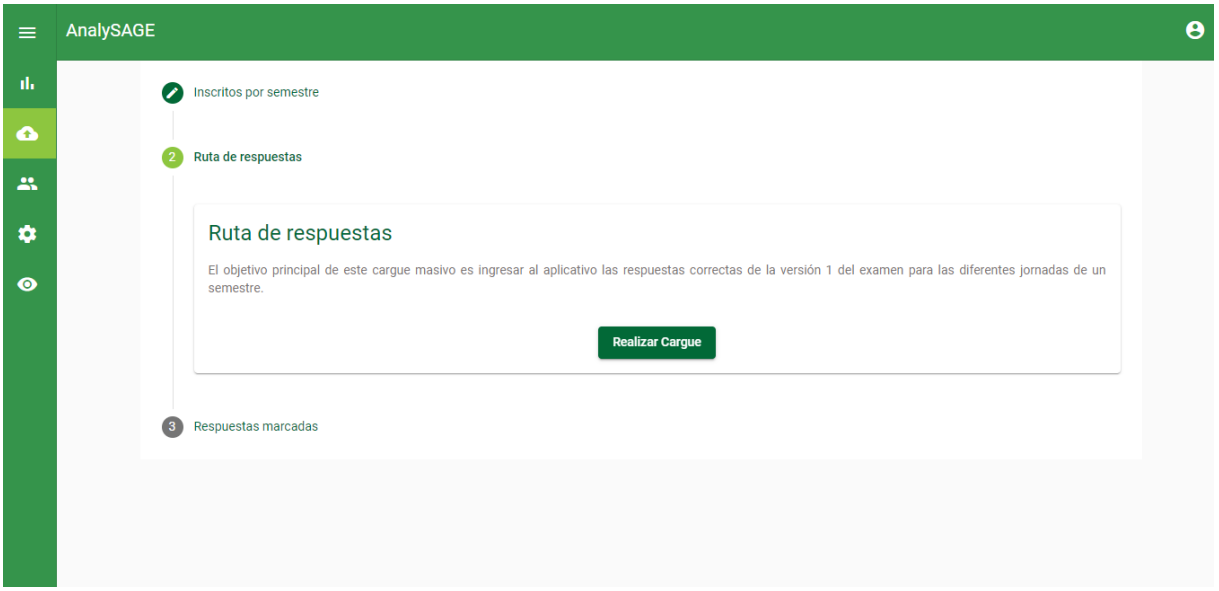
La tercera carga tiene el objetivo de ingresar al aplicativo las respuestas marcadas por cada uno de los aspirantes de un semestre (ver gráficas 45 y 46).



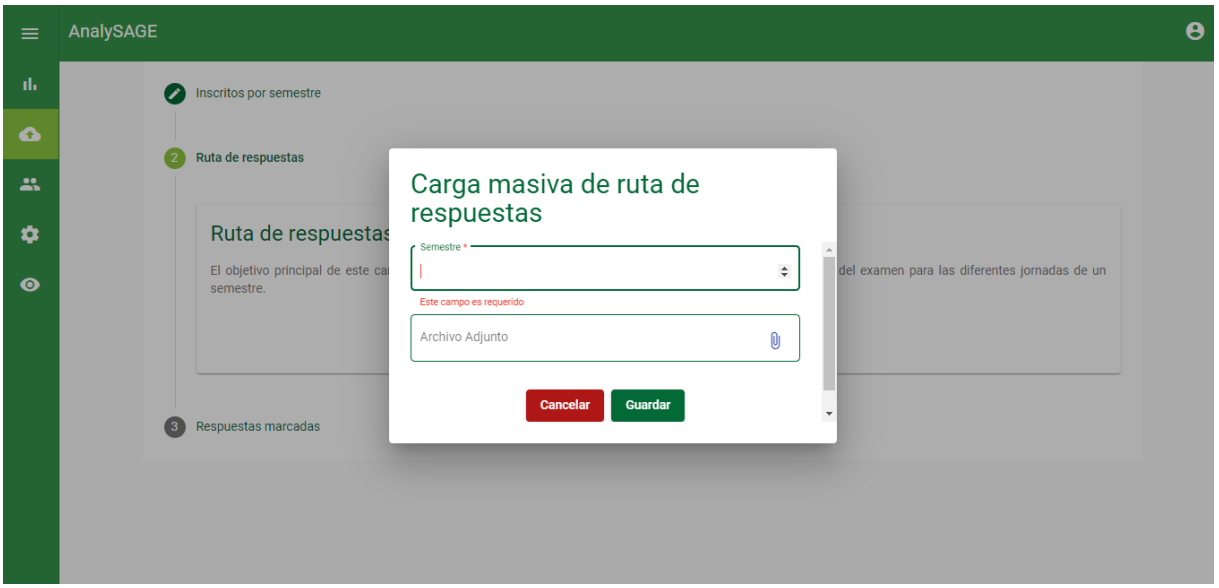
Gráfica 41. Paso 1 de carga de inscritos por semestre.



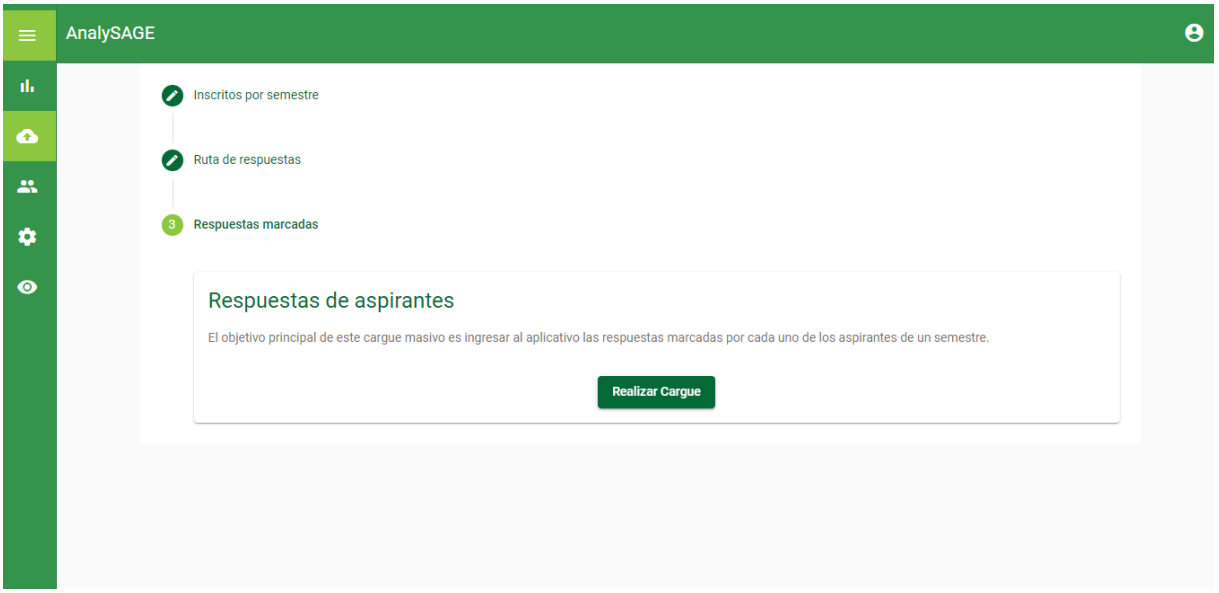
Gráfica 42. Modal emergente para carga masiva de inscritos.



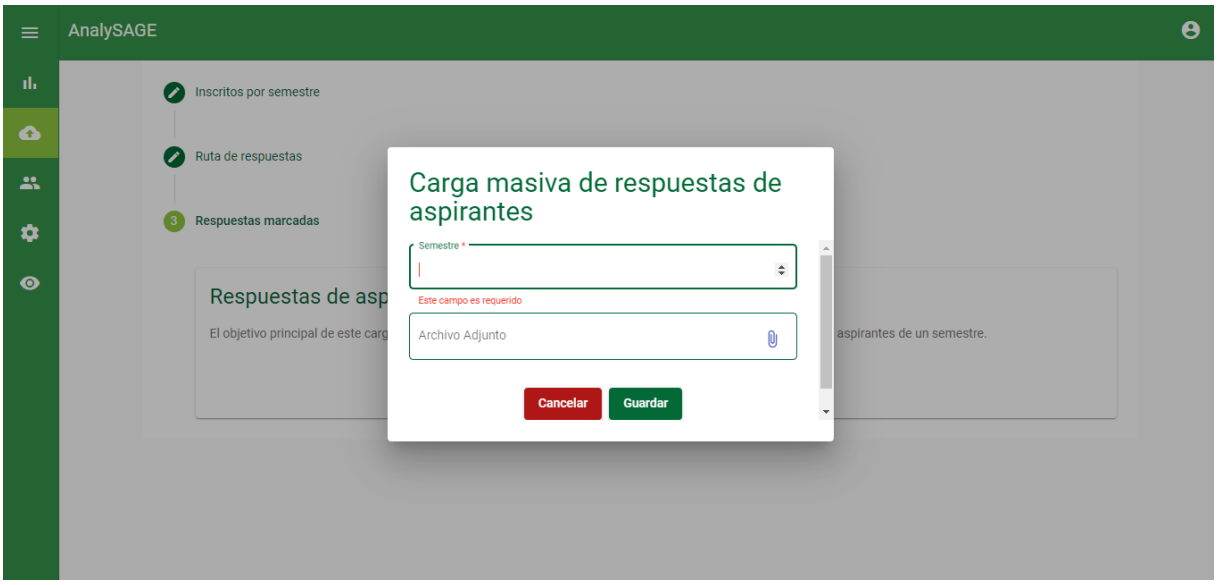
Gráfica 43. Paso 2 de carga de ruta de respuestas.



Gráfica 44. Modal emergente para carga masiva de ruta de respuestas.

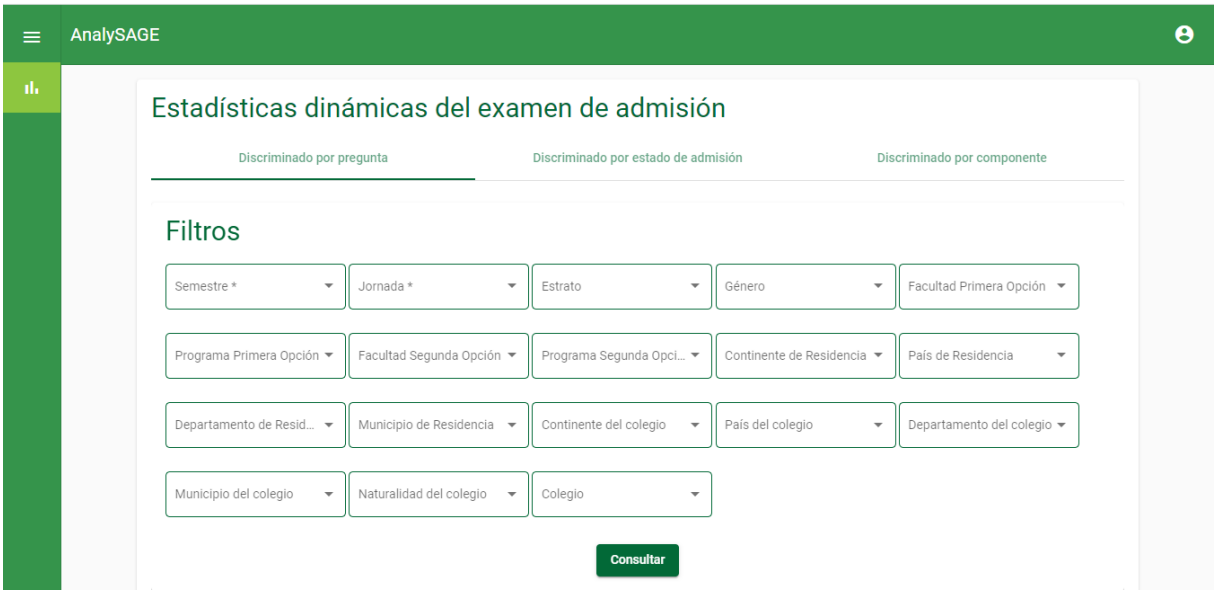


Gráfica 45. Paso 3 de carga de respuestas de los aspirantes.



Gráfica 46. Modal emergente para carga masiva de respuestas de aspirantes.

De las gráficas 2 a la 46 se muestran las posibilidades que tiene un usuario con rol Administrador. Sin embargo también existe el rol Usuario, el cual puede ver esencialmente 1 módulo, el cual es las estadísticas dinámicas (ver gráfica 47).



Gráfica 47. Módulo disponible para los usuarios con rol Usuario.

Conclusiones

1. El levantamiento de requerimientos fue muy importante durante todo el proceso, ya que permitió establecer los límites del proyecto. Sin haber realizado los requerimientos no se habría podido seguir un plan de trabajo que llevara a los resultados obtenidos.
2. El uso de buenas prácticas de programación contribuyó a que el código tanto del frontend como del backend quedara organizado, fácil de leer, escalable y mantenible. De tal manera que si en el futuro el área de Admisiones y Registro quisiera escalar o agregar nuevos módulos en la aplicación es completamente viable y va a ser fácil para los desarrolladores entender lo que actualmente está implementado.
3. El diseño previo del módulo de almacenamiento y gestión de la información permitió establecer un modelo de bases de datos relacional óptimo que contribuyó al cumplimiento del objetivo respecto a la carga de la información sobre la presentación del examen de admisión. El modelo de base de datos es tal que se pueden realizar otro tipo de funcionalidades en la aplicación respecto al examen de admisión y de sus aspirantes, por lo tanto queda abierta la posibilidad de escalar su uso.
4. El uso de consultas dinámicas usando el framework Laravel le dio un amplio panorama de posibilidades al departamento de Admisiones y Registro, respecto a las estadísticas. Se tuvieron buenas prácticas como el manejo de índices en las tablas de la base de datos, para optimizar el tiempo de las consultas ya que el volumen de información es significativamente alto.
5. El uso de la librería open source llamada Chart Js le dio un aspecto muy estético a todas las gráficas de la aplicación, esta librería permite realizar gráficas de barra y de torta, desplegando cantidades y porcentajes de forma dinámica de acuerdo a los filtros seleccionados por el usuario.
6. El uso de los frameworks Angular y Laravel permitió modularizar la aplicación al nivel de que permite entender muy fácilmente el código. Además se hizo uso de estos frameworks para proveer seguridad a la aplicación por ejemplo con el uso del JWT token para la validación de sesiones y roles por parte del backend y por parte del frontend se usaron guardianes que protegen las vistas de la aplicación. De esta manera no se puede hacer robo de roles o permisos que pongan en riesgo la integridad y la confidencialidad de los datos del examen de admisión.
7. Con la aplicación desarrollada se logró el objetivo de asistir al área de Admisiones y Registro en el análisis de los resultados del examen de admisión de la Universidad de Antioquia.

Cibergrafía

[1] Openclassrooms. 2017. [En línea]. Disponible en: https://blog.openclassrooms.com/es/2017/09/11/que-es-el-desarrollo-web/

[2] Rafarjonilla. 2016. [En línea]. Disponible en: https://rafarjonilla.com/que-es/backend/

[3] I. Bautista. Servnet. 2021. [En línea]. Disponible en: https://www.servnet.mx/blog/backend-y-frontend-partes-fundamentales-de-la-programacion-de-una-aplicacion-web

[4] M. Gonçalves. Hiberus. 2021. [En línea]. Disponible en: https://www.hiberus.com/crecemos-contigo/que-es-angular-y-para-que-sirve/

[5] F. Cristancho. Talently. 2022. [En línea]. Disponible en: https://talently.tech/blog/que-es-laravel/#:~:text=%C2%BFQu%C3%A9%20es%20Laravel%20en%20programaci%C3%B3n,de%20paquetes%20y%20extensiones%20compatibles.

[6] Wikipedia. 2022. [En línea]. Disponible en: https://es.wikipedia.org/wiki/Chart.js

[7] Paessler. 2023. [En línea]. Disponible en: https://www.paessler.com/es/it-explained/database

[8] Kikopalomares. 2023. [En línea]. Disponible en: https://kikopalomares.com/clases/que-es-una-web-spa-single-page-application

[9] R. Hernandez. Freecodecamp. 2021. [En línea]. Disponible en: https://www.freecodecamp.org/espanol/news/el-modelo-de-arquitectura-view-controller-pattern/

[10] Developer Mozilla. 2022. [En línea]. Disponible en: https://developer.mozilla.org/es/docs/Learn/JavaScript/Client-side_web_APIs/Introduction

[11] Ilimit. 2020. [En línea]. Disponible en: https://www.ilimit.com/blog/arquitecturas-monoliticas-o-arquitectura-de-microservicios-ventajas-e-inconvenientes/

[12] Decidesoluciones. 2019. [En línea]. Disponible en: https://decidesoluciones.es/arquitectura-de-microservicios/