INFORME DE PROYECTO DE GRADO

RESOLUCIÓN DE ECUACIONES POR MÉTODOS ARITMÉTICOS EN ENSEÑANZA MEDIA

VERSIÓN WEB

Bruno Michetti - Franco Pariani

Tutores
Sylvia da Rosa - Federico Gómez

Clientes
Nora Ravaioli - Teresa Pérez

UdelaR-InCo Montevideo, Uruguay
13 de diciembre de 2018

Resumen

El presente informe describe el proyecto de grado *Resolución de ecuaciones por métodos aritméticos en enseñanza media - Versión web*. El objetivo principal del proyecto consiste en la construcción de un sistema web que permite a docentes de matemática de educación media, organizar, gestionar y realizar un seguimiento del avance del aprendizaje de sus alumnos en la resolución de ecuaciones por el método "Cover Up" o "la tapadita". Este método consiste en seleccionar una parte de la ecuación que contenga a la incógnita y hallar su valor. Eso lleva a plantear una sub-ecuación, con la que se repite el proceso hasta que la forma de la sub-ecuación es *incógnita = valor*. Las ecuaciones son resueltas por los estudiantes en una aplicación de escritorio, que se desarrolló en un proyecto realizado en 2017, para ser usada sin conexión a Internet.

Por lo tanto, además de los aspectos propios de la construcción del sistema web, se plantearon los objetivos específicos de, por un lado, resolver la comunicación entre la aplicación web y la aplicación de escritorio, y por otro lado crear y añadir nuevas funcionalidades a la aplicación de escritorio que son significativas para que los docentes puedan realizar el seguimiento de los estudiantes.

Tanto el sistema web como la aplicación de escritorio, fueron creados a partir de especificaciones brindadas por las profesoras de Matemática en Educación Secundaria y de Didáctica de la Matemática en formación docente, Nora Ravaioli y Teresa Pérez.

El informe está organizado en las siguientes secciones:

- Introducción, donde se describe el problema planteado y las decisiones tomadas en base al análisis de aplicaciones relacionadas y distintas tecnologías, entre otros aspectos.
- Requisitos, donde se describen los funcionales tanto de la aplicación web como los agregados a la aplicación de escritorio, así como los requisitos no funcionales.
- Arquitectura y diseño, donde se describe la estructura sobre la cual se construyó la solución, a través de una vista lógica y física del sistema, de los diagramas de clases de diseño y de los modelos entidad-relación.
- Implementación y algoritmos, donde se describe la implementación de las funcionalidades del sistema, y los principales algoritmos que resuelven las cuestiones centrales del problema.
- Testing, donde se describen las pruebas realizadas, y la definición de las distintas estrategias y criterios tomados para las mismas.
- Conclusiones y trabajo futuro, donde se destacan algunos aspectos del trabajo realizado, relacionados principalmente a los objetivos planteados y al alcance del proyecto, y se dejan propuestas para futuros trabajos y/o proyectos.

Índice

1. Introducción

Este documento es el informe del proyecto de grado llamado *Resolución de ecuaciones por métodos aritméticos en enseñanza media - Versión web*, realizado por Bruno Michetti y Franco Pariani bajo la tutoría de Sylvia da Rosa y Federico Gómez. El objetivo principal del proyecto consiste en la construcción de un sistema web que brinda apoyo a docentes de matemática de educación media, para organizar, gestionar y realizar un seguimiento del avance del aprendizaje de sus alumnos en la resolución de ecuaciones por el método "Cover Up" o "la tapadita".

En el proyecto de grado realizado en 2017, Camila Rojí y Martín Poli desarrollaron un sistema que implementa el método mencionado que permite a los estudiantes resolver ecuaciones realizando solamente operaciones aritméticas. Este método consiste en seleccionar una parte de la ecuación que contenga a la incógnita y hallar su valor. Ello lleva a plantear una sub-ecuación, con la que se repite el proceso hasta que la forma de la sub-ecuación es *incógnita = valor*. Tanto el proyecto de 2017 como el que se presenta en este informe, han sido diseñados y elaborados de acuerdo a especificaciones provistas por las profesoras de Matemática en Educación Secundaria y de Didáctica de la Matemática en formación docente, Nora Ravaioli y Teresa Pérez.

Para referir al sistema del proyecto de 2017, se usará la expresión "aplicación de escritorio", dado que fue desarrollado para utilizarse sin conexión a Internet a pedido de las profesoras, con el objetivo de abarcar a la mayor cantidad posible de estudiantes y profesores del país. La aplicación de escritorio permite crear y resolver ecuaciones, y visualizar información básica del avance del alumno en su trabajo. En el presente proyecto se ha desarrollado un sistema web, de aquí en más referido como "aplicación web", que provee funcionalidades para brindar soporte a los docentes en cuestiones relativas a la organización, visualización y gestión de la información, que les permite un análisis más profundo sobre el proceso de aprendizaje de sus alumnos. Algunos ejemplos son: crear ecuaciones, clasificarlas, agruparlas en paquetes, registrar grupos de alumnos, asignar paquetes a determinados grupos, entre otras. La aplicación de escritorio pasa a formar parte del nuevo sistema, lo cual implica redefinir requisitos para la misma y extender sus funcionalidades (ver sección 2.3).

Este proyecto cuenta con tres tipos de usuarios, el usuario docente que puede realizar, entre otras cosas, las funcionalidades mencionadas, el usuario alumno que trabaja con la aplicación de escritorio, y el usuario coordinador que se encarga de clasificar las ecuaciones accesibles a todos los usuarios del sistema.

1.1. Problema planteado y primeras definiciones

El problema general puede definirse de la siguiente manera: a partir de la aplicación de escritorio, obtener un sistema web que permita a los docentes gestionar, organizar y analizar cuestiones relativas a la enseñanza-aprendizaje de la temática concerniente a la resolución de ecuaciones usando el método de "la tapadita" (Cover Up). La construcción de la solución comprendió varias etapas. Al comienzo del proyecto se realizaron reuniones con las profesoras Pérez y Ravaioli para determinar un conjunto inicial de características del sistema a desarrollar, que se resume en los siguientes puntos:

- Un sistema web en el cual los usuarios típicos son docentes de matemática que se registran con su correo electrónico.

- Dicho sistema debe comunicarse con la aplicación de escritorio, principalmente para subir el avance del trabajo de los alumnos y descargar ecuaciones.
- El docente debe poder registrarse fácilmente y poder visualizar el avance de sus alumnos que trabajan con la aplicación de escritorio.
- Tener una base de datos central con ecuaciones creadas por docentes de todo el país.

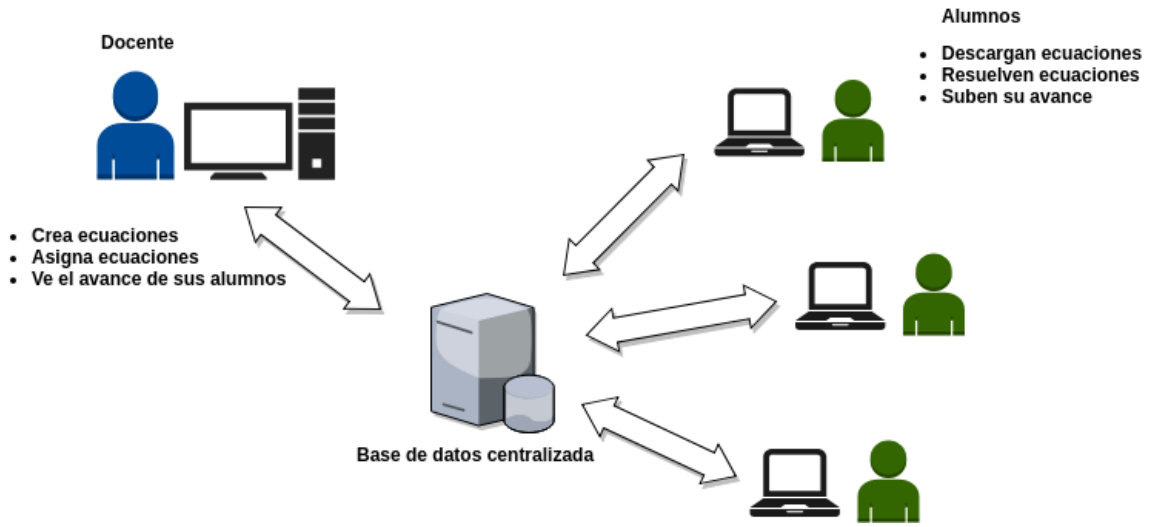


Figura 1: Esquema de los puntos principales

A partir de este conjunto inicial de características, se realizó un trabajo de investigación, con el objetivo de crear una propuesta de requisitos que reflejara lo discutido. El trabajo de investigación se centró en examinar aplicaciones ya existentes que puedan enfrentar de una forma u otra el problema planteado; en analizar diferentes lenguajes de programación y tecnologías que puedan usarse para la implementación del proyecto; y en el estudio de herramientas que resulten útiles para atacar las cuestiones centrales como lo son crear ecuaciones, y gestionar el avance de los alumnos en plataformas web.

1.2. Aplicaciones examinadas

En base a lo conversado con las profesoras, se examinaron aplicaciones ya existentes que puedan de cierta forma brindar una solución al problema, o por lo menos resolver alguno de los puntos que se plantearon. Se tomaron como puntos de partida los aspectos cruciales que se intercambiaron en las reuniones iniciales. Se enumeran a continuación las aplicaciones que se observaron y se usaron como insumo para la construcción de los requerimientos.

1.2.1. Moodle

Moodle[1] es una plataforma de aprendizaje creada para brindarle a educadores, administradores y estudiantes un sistema integrado único, robusto y seguro para la construcción y mantenimiento de ambientes de aprendizaje personalizados. Dicha plataforma

contempla la seguridad de los datos y la privacidad del usuario, implementando controles de seguridad que son constantemente actualizados, y protegiendo el software contra acceso no autorizado, pérdida de datos y mal uso, ya que puede ser desplegado fácilmente en un servidor, o en una nube segura privada para un control completo. Esta aplicación examinada es una plataforma construida de forma colaborativa, y es ampliamente usada en todo el mundo. Si bien tiene muchas funcionalidades que exceden ampliamente un posible alcance para un proyecto de grado, posee ciertas características que sirvieron como ideas para la propuesta de la solución. Las mismas son:

- Permite crear usuarios de tipo docente, estudiante y administrador.
- Permite registrar y administrar cursos de cualquier nivel educativo.
- Permite a los docentes realizar un seguimiento de sus alumnos, a través de:
  - Asignación de tareas que deben entregar los estudiantes mediante la plataforma.
  - Creación de pruebas en línea que pueden resolver los estudiantes, dejando un registro de sus resultados.
  - Definir períodos de tiempo en los que se debe realizar una actividad.
  - Crear encuestas estudiantiles.
  - Visualizar estadísticas sobre diferentes actividades realizadas por estudiantes.
  - Crear grupos de estudiantes para dividir las actividades o entregas que deben realizar según dichos grupos.

El entorno de aprendizaje virtual (EVA) de la Universidad de la República, es una plataforma Moodle, y actualmente la mayoría de los cursos de la universidad se encuentran registrados en dicho entorno.

1.2.2. ThatQuiz

ThatQuiz[2] es una plataforma web que permite registrar a docentes y estudiantes, y crear y administrar pruebas. Se trata de un recurso educativo abierto cuyas características principales se enumeran en los siguientes puntos:

- Permite crear pruebas de diferentes tipos, y sobre diferentes disciplinas.
- Permite registrar estudiantes y grupos de estudiantes para poder hacer un seguimiento de su trabajo.
- Los estudiantes pueden acceder de forma independiente a la plataforma y usarla tanto afuera como adentro del aula.
- Permite asignar diferentes niveles de dificultad a las diferentes pruebas.
- Permite visualizar resultados, tiempos de trabajo de los alumnos, y los puntos donde tienen mayores dificultades.
- Permite crear pruebas personalizadas y asignarlas a alumnos o a grupos.

Si bien se pueden hacer pruebas sobre geometría, idiomas, geografía, etc, el equipo investigó la parte de resolución de ecuaciones por su relación con el tema del proyecto. Se presenta una imagen en dicha plataforma:

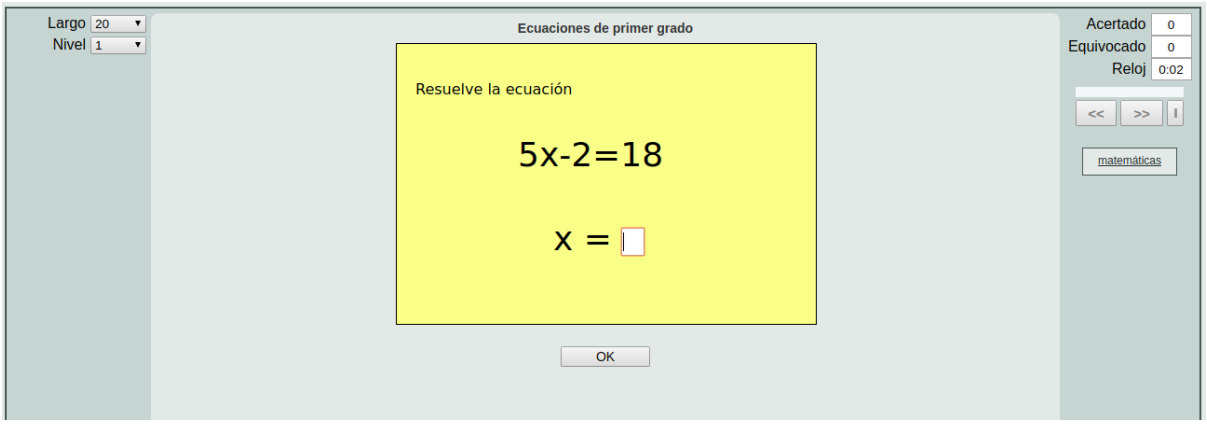


Figura 2: Ecuaciones en ThatQuiz.

Se concluye que lo que permite ThatQuiz es generar ecuaciones y corroborar las soluciones, midiendo el tiempo, sin contar con un proceso interactivo que permita seguir la resolución de la ecuación paso a paso, como lo hace la aplicación de escritorio.

1.2.3. MathPapa

MathPapa[3] es una aplicación web interactiva que le permite al usuario resolver problemas matemáticos visualizando la ejecución paso a paso hasta llegar a la solución. Los aspectos más importantes son:

- Las lecciones.
- Las prácticas.
- La calculadora.

En las lecciones se ofrecen interfaces de usuario interactivas en las que la aplicación presenta problemas matemáticos y asiste al usuario en el proceso de resolución, más desde un enfoque teórico que práctico, y posee además videos con explicaciones de algoritmos matemáticos como lo es la resolución de ecuaciones.

Las prácticas son propuestas de ejercicios sobre diferentes temas de matemática (ecuaciones, fracciones, decimales, etc) en las cuales el usuario debe ingresar la solución y la aplicación la verifica e indica el éxito o los posibles errores. Por último, la calculadora le permite al usuario ingresar un conjunto acotado de problemas en un formato especificado, y MathPapa lo resuelve mostrando cómo llegó a la solución paso a paso.

El equipo se interesó en este último punto, y se enfocó en el manejo e interacción de la aplicación respecto al ingreso y resolución de ecuaciones. A continuación, una imagen con un ejemplo de la calculadora:

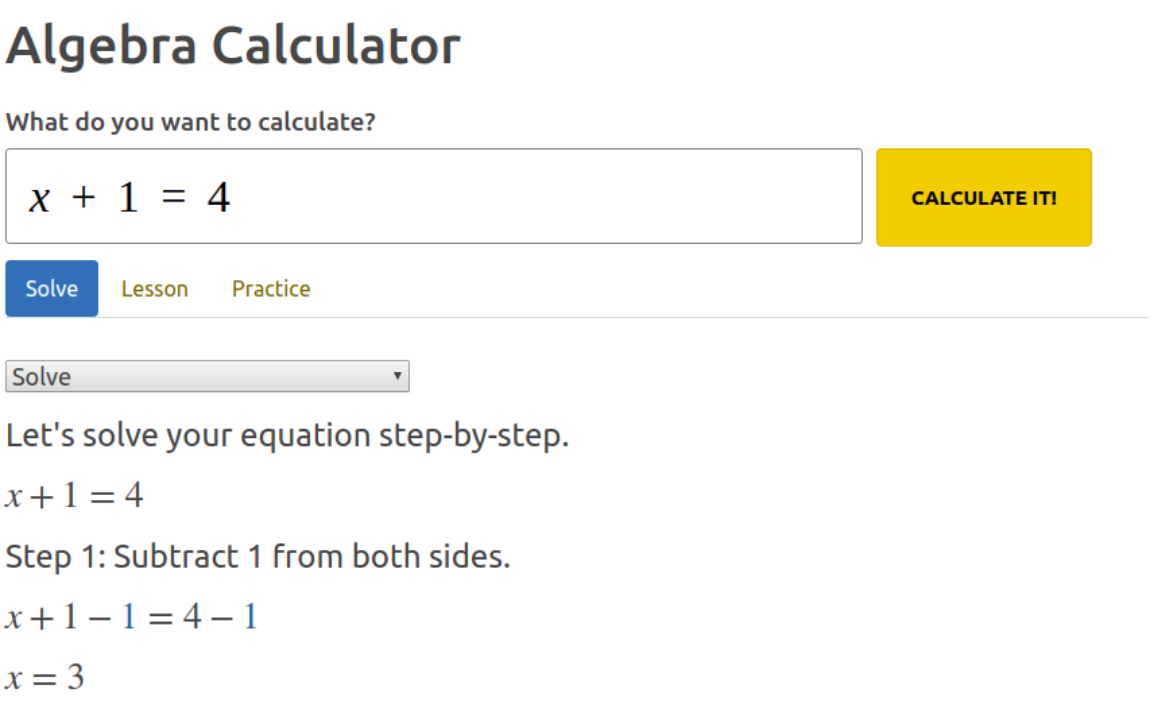


Figura 3: Resolución de ecuaciones con MathPapa.

1.3. Estudio de tecnologías

Una de las tareas centrales de la etapa de investigación fue el estudio de las tecnologías y lenguajes de programación que pudiesen satisfacer las necesidades del sistema. Se tuvo en cuenta distintos factores, como capacidad de procesamiento de las computadoras a utilizar, experiencias personales, tipo de sistema a desarrollar e interoperabilidad de las tecnologías. Se definieron los siguientes criterios para la selección:

- que los lenguajes y herramientas a utilizar sean gratuitos,
- que tengan versiones actualizadas y estables,
- que sean orientados al desarrollo de sistemas web.

Del análisis de diversos lenguajes y herramientas, surgieron las siguientes posibilidades para el desarrollo del sistema:

- Utilizar Angular 5[4] o ReactJs[5] para el front-end, es decir la programación del lado del cliente, y Ruby on Rails[6] o C# .Net[7] como back-end API, es decir para la programación del lado del servidor.
- Utilizar Ruby on rails tanto para front-end como para back-end.

Para conocer en profundidad la potencialidad de los lenguajes y herramientas seleccionados, se realizaron pruebas técnicas y tutoriales de uso básico de cada uno de ellos, que ayudaron a comprender sus ventajas y desventajas. Como resultado de este proceso se optó por la construcción de una API en Ruby on rails para dar soporte de back-end al sistema. Esta decisión se sustenta en la baja capacidad de procesamiento que requiere

Ruby on rails, en su facilidad y rapidez de desarrollo, y también en experiencias personales. A grandes rasgos, Ruby on rails es un framework para el desarrollo de aplicaciones web, escrito en el lenguaje Ruby, que se basa en dos principios:

- DRY(del inglés *Don't Repeat Yourself*): fomenta la modularización y reutilización, y plantea que escribir el mismo código una y otra vez, es una mala práctica.
- Convención sobre configuración: posee conjuntos de configuraciones por defecto y le quita al programador la necesidad de tener muchos archivos extensos de configuración.

Por otro lado, se realizó un análisis de interoperabilidad entre tecnologías para el lado del cliente y Ruby on rails, optando por desarrollar el front-end del sistema con Angular 5. Si bien es posible desarrollarlo tanto con Ruby on rails como con ReactJs, se decidió utilizar Angular 5 para aprovechar la experiencia de uno de los integrantes del proyecto con dicho lenguaje y reducir la curva de aprendizaje[1]. En pocas palabras, Angular es un framework para aplicaciones web desarrollado en TypeScript[8], de código abierto que permite construir y mantener aplicaciones web en HTML y Typescript. Luego de la elección, el equipo se dedicó a interiorizarse en el uso de los lenguajes elegidos, al mismo tiempo que fue desarrollando un documento con posibles requerimientos a discutir con las profesoras.

1.4. Modelo de proceso

Al comienzo del proyecto, se estudiaron los posibles modelos de proceso a seguir, tomando como punto de partida el problema planteado. Dada la naturaleza del mismo el equipo decidió utilizar el modelo RUP (del inglés Rational Unified Process) que tiene componentes de los tres modelos genéricos más conocidos en la ingeniería de software: modelo en cascada, desarrollo incremental, e ingeniería orientada a la reutilización. La naturaleza del problema implica un proceso no demasiado rígido, en el que si bien es importante la retroalimentación de las profesoras para facilitar la realización de cambios, no es indispensable su constante participación. Por estas razones, se optó por el modelo RUP, modelo que además propone un conjunto de buenas prácticas que se consideran útiles y necesarias para el proyecto: desarrollo iterativo, gestión de requerimientos, uso de arquitectura basada en componentes, verificación de calidad, control de cambios y modelado visual. Este modelo permite desarrollar el producto en base a las prioridades establecidas por el cliente, generando una base sólida e incrementando su valor a medida que transcurre el tiempo. El modelo elegido se compone de cuatro fases: concepción, elaboración, construcción y transición. La planificación de cada fase se explica en el ítem 8.9 (Plan de proyecto) de la sección Anexos.

1.5. Prototipos

Para estudiar la factibilidad de poder atacar las cuestiones principales del problema planteado utilizando los lenguajes y herramientas seleccionados, se realizaron prototipos que fueron validados por las profesoras, y que permitieron conocer frameworks y librerías

[1]La curva de aprendizaje se utiliza para describir la dificultad en el aprendizaje de una tarea. La experiencia es un vehículo de aprendizaje que reduce dicha dificultad.

útiles para enfrentar los desafíos técnicos. Los prototipos realizados tuvieron como eje central los siguientes puntos:

- Creación y visualización de ecuaciones en la aplicación web.
- Visualización desde la aplicación web del avance de los alumnos en la resolución de ecuaciones, llevada a cabo en la aplicación de escritorio.

Sobre el primer punto mencionado, se tenía claro que algo importante a lograr en la aplicación web era permitir al docente crear ecuaciones de forma amigable y sencilla, por lo que se buscaron tecnologías para la visualización de símbolos y fórmulas matemáticas compatibles con Angular. La primera herramienta probada con éxito fue MathJax [9], una biblioteca JavaScript que permite visualizar fórmulas matemáticas en navegadores web utilizando LaTeX, de licencia libre y con soporte a múltiples navegadores. En la página oficial de Mathjax se encuentra disponible una demo de cómo escribir las fórmulas matemáticas y de cómo se ven en un navegador:

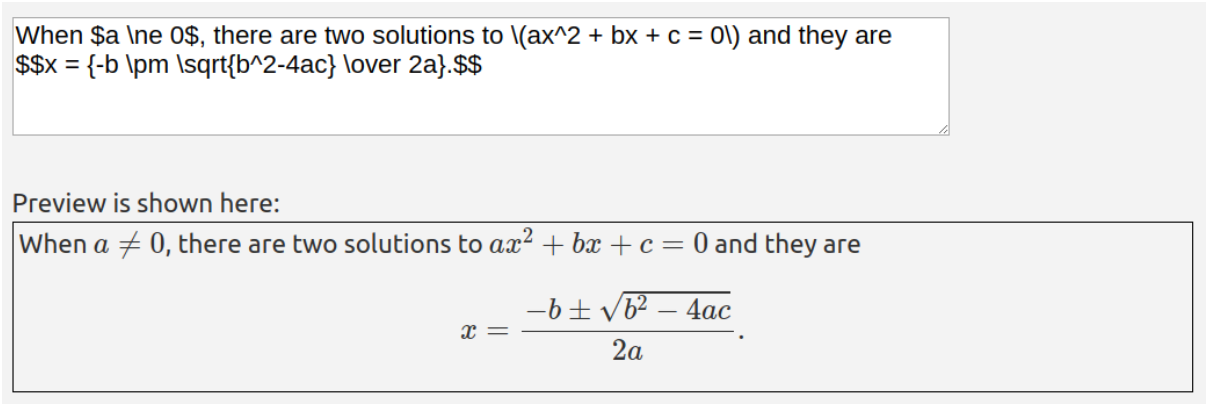


Figura 4: Demo de mathjax

Como se puede apreciar, en el cuadro superior se tiene el texto en crudo que se debe ingresar, para que sea procesado por Mathjax y se despliegue la vista de la fórmula en el cuadro inferior. Dado que es compatible con Angular, se comenzó a trabajar en un prototipo construido en dicho lenguaje, que incorpora Mathjax y se adapta a la creación y visualización de ecuaciones, como se puede ver a continuación:

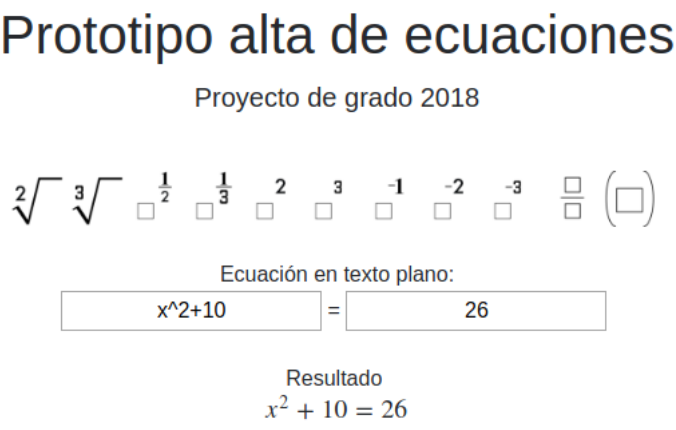


Figura 5: Prototipo de alta ecuación

Durante el desarrollo del sistema, surgieron inconvenientes con la compatibilidad de MathJax y las últimas versiones de Angular, por lo que se investigaron otras bibliotecas que fueran compatibles con Angular y cumplieran el rol que cumplía Mathjax. La herramienta finalmente utilizada es ngKaTeX[10], un módulo de Angular que utiliza una librería JavaScript llamada KaTeX, que permite visualizar fórmulas matemáticas en páginas web.

Para atacar el segundo punto central en los prototipos, es decir, el de visualizar el avance de los alumnos en la aplicación de escritorio a través de la aplicación web, se trabajó en dos aspectos:

- Visualización a través de gráficas, sobre estadísticas de los estudiantes en un grupo trabajando con ecuaciones.
- Visualización a través de una matriz, donde cada entrada **[i,j]** de la misma corresponde al estado de resolución de la ecuación **j**, realizada por el alumno **i**.

Para las gráficas se utilizó la librería JavaScript ChartJs[11] la cuál permite crear gráficas HTML5, y para las matrices se utilizó únicamente una tabla con estilos de Bootstrap[12]. A continuación, se presentan los prototipos realizados:

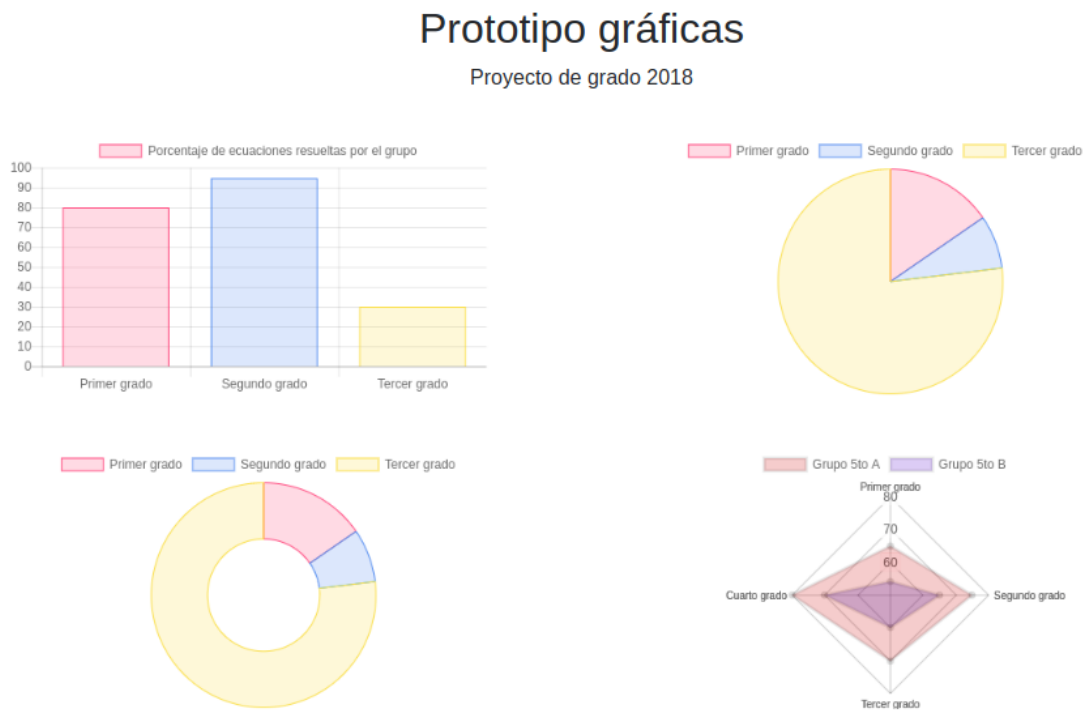


Figura 6: Prototipo de gráficas

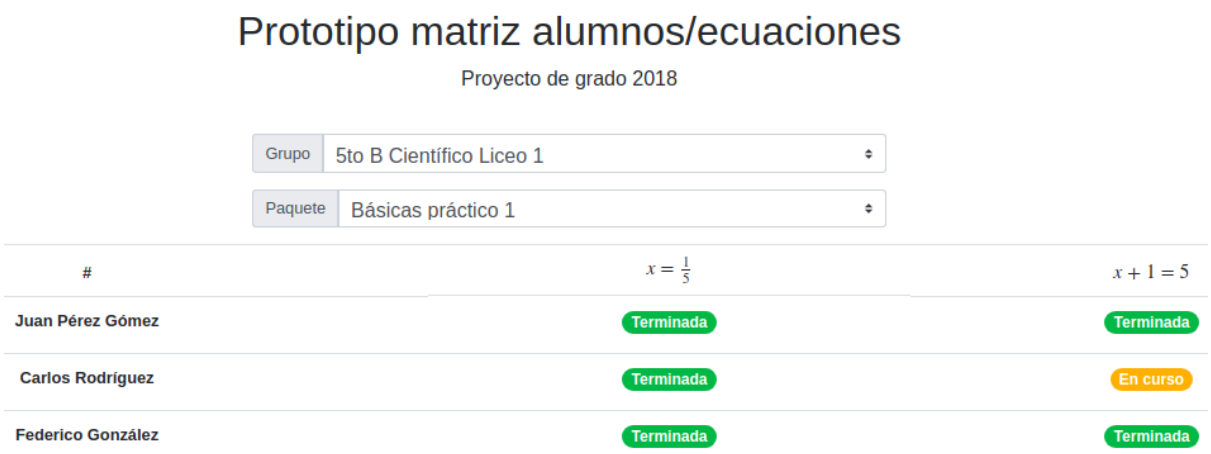


Figura 7: Prototipo de matriz de progreso

Los prototipos mencionados fueron presentados a las profesoras, que optaron por emplear solamente la matriz para la visualización del avance de los alumnos.

2. Requisitos

La superación de los obstáculos encontrados en los prototipos y la aprobación de los mismos por parte de las profesoras ayudó a definir con claridad la propuesta de requisitos que se describe a continuación.

2.1. Descripción general

A partir del problema planteado y de las primeras definiciones descritas en la sección anterior, se creó un documento de requisitos para el desarrollo de un sistema que pudiese dar solución a dicho problema, y se adecuara al tamaño y tiempo de trabajo correspondiente a un proyecto de grado. A grandes rasgos, se trata de una aplicación web en la que los usuarios son típicamente docentes de matemática de educación media que gestionan, crean, y asignan a sus alumnos conjuntos de ecuaciones, para que éstos puedan trabajar en la aplicación de escritorio. Esta última se comunica con la aplicación web para descargar las ecuaciones asignadas, y subir el avance del alumno sobre aquellas con las que viene trabajando, de manera de que el docente pueda hacer un análisis sobre el aprendizaje. Además cada ecuación creada es almacenada en una base de datos central, y puede ser utilizada por cualquier otro docente registrado para trabajarlas en la aplicación de escritorio con sus respectivos grupos de alumnos. En resumen, el presente proyecto de grado tiene tanto requisitos para el nuevo sistema web, como para la aplicación de escritorio desarrollada en el proyecto previo, como se describe en los siguientes puntos:

- El docente puede registrarse en la aplicación web con su correo electrónico.
- El docente puede crear ecuaciones mediante una botonera que provee los operadores raíz, potencia, cociente y paréntesis, al igual que las que se pueden construir en la aplicación de escritorio.
- El sistema cuenta con un conjunto inicial de ecuaciones predefinido.
- El docente puede clasificar las ecuaciones que crea, y visualizarlas aplicando filtros a partir de las diferentes categorías (ver Tabla 1).
- Las ecuaciones creadas se comparten automáticamente y se encuentran disponibles en la base de datos para cualquier usuario de la aplicación web.
- El docente puede agrupar las ecuaciones en paquetes.
- El docente puede registrar grupos de alumnos y asignar a los mismos paquetes de ecuaciones para que trabajen desde la aplicación de escritorio.
- Cada alumno se puede asociar a la aplicación de escritorio, para luego subir el avance de su trabajo (ver sección 2.3.1).
- El docente puede visualizar el avance de cada alumno respecto de cada ecuación asignada. Para un alumno y una ecuación, el avance se describe a través de un estado que puede ser: *sin trabajar, trabajada, terminada* o *no sincronizada* (ver Figura 7).

- El sistema permite sincronizar la información entre la aplicación web y la de escritorio, de manera de que en el lado del docente se pueda ver el avance en la resolución de ecuaciones, y en el lado del alumno se puedan recibir las nuevas ecuaciones para trabajar.

En las secciones 2.2 y 2.4 se desarrollan los requisitos funcionales y no funcionales respectivamente, definidos dentro del alcance del proyecto para la aplicación web.

2.2. Requisitos funcionales de la aplicación web

En esta sección se presentan los requisitos funcionales de la aplicación web. Es importante recordar que existen dos tipos de usuarios para dicha aplicación: docentes y coordinadores. El usuario típico de la aplicación web es el docente, que puede realizar las funcionalidades descritas en esta sección, y el coordinador puede ver y clasificar las ecuaciones accesibles por todos los usuarios.

2.2.1. Registro de docente

Los docentes se registran en el sistema ingresando nombre, apellido, fecha de nacimiento, e-mail y contraseña. Cada nuevo docente registrado se identifica dentro del sistema por su e-mail.

2.2.2. Inicio de sesión

Luego de registrado el docente, debe iniciar sesión en el sistema ingresando su e-mail y contraseña. El inicio de sesión es requerido para realizar cualquier tipo de acción dentro del sistema.

2.2.3. Cierre de sesión

Luego de iniciar sesión, el docente puede cerrarla y abrirla nuevamente cuando desee.

2.2.4. Registro de grupos de alumnos

El docente puede registrar en el sistema a sus grupos de alumnos. Para ello debe indicar un nombre para el grupo, su nivel (si es primaria, secundaria o UTU), el departamento del país en el que se encuentra, si pertenece a una institución pública o privada, y el año de dictado. El grupo registrado se identifica con un código alfanumérico generado automáticamente por el sistema, de 3 letras mayúsculas seguidas de 3 números. Dicho código sirve para que cada alumno del grupo pueda asociarse al mismo desde la aplicación de escritorio. Cada grupo existente en el sistema tiene su único código.

2.2.5. Creación de ecuaciones

El docente registrado puede crear ecuaciones de hasta tercer grado (lineales, cuadráticas y/o cúbicas), racionales, con radicales o combinaciones de estas, con una sola incógnita que puede aparecer en cualquier lado de la igualdad. Las ecuaciones se construyen ingresando texto que respete el formato definido para el sistema, o utilizando botoneras que generan el texto correspondiente a: raíz cuadrada, raíz cúbica, potencia (que puede ser: 2,

3, -1, -2, -3, 1/2 o 1/3), división y paréntesis. Además se pueden ingresar los operadores básicos de suma(+), resta(-) y multiplicación(*). El conjunto de ecuaciones posibles de la aplicación web es el mismo que el de la aplicación de escritorio. Al crear una ecuación el docente puede clasificarla seleccionando alguna clasificación de las categorías presentadas en la siguiente tabla:

Categoría	**Clasificación**
Tipo de ecuación	Lineal
	Cuadrática
	Cúbica
	Racional
	Irracional
	Otra
	Sin clasificar
Estructura	Incógnita en primer miembro
	Incógnita en segundo miembro
	Sin clasificar
Denominación de la incógnita	x
	y
	Otra
	Sin clasificar
Conjunto solución	Vacío
	$S \subset \mathbb{N}$
	$S \subset \mathbb{Z}$
	$S \subset \mathbb{Q}$
	Otro
	Sin clasificar

Tabla 1: Categorías y sus respectivas clasificaciones.

Por defecto, la clasificación para cada categoría es *sin clasificar*, y el docente puede cambiar alguna o todas. La ecuación creada se almacena en la base de datos central de forma automática y se hace visible para todo docente que se registre en el sistema. La clasificación que un docente le asigna a la ecuación, es únicamente visible para él, es decir que para cada docente y cada ecuación que crea, existe una clasificación. Cada docente tiene su conjunto de ecuaciones creadas, y puede ver el conjunto de ecuaciones del sistema, llamado "conjunto global" de ecuaciones.

Cada ecuación tiene dentro del sistema un número entero que la identifica, y no existen dos instancias de ecuaciones iguales en el sistema; es decir que si se crea una ecuación con los mismos elementos (operaciones, números, incógnita) y en el mismo orden de una ecuación ya existente, el sistema la acepta pero no genera un identificador nuevo. En la sección 4.6 se explica detalladamente el proceso de crear y clasificar una ecuación, así como la relación con el conjunto global de ecuaciones.

2.2.6. Creación de paquete de ecuaciones

El docente puede agrupar ecuaciones (creadas o del conjunto global) y crear paquetes de ecuaciones. Para crear un paquete debe ingresar un nombre, una descripción, y selec-

cionar las ecuaciones que formarán parte del mismo. Un paquete puede contener una sola ecuación.

2.2.7. Visualización de ecuaciones

El docente puede ver las ecuaciones creadas por él y las del conjunto global. Para cada ecuación se visualiza su identificador global y sus distintas clasificaciones según la categoría (ver Tabla 1). Para visualizarlas puede aplicar filtros según la clasificación.

2.2.8. Visualización de paquete de ecuaciones

El docente puede ver los paquetes de ecuaciones creados. Para cada uno se muestra el nombre y la descripción. Además puede seleccionar un paquete y ver la o las ecuaciones que contiene.

2.2.9. Clasificación de ecuaciones creadas

Al crear una ecuación, el docente puede clasificarla según la Tabla 1 y puede volver a hacerlo cada vez que lo desee.

2.2.10. Asignación de paquetes de ecuaciones a grupo de alumnos

El docente puede asignar uno o más paquetes a un grupo de alumnos. Tanto los paquetes como el grupo, deben haber sido creados por el docente. Esto asocia las ecuaciones pertenecientes a cada paquete seleccionado, con el grupo en cuestión, para que posteriormente el alumno que se registre en ese grupo, pueda descargarlas y trabajar con ellas.

2.2.11. Ver avance de grupos de alumnos

El docente puede seleccionar un grupo, y un paquete, para ver el avance de los alumnos de dicho grupo con respecto a las ecuaciones pertenecientes al paquete. El avance se representa con una matriz donde cada entrada **[i,j]** de la misma corresponde al estado de resolución de la ecuación **j**, realizada por el alumno **i**. Los estados son los mencionados anteriormente: *sin trabajar, trabajada, terminada* o *no sincronizada.*

2.2.12. Clasificación de ecuaciones del conjunto global de ecuaciones

Cuando un docente crea una ecuación, puede clasificarla, pero dicha clasificación es visible sólo para él. Además de lo mencionado, cada ecuación del sistema posee una clasificación que se denomina global y se le asigna automáticamente el valor *sin clasificar* en cada categoría. El coordinador puede clasificar y reclasificar las ecuaciones pertenecientes al conjunto global de ecuaciones. La clasificación de los coordinadores afecta sólo a la clasificación global sin cambiar la clasificación asignada por el docente que la creó.

2.2.13. Recuperación de contraseña

El docente y/o coordinador puede recuperar su contraseña en caso de que la olvide, utilizando su correo electrónico.

2.2.14. Borrado de grupo

El docente puede borrar un grupo del sistema. Dicho grupo debe haber sido registrado por el docente.

2.3. Nuevos requisitos funcionales de la aplicación de escritorio

En esta sección se enumeran los nuevos requisitos de la aplicación de escritorio, definidos para poder extender el sistema y comunicarlo con la aplicación web.

2.3.1. Asociación de alumno con la aplicación de escritorio

Cualquier alumno se puede asociar a la aplicación de escritorio, ingresando sus datos: nombre, apellido y cédula. Puede asociarse un alumno a la vez, y cada alumno asociado puede sincronizar su información.

2.3.2. Registro de alumno en grupo

El alumno que esté asociado a la aplicación puede registrarse a un grupo utilizando el código generado por el sistema y brindado por su docente. Puede registrarse a más de un grupo.

2.3.3. Sincronización de información del alumno

El alumno que esté asociado a la aplicación puede sincronizar información dentro de los grupos a los que esté registrado. Al seleccionar un grupo y elegir la opción de sincronizar, se sube el avance del alumno con respecto a las ecuaciones de los paquetes asignados al grupo. Además se bajan a la aplicación las nuevas ecuaciones que se hayan asignado al grupo.

2.3.4. Borrado de grupo

El alumno asociado a la aplicación de escritorio puede borrarse de un grupo al que previamente se registró.

2.3.5. Cambio de funcionalidad: Descargar ecuaciones

En la aplicación de escritorio se tiene la funcionalidad de descargar ecuaciones de un repositorio central. Dado que para el nuevo sistema, se centraliza la base de datos en el servidor web, se decidió cambiar esta funcionalidad por la siguiente: el alumno puede descargar ecuaciones del conjunto global de ecuaciones, eligiendo un rango de fechas cuyas posibilidades son: *Todas las fechas, Creadas desde ayer, Creadas hace una semana*

y *Creadas hace un mes*. Además debe elegir la cantidad de ecuaciones que desea descargar: *1,5,10,15* o *20*.

2.3.6. Eliminación de funcionalidad: Subir ecuaciones

Dado que en el nuevo sistema existe un conjunto global de ecuaciones al que se agregan automáticamente luego de ser creadas desde la aplicación web, y tomando en cuenta el tamaño del proyecto en cuanto a trabajo a realizar, se decidió eliminar la funcionalidad de que el alumno pueda subir ecuaciones. El alumno puede crear ecuaciones en su aplicación de escritorio para trabajar con ellas, pero no puede subirlas al conjunto global, sólo los docentes pueden hacerlo desde la aplicación web.

2.4. Requisitos no funcionales

2.4.1. Seguridad

Dado que la aplicación web tiene datos sensibles de los docentes, se definieron requisitos no funcionales de seguridad. Los mismos son:

- **Cifrado:** Todo mensaje entre el servidor y el cliente de la aplicación es encriptado utilizando el protocolo HTTPS[13] con certificado SSL/TLS[14].
- **Políticas de contraseñas:** Las contraseñas deben ser definidas aplicando reglas que obedecen a políticas de seguridad.

2.4.2. Usuarios concurrentes

El sistema debe soportar usuarios concurrentes.

2.4.3. Usabilidad

El sistema debe presentar una interfaz de usuario amigable, intuitiva y fácil de utilizar.

2.4.4. Coordinadores del sistema

El sistema debe contar con 4 usuarios coordinadores por defecto. Dos son usuarios coordinadores genéricos, y los otros dos son usuarios con los correos electrónicos de las profesoras Nora Ravaioli y Teresa Pérez.

2.4.5. Ingreso de ecuaciones

El ingreso de las ecuaciones se debe poder realizar con botonera la cual debe contar con las operaciones (raíz, potencias, cociente y paréntesis) y los espacios para escribir los operadores básicos (+, -, *) y números, así como la incógnita.

2.4.6. Ecuaciones iniciales

El sistema cuenta inicialmente con 40 ecuaciones cargadas, definidas y clasificadas por las profesoras Nora Ravaioli y Teresa Pérez. Además de ser accesibles a través de la aplicación web, las mismas se cargan inicialmente como ecuaciones por defecto en la aplicación de escritorio; es decir que se trata de un conjunto de ecuaciones iniciales para ambos sistemas.

2.4.7. Idioma

Los textos de la aplicación web deben estar en idioma Español.

2.4.8. Material de capacitación

Se debe diseñar material con las características generales del producto y ejemplos de uso. Se debe crear un video-tutorial y un manual escrito, explicando cómo utilizar las funcionalidades principales del sistema. Dicho material debe estar accesible en el inicio de la aplicación web.

2.4.9. Guía de configuración e instalación

Se debe diseñar un instructivo de cómo instalar el servidor web en el servidor central.

2.4.10. Navegadores

La aplicación web debe funcionar correctamente en los navegadores Google Chrome, Firefox y Safari.

2.4.11. Instalador de aplicación de escritorio para Windows

La aplicación de escritorio debe poderse ejecutar en computadoras con sistema operativo Windows. Para esto se debe generar un nuevo instalador.

2.4.12. Ecuaciones posibles

El conjunto de ecuaciones que se pueden crear en la aplicación web debe ser el mismo que el de la aplicación de escritorio.

3. Arquitectura y diseño

En esta sección se presenta la arquitectura y diseño del sistema, contemplando tanto a la aplicación web como a la aplicación de escritorio y la comunicación entre las mismas.

3.1. Vista lógica y física del sistema

A continuación se muestra un diagrama que representa al sistema enfocado desde una vista tanto lógica como física. Se intenta con el mismo representar la comunicación entre los sistemas y la estructura sobre la que se lleva a cabo la solución.

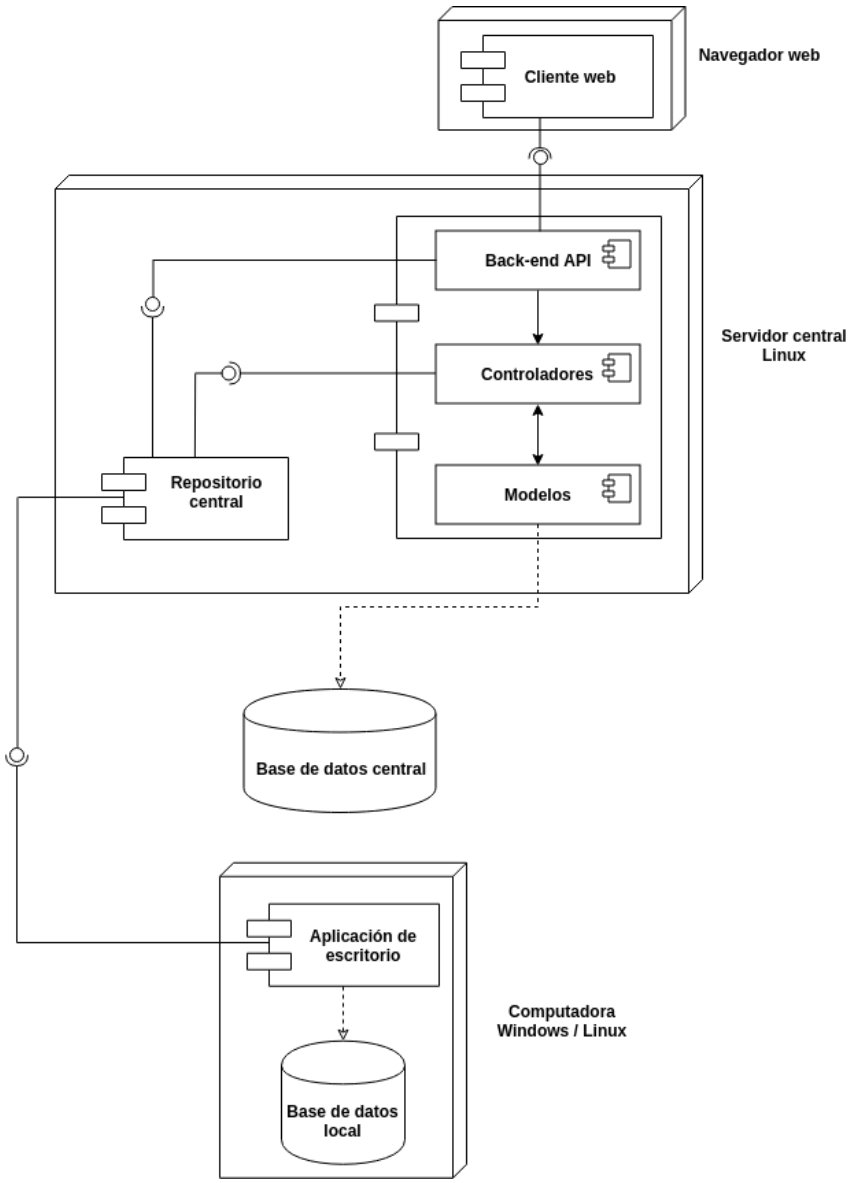


Figura 8: Vista lógica y física del sistema.

Se explican en las siguientes secciones, los componentes del diagrama y las comunicaciones entre los mismos.

3.1.1. Cliente web

Representa el front-end de la aplicación web y consume los servicios brindados por la API-REST del back-end. Se ejecuta en un navegador y es el encargado de implementar la interfaz gráfica con la cual interactúan los docentes y coordinadores.

3.1.2. Back-end API

Una API-REST es un estándar lógico y eficiente para la creación de servicios web. Se utiliza comúnmente como una interfaz para comunicar dos componentes de software a través de HTTP/HTTPS. La back-end API se ejecuta en el servidor central con sistema operativo Linux y es una API-REST que expone los servicios proporcionados por el back-end de la aplicación web.

3.1.3. Comunicación entre cliente web y back-end API

El cliente web consume los servicios que expone la back-end API.

3.1.4. Controladores

Los controladores residen en el servidor central y son los componentes que manejan la lógica de la aplicación web. Se encargan de interactuar con los modelos y atender las solicitudes provenientes de los clientes web brindándoles una respuesta. En caso de ser necesario comunicarse con servicios externos o con el repositorio central de la aplicación de escritorio, son los controladores quienes realizan dicha acción. Se mantienen controladores para los usuarios, ecuaciones, paquetes, sesiones, alumnos y grupos.

3.1.5. Modelos

Los modelos residen en el servidor central y son clases que representan tablas de la base de datos. Se encargan de brindar todas las operaciones necesarias para el acceso y modificación de los datos que se encuentran en la base de datos central.

3.1.6. Comunicación entre modelos y controladores

Los controladores utilizan a los modelos para llevar a cabo las operaciones relacionadas a la base de datos central.

3.1.7. Repositorio central

El repositorio central es un servidor que provee servicios para crear y descargar ecuaciones a y desde la base de datos central, así como también para sincronizar el avance de un alumno en un grupo.

3.1.8. Base de datos central

Contiene toda la información global del sistema, es decir los docentes, sus grupos, las ecuaciones, las clasificaciones, los paquetes de ecuaciones, los alumnos registrados en los grupos junto a su avance y las relaciones necesarias entre las diferentes entidades.

3.1.9. Comunicación entre los modelos y la base de datos central

Los modelos son los únicos componentes del sistema que ejecutan las operaciones sobre la base de datos central. Dichas operaciones son las de lectura, creación, eliminación y modificación.

3.1.10. Comunicación entre la back-end API y el repositorio central

El repositorio central consume servicios de la back-end API concernientes a la sincronización de información de alumnos y a la descarga de ecuaciones.

3.1.11. Comunicación entre los controladores y el repositorio central

Los controladores consumen servicios del repositorio central concernientes a la validación y creación de ecuaciones.

3.1.12. Aplicación de escritorio

Esta aplicación se ejecuta en la computadora de cada alumno, y además de hacer todo lo correspondiente al proyecto de grado de Camila Rojí y Martín Poli, debe poder asociar un alumno a la aplicación, permitiéndole sincronizar su avance, registrarse en grupos y descargar ecuaciones.

3.1.13. Comunicación entre aplicación de escritorio y repositorio central

La aplicación de escritorio consume servicios del repositorio central concernientes a la subida del avance del alumno en un grupo y a la descarga de ecuaciones.

3.1.14. Base de datos local

La base de datos local se encuentra en la computadora del alumno, y en ella se guarda la asociación del alumno a la aplicación, las ecuaciones creadas y descargadas, los estados de las resoluciones, los grupos en los que se registra, y las asociaciones entre las ecuaciones y los grupos.

3.1.15. Comunicación entre aplicación de escritorio y base de datos local

La aplicación de escritorio se comunica con la base para poder llevar a cabo las funcionalidades nuevas y las del proyecto anterior.

3.1.16. Tecnologías utilizadas

A continuación se enumeran las tecnologías que se utilizan en las distintas partes de la arquitectura presentadas en el diagrama anterior:

- Angular 5 en el cliente web.
- Ruby on rails en la back-end API.
- PostgreSQL en la base de datos central.
- Java en el repositorio central y en la aplicación de escritorio.
- SQLite en la base de datos local.

3.2. Diagrama de clases de diseño

En esta sección se presenta el diagrama de clases de diseño de la aplicación web. El mismo se muestra desde dos vistas: la vista de las operaciones que posee, y la vista de las interacciones. Se tienen los controladores y las entidades que conjuntamente implementan las funcionalidades que resuelven el problema planteado.

3.2.1. Vista de las operaciones

A continuación se presentan las clases y sus respectivas operaciones. El diagrama se muestra divido en 4 partes dado que mostrarlo en una sola figura no es posible por el tamaño del mismo:

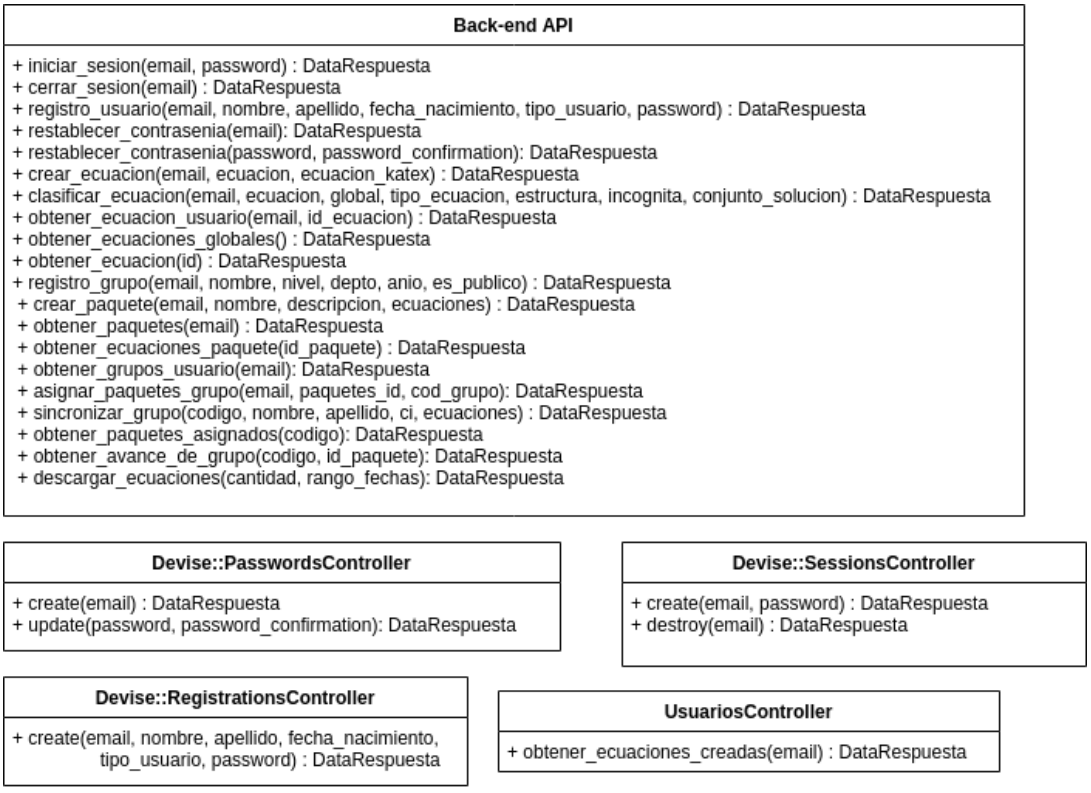


Figura 9: Vista de las operaciones parte 1.

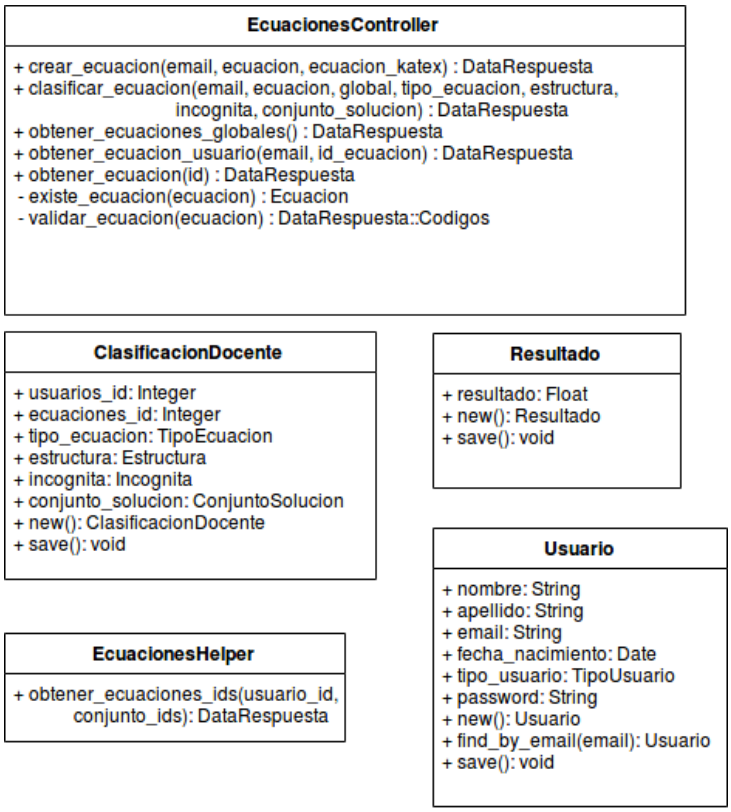


Figura 10: Vista de las operaciones parte 2.

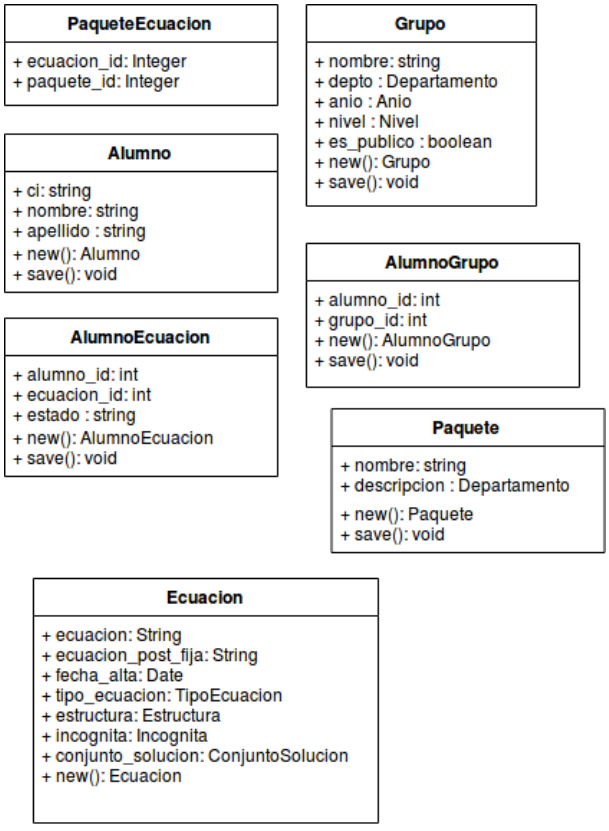


Figura 11: Vista de las operaciones parte 3.

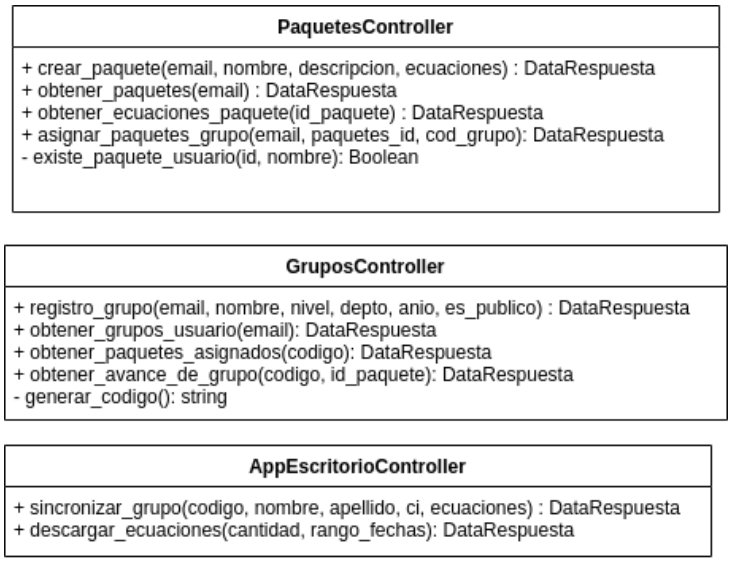


Figura 12: Vista de las operaciones parte 4.

3.2.2. DataTypes y enumerados

Como se puede apreciar en las figuras de la vista de las operaciones, todas las llamadas responden con un elemento del tipo *DataRespuesta*. Esta decisión de diseño se tomó para estandarizar la comunicación entre el cliente y el servidor y simplificar la implementación. El DataType *DataRespuesta* contiene tres campos:

- *success*: es verdadero si la operación se ejecutó con éxito. Falso en caso contrario.
- *errors*: si la operación contiene errores, se devuelve en este campo una lista con los mismos.
- *data*: si la operación devuelve información, se hace en este campo.

El mencionado campo *data*, puede tener varios tipos gracias a la propiedad de tipado dinámico de Ruby. En él se devuelve la información necesaria según la operación, y esta puede ser un conjunto de strings, un conjunto de DataTypes, o un tipo simple. A continuación se presentan los DataTypes y enumerados del sistema:

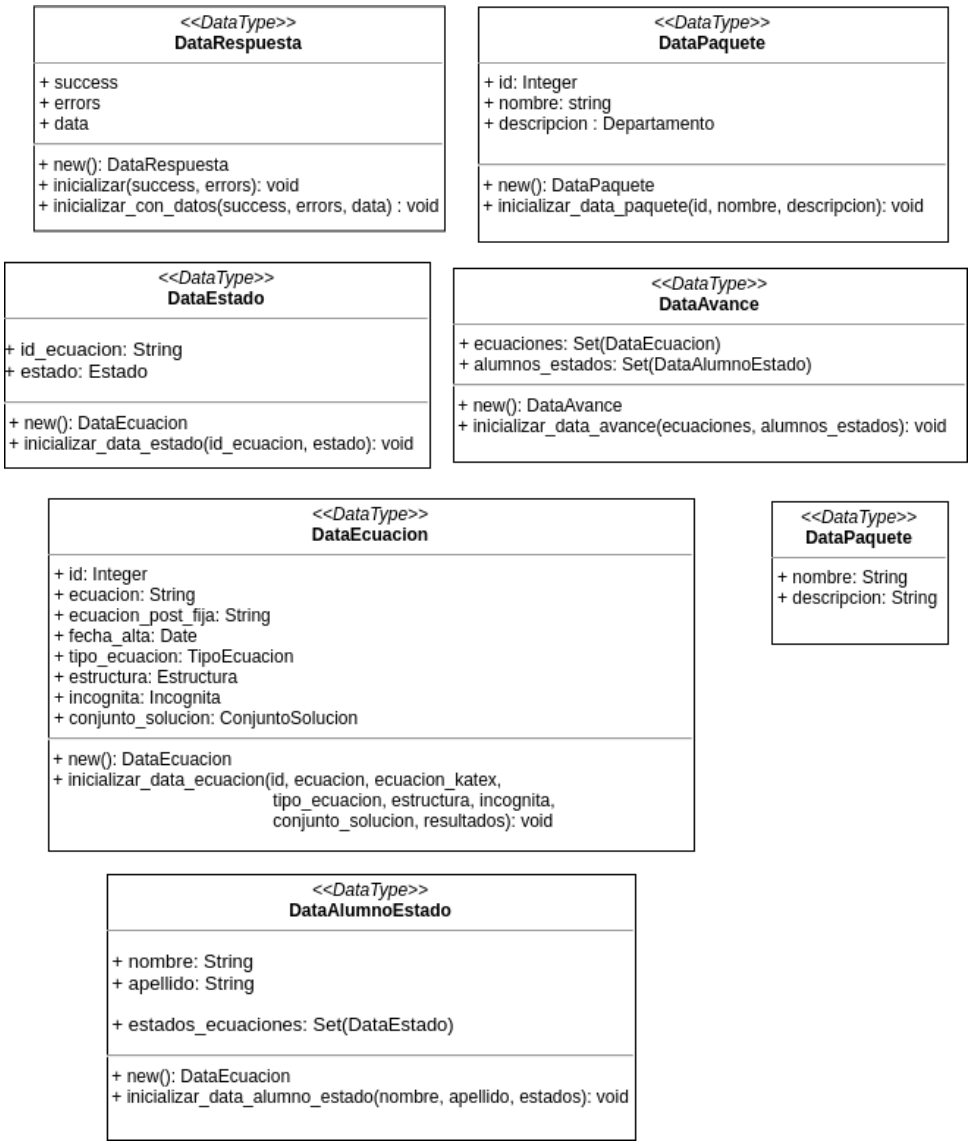


Figura 13: Datatypes del sistema.

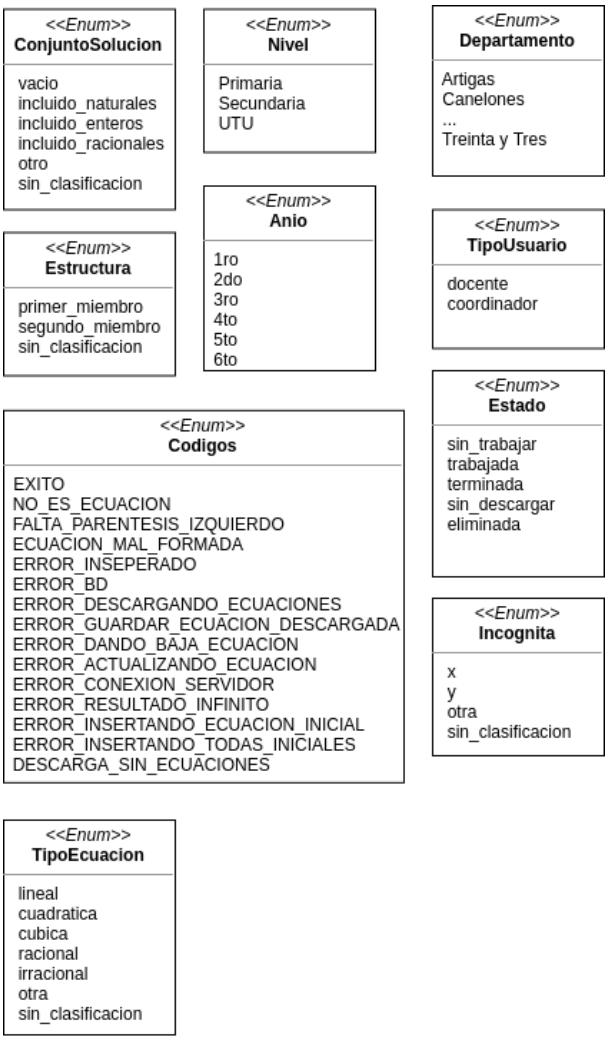


Figura 14: Enumerados del sistema.

3.2.3. Vista de las interacciones

A continuación se presentan las clases y las interacciones entre las mismas. El diagrama se muestra dividido en 3 partes dado que mostrarlo en una sola figura no es posible por el tamaño del mismo:

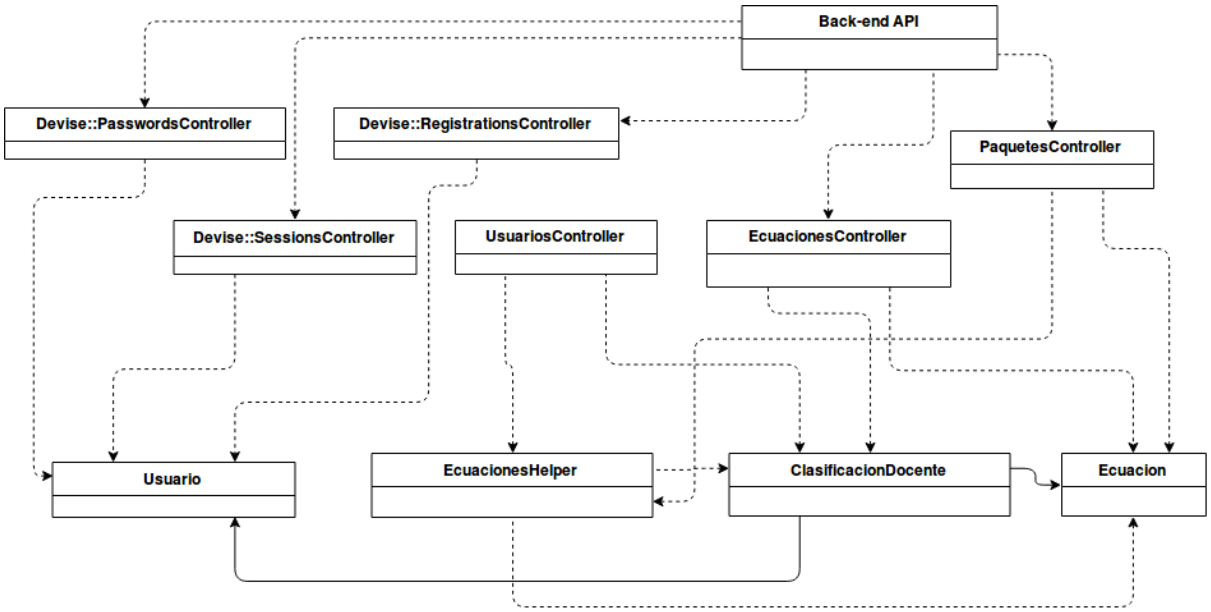


Figura 15: Vista de las interacciones parte 1.

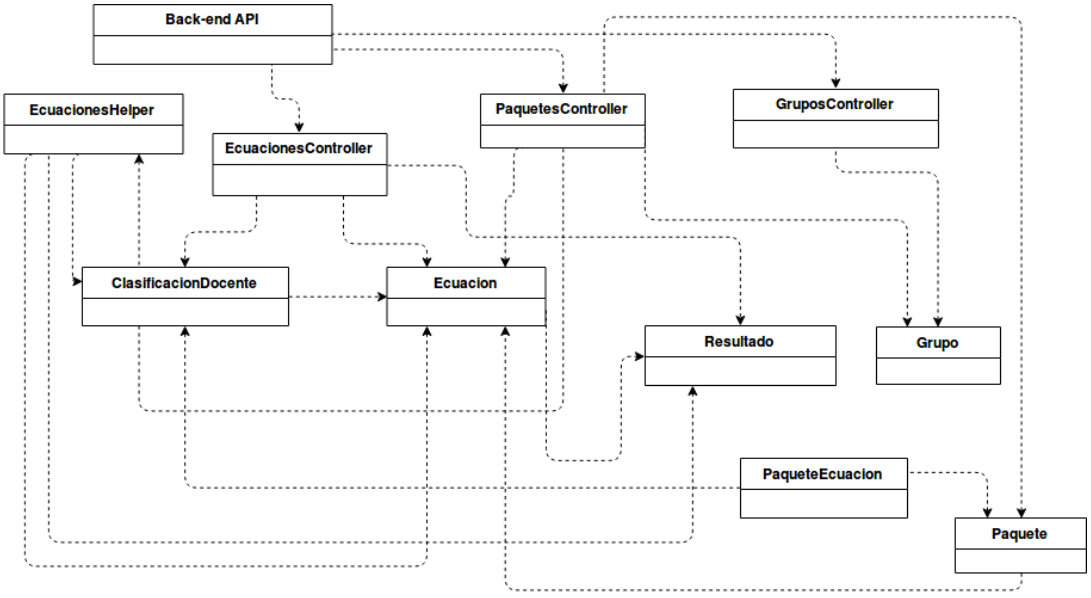


Figura 16: Vista de las interacciones parte 2.

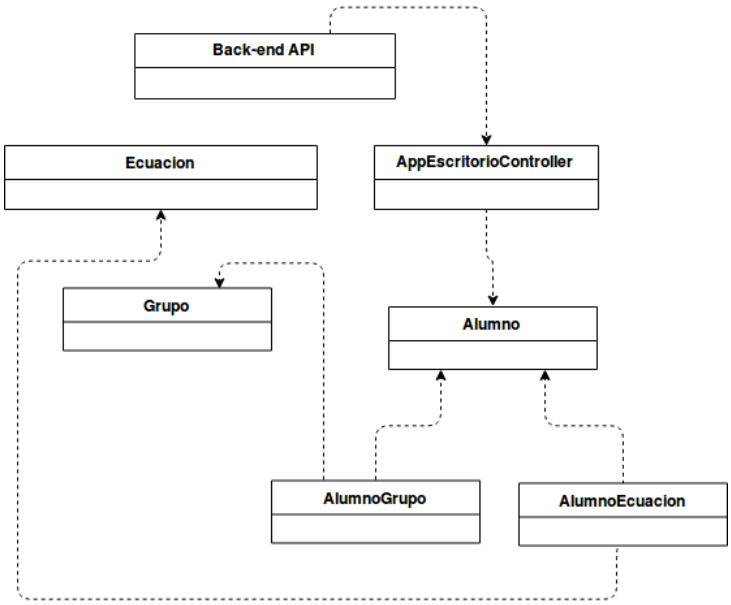


Figura 17: Vista de las interacciones parte 3.

Ambas vistas conforman al diagrama de clases de diseño de la aplicación web. Dicho diagrama muestra la estructura y las comunicaciones necesarias para la construcción de la solución al problema planteado. Las operaciones e interacciones se clarifican en la sección de implementación y algoritmos (sección 4).

3.3. Modelos de entidad-relación

En esta sección se presenta el diseño de la base de datos central. En el proyecto de Camila Rojí y Martín Poli, se creó un repositorio central con una base de datos conteniendo las ecuaciones y sus respectivas soluciones, subidas por alumnos usuarios de la aplicación de escritorio. Para el presente proyecto dicha base se extendió y se desacopló del repositorio central. Por lo tanto la misma cuenta con muchas más tablas necesarias para el nuevo sistema. En cuanto a la base de datos de la aplicación que se encuentra local en la computadora del alumno, se introdujeron cambios sobre la misma, para poder persistir la asociación del alumno a la aplicación, los grupos a los que se registra, y las ecuaciones asociadas a los distintos grupos (a través de los paquetes). A continuación se muestran los MER (Modelos de Entidad-Relación) de la base de datos central y la base de datos local.

3.3.1. MER global

El MER de la base de datos central se nombró como MER global, y se presenta a continuación:

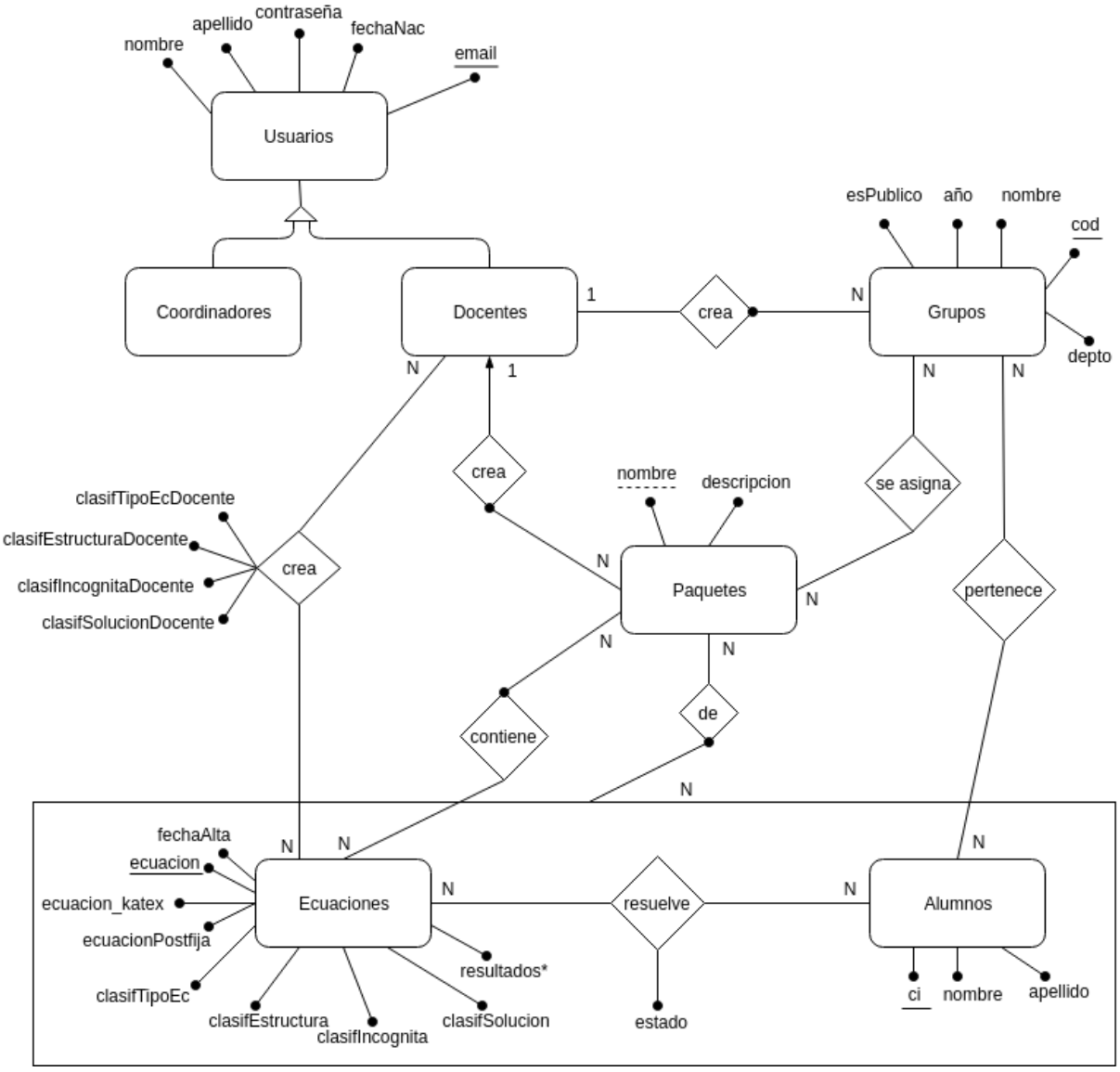


Figura 18: MER global.

Los docentes y coordinadores se encuentran en la tabla *Usuarios* identificados por su e-mail, y la forma que se utilizó para implementar la generalización es con un enumerado, que es "coordinador" si el usuario es coordinador, y "docente" en caso de que sea docente. Se eligió esta opción dado que resulta sencilla su implementación, los conjuntos de docentes y coordinadores son disjuntos, y la unión de ambos conjuntos contiene a todos los usuarios posibles de la aplicación web. Las tablas *Grupos* y *Paquetes* contienen a los grupos y paquetes de los docentes respectivamente, por lo que son entidades débiles relacionadas a éstos últimos. Cada grupo tiene un código único generado por el sistema, y los paquetes de un mismo docente no pueden repetir su nombre. Además interesa saber qué paquete está asignado a qué grupo.

Por otro lado, las ecuaciones del sistema se guardan en la tabla *Ecuaciones*, y cada vez que un docente crea una ecuación, se persiste la clasificación (para cada categoría: *tipo, incógnita, estructura, conjunto solución*) que asigna sobre la misma. Las ecuaciones son únicas en la base de datos central por su atributo *ecuacion*, y la relación de igualdad se define por: las operaciones, la incógnita, los números que posee, y el orden de los mismos.

En los siguientes puntos se explica con ejemplos lo mencionado previamente:

- La ecuación $\boldsymbol{x + 1 = 3}$ es distinta a la ecuación $\boldsymbol{1 + x = 3}$, a $\boldsymbol{3 = x + 1}$ y a $\boldsymbol{z + 1 = 3}$, por más que todas tengan las mismas operaciones, números y solución.
- Si el docente d_1 crea una ecuación, y luego el docente d_2 crea otra exactamente igual, el sistema también acepta esta última, pero no genera una nueva entrada en la tabla de la base de datos.

Si bien para cada par docente-ecuación (siendo la ecuación creada por el docente) se guarda la clasificación que dicho docente define, en la tabla *Ecuaciones* (Fig. 18) se tiene también una clasificación para cada ecuación definida por los usuarios coordinadores, que inicialmente se setea de forma automática con el valor *sin clasificar*. A continuación se explica con un ejemplo:

- El docente d_1 crea la ecuación e_1 con la clasificación:
  $c(d_1, e_1)$ = *(lineal, x, primer miembro,* $S \subset \mathbb{N}$*)*. Ésta clasificación existe para el docente y la ecuación creada.
- Automáticamente la ecuación e_1 se crea en la tabla *Ecuaciones* con una clasificación por defecto:
  $c(e_1)$ = *(S/C, S/C, S/C, S/C)*. Es decir, con todas las categorías definidas con el valor *sin clasificar*.
- El docente d_2 crea también la ecuación e_1 definiendo su propia clasificación $c(d_2, e_1)$ que no tiene por qué coincidir ni con $c(d_1, e_1)$ ni $c(e_1)$.
- Un coordinador puede cambiar la clasificación de e_1 y se modifica $c(e_1)$ sin afectar a $c(d_1, e_1)$ ni a $c(d_2, e_1)$.

Para los alumnos, interesa persistir en qué grupos están registrados, y qué ecuaciones tienen asignadas. Para las ecuaciones que tienen asignadas, es importante llevar el estado de resolución de las mismas, que es lo que utiliza el sistema para mostrar al docente el avance de sus alumnos.

3.3.2. MER local

El MER de la base de datos que se encuentra en la computadora del alumno, tiene el nombre de MER local, y fue modificada para la expansión del sistema y para satisfacer los nuevos requisitos. A continuación se muestra el MER local:

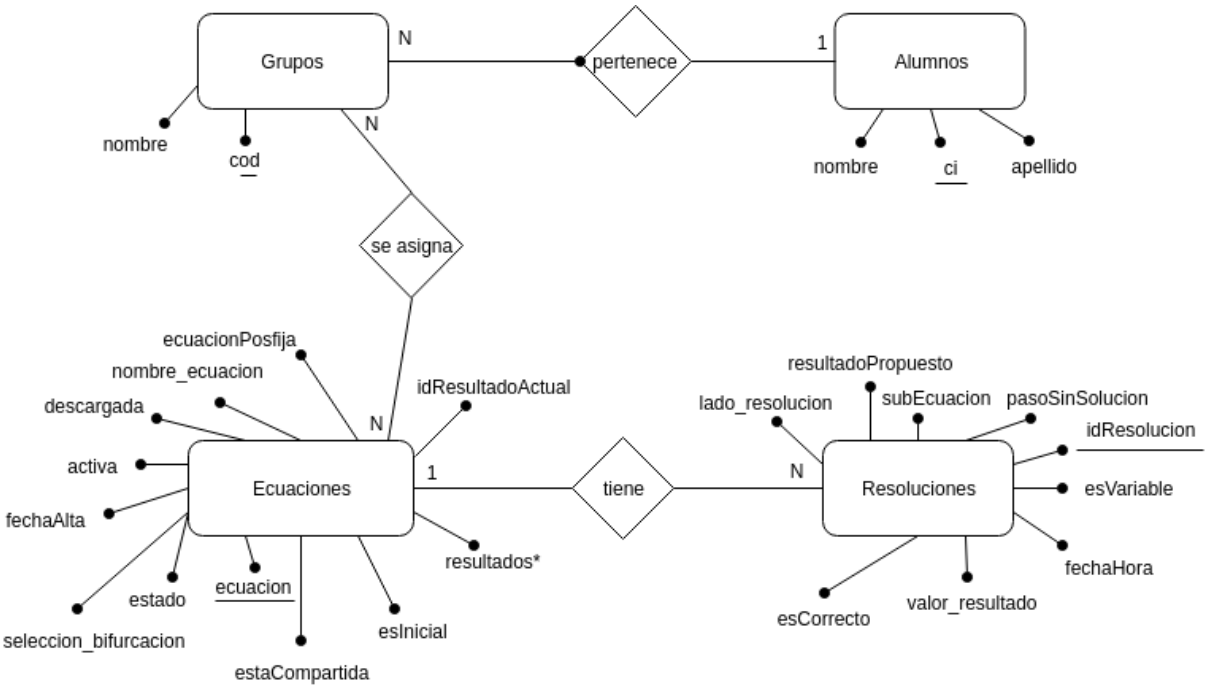


Figura 19: MER local.

Sólo un alumno puede estar asociado a la vez a la aplicación de escritorio, por lo que existe una sola instancia del mismo. Un alumno puede registrarse en varios grupos, almacenados en la tabla *Grupos*, que pueden ser de distintos docentes. La tabla *Ecuaciones* posee las ecuaciones que crea, descarga u obtiene al sincronizar el alumno y se mantiene su conjunto solución en la tabla *Resoluciones*. Además se mantiene el estado de resolución para cada una. Cuando se asocia un paquete a un grupo, y luego el alumno registrado en dicho grupo sincroniza la información, todas las ecuaciones del paquete se asocian al grupo en la aplicación de escritorio. Las tablas de las entidades fuertes, de las relaciones, las restricciones en lenguaje natural, y los atributos de ambos modelos, se explican en detalle en la sección Anexos, ítem Arquitectura y diseño.

4. Implementación y algoritmos

En esta sección se presentan los puntos centrales del desarrollo de la aplicación web, la aplicación de escritorio, y la comunicación entre las mismas. Se muestran los algoritmos centrales, y los aspectos principales de la implementación en los lenguajes utilizados.

La aplicación de escritorio no cambió las especificaciones de implementación con respecto al proyecto anterior. Sin embargo, se generó un nuevo instalador para computadoras con sistema operativo Windows con las herramientas Launch4j[15] e InnoSetup[16]. Con respecto a la nueva aplicación web, la misma se desarrolló para ser accedida desde los navegadores Google Chrome, Firefox y Safari.

4.1. Aspectos generales

A partir del documento de requisitos se construyó el documento de casos de uso que sirvió como punto de partida para el desarrollo de la implementación. Para dicha implementación se tomaron una serie de decisiones técnicas que se abordan en esta sección. La aplicación web se dividió conceptualmente en dos grandes partes:

- Capa de presentación: componente encargado de la interfaz gráfica y la interacción con el usuario, implementado en Angular 5.
- Capa lógica: componente encargado de procesar las solicitudes de la capa de presentación y retornar la salida correspondiente, implementado en Ruby on rails.

En la capa de presentación se tienen diferentes componentes, cada uno encargado de la interfaz de usuario concerniente a cada caso de uso. Cada componente es un conjunto de archivos de Angular, donde cada uno de ellos tiene diferentes funciones, que son:

- Interactuar con el usuario tomando la entrada y mostrando la salida.
- Incluir las librerías que sean necesarias.
- Exportar las funcionalidades que puedan necesitar otros componentes.
- Realizar las llamadas a la capa lógica.

Dichas llamadas se hacen a través de la implementación de servicios, que se mapean uno a uno con las funcionalidades que expone la back-end API. Los componentes entonces interactúan con el usuario y utilizan los servicios para comunicarse con el servidor.

Por otro lado, la capa lógica implementada en Ruby on rails define los puntos de entrada hacia las implementaciones de back-end en una API-REST. La misma sigue el patrón de arquitectura de software MVC, en donde se dejó sin efecto el componente de la Vista ya que es implementado por la capa de presentación en Angular 5. Las rutas de la API se definen en el archivo *routes.rb* apuntando cada una de ellas a una acción de un cierto controlador. Esto quiere decir que ante una solicitud, el ruteador de Ruby on rails dispara un evento que invoca la acción del controlador correspondiente a la solicitud y es éste quien se encarga de procesarla. Cuando un controlador recibe la solicitud, realiza las operaciones necesarias interactuando con los modelos y retorna un objeto del tipo *DataRespuesta* en formato JSON. Los elementos del back-end se agrupan en tres secciones principales:

- Controladores.
- Modelos.
- Data.

En Data se mantienen los distintos tipos de datos que son enviados hacia el front-end, mientras que en controladores y modelos se mantienen todos los archivos correspondientes a dichos componentes del patrón MVC. A continuación se explican las implementaciones de las soluciones de los casos de uso centrales del nuevo sistema.

4.2. Control de versiones

Para el control de versiones del código fuente se utiliza un repositorio Git en Bitbucket[17] en el cual se aloja tanto la API de back-end como el sitio de front-end. La rama principal es *master*, mientras que los casos de uso se desarrollan en *feature-branches* y una vez finalizados se integran a la rama master. Los cambios pequeños son incorporados directamente a la rama master. El flujo utilizado es el siguiente:

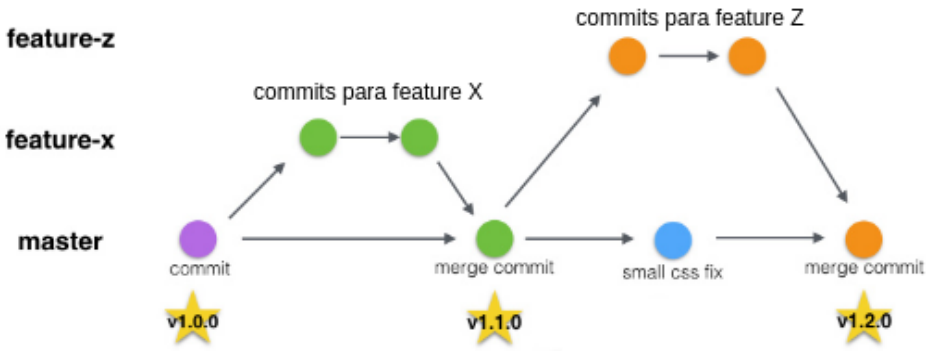


Figura 20: Flujo de Git elegido.

4.3. Coordinadores y ecuaciones por defecto

Para satisfacer las necesidades del sistema, la base de datos cuenta con información cargada desde el inicio. Dicha información refiere a cuatro usuarios coordinadores por defecto, dos de ellos asociados a las profesoras Nora Ravaioli y Teresa Pérez, y los otros dos genéricos. Por otro lado, se cuenta con cuarenta ecuaciones globales cargadas, con sus clasificaciones y resultados correspondientes. Estas ecuaciones son las mismas que se cargaron en la aplicación de escritorio en el proyecto de grado elaborado por Camila Rojí y Martin Poli.

4.4. Manejo de la sesión

Una de las tareas centrales al trabajar con usuarios en un sistema web es la de administrar la sesión. Para resolver este problema se utilizó autenticación basada en tokens, que se explica a continuación.

Un token es una cadena de caracteres que no tiene un significado en sí mismo, pero combinado con un correcto sistema de administración de tokens, es una pieza vital en la seguridad de una aplicación. La autenticación basada en tokens trabaja asegurándose de que cada solicitud al servidor llegue acompañada de un token firmado, el cual es verificado por el servidor para corroborar su autenticidad y retornar una respuesta.

En la implementación de back-end se utilizó la librería Devise[18] provista para Ruby on rails. Dado que se construyó una API y la librería está pensada para utilizarse en Ruby on rails sin restricciones (es decir, en Ruby on rails completo con vistas incluidas, y no únicamente como API), se procedió a utilizar la librería DeviseTokenAuth[19] basada en Devise. Cuando un usuario envía sus credenciales para iniciar sesión desde el cliente web, si éstas son correctas, DeviseTokenAuth crea un JSON web token (JWT token), lo almacena en la base de datos y lo envía al cliente. El cliente debe realizar las solicitudes subsecuentes con dicho token en los *Headers* de las mismas para que el servidor autentique al usuario comparando el token recibido con el almacenado.

En el front-end se utilizó la librería Angular2Token[20] que se encarga de administrar el token de la sesión y verificar ante cada acción del usuario que el token almacenado del lado del cliente sea válido invocando a la función *validate_token* del back-end (más específicamente, de DeviseTokenAuth). A su vez, en cada solicitud al servidor, Angular2Token envía en los *Headers* los datos correspondientes al usuario y la sesión. Con esto el equipo se asegura el manejo de la sesión del lado del cliente.

A continuación se muestra un diagrama de interacción entre el front-end y el back-end para clarificar el manejo de la sesión.

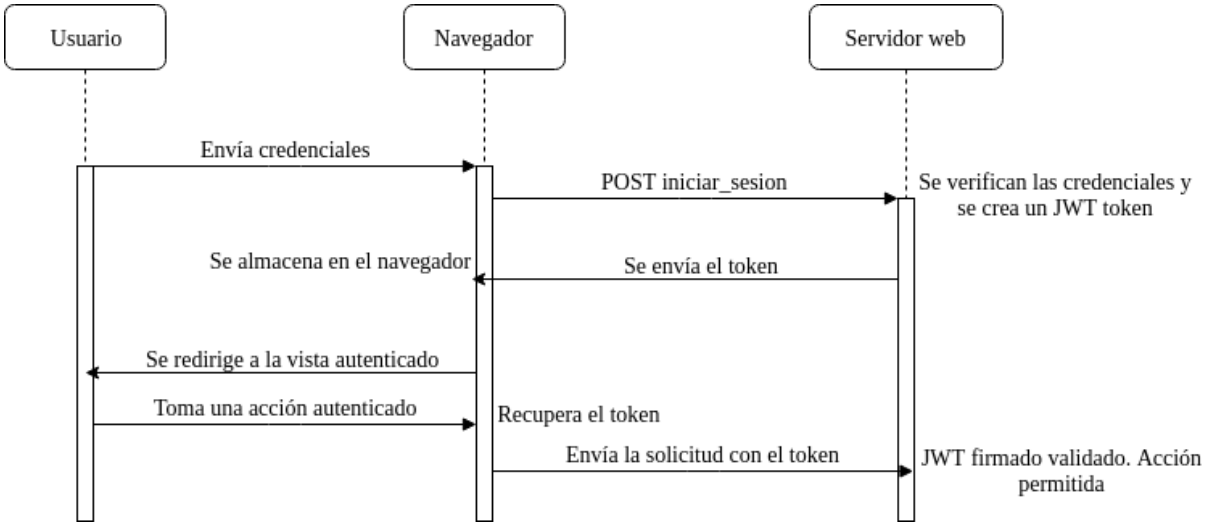


Figura 21: Flujo de autenticación.

4.5. Recuperación de contraseña

Una característica importante de todo sistema con acceso mediante credenciales es contar con la posibilidad de recuperar la contraseña. Esta funcionalidad se realizó con la ayuda de la librería Devise mencionada previamente, y la plataforma SendGrid[21] para el envío de e-mails.

En el front-end el usuario solicita la recuperación de contraseña ingresando su e-mail. El sistema, mediante Devise, genera automáticamente un token el cual es enviado a través

de un correo electrónico a la dirección ingresada por el usuario. El mismo es enviado de manera instantánea por la plataforma SendGrid y luce de la siguiente manera:

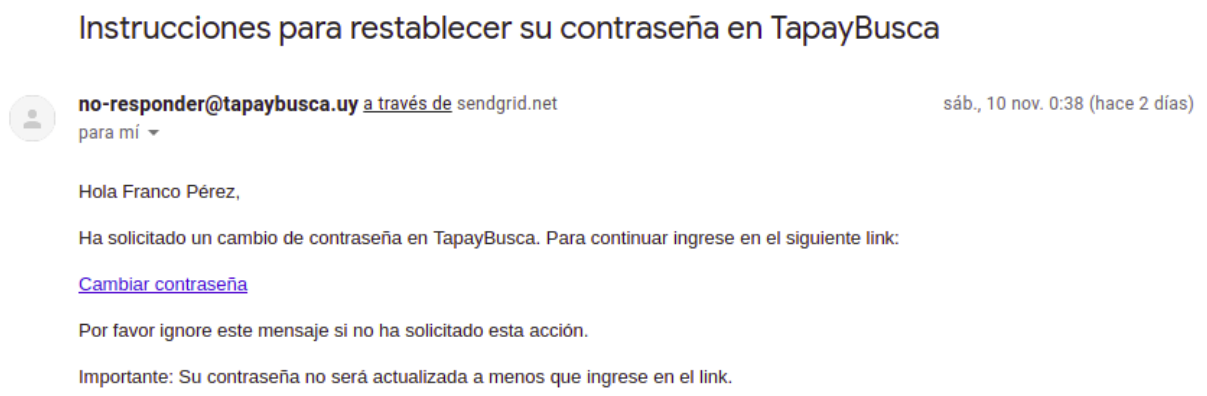


Figura 22: E-mail de recuperación de contraseña.

El usuario debe presionar en el link recibido para acceder a cambiar su contraseña. La estructura del link se puede apreciar en la siguiente imagen:

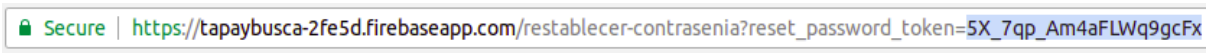


Figura 23: Link de recuperación de contraseña.

Una vez que el usuario accede al link (asociado a una vista del front-end de la aplicación), se valida el token contra el back-end y en caso de éxito puede proceder a ingresar la nueva contraseña. El flujo planteado se presenta a continuación:

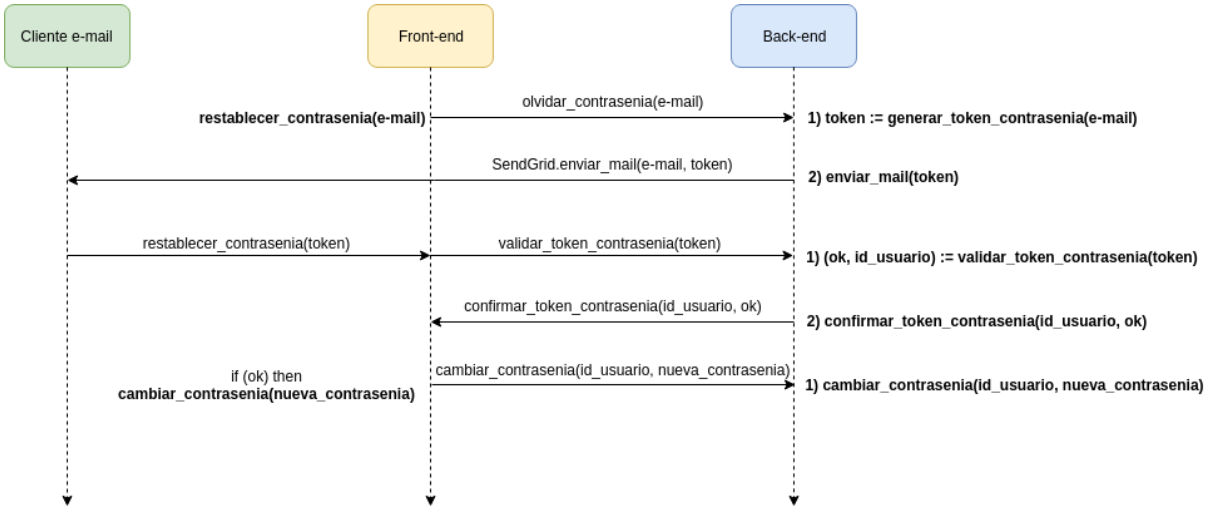


Figura 24: Flujo de recuperación de contraseña.

4.6. Creación de ecuación

En el proyecto realizado en 2017 por Martín Poli y Camila Rojí, los alumnos pueden crear ecuaciones desde la aplicación de escritorio. Esta funcionalidad fue implementada en el nuevo sistema web para ser accesible por los docentes de manera online.

4.6.1. Reutilización de componente de software

Para resolver la creación de la ecuación se evaluaron tres opciones:

- Desarrollar el algoritmo desde el principio en el nuevo sistema.
- Realizar el intento de traducir el código desarrollado en Java a código Ruby.
- Reutilizar el código Java desarrollado en la aplicación de escritorio.

La primera tiene la ventaja de independizarse del código generado en otro lenguaje y de un servicio externo, sin embargo lleva un gran esfuerzo dado que es un problema complejo. La segunda implica un esfuerzo importante debido a la magnitud del algoritmo y a la alta cantidad de líneas de código a traducir. Por último, la tercera es la más conveniente en términos de esfuerzo, reutilización y mitigación de errores. Por estos motivos se procedió a reutilizar el código escrito en Java por Martín Poli y Camila Rojí. Un aspecto a destacar era que el código debía ser accesible en un web service para poder ser consumido desde el servidor web. Para esto, se resolvió trasladar el código del algoritmo implementado en la aplicación de escritorio al web service del repositorio central, y de esta manera crear un nuevo punto de entrada que sirviera de generador y validador de nuevas ecuaciones. De esta manera, el repositorio central ha de tener dos nuevos servicios:

- $validarEcuacion(ecuacion) : Boolean$
  Toma una ecuación, e invoca el método desarrollado por Martín Poli y Camila Rojí, retornando verdadero si la ecuación es válida según dicho método, y falso en caso contrario.
- $altaEcuacion(ecuacion) : (ecuacion, resultados, ecuacionPostFija)$
  Toma la ecuación, e invoca el método desarrollado por Martín Poli y Camila Rojí, que la resuelve y construye su notación postfija (notación que es útil para la aplicación de escritorio).

En definitiva, se resolvió reutilizar componentes de software de la aplicación de escritorio, para resolver la creación de ecuaciones en el servidor central. A continuación se presenta un diagrama, el cual muestra las interacciones con el componente reutilizado:

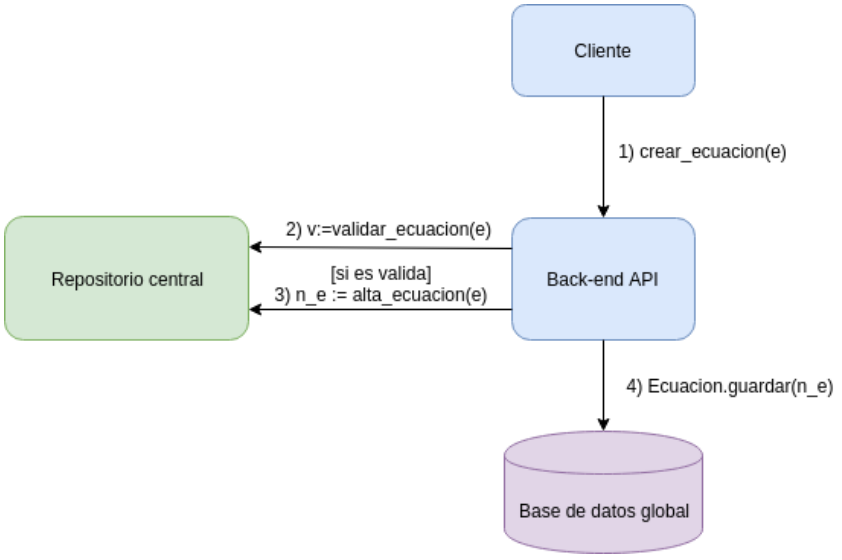


Figura 25: Diagrama de crear ecuación.

Se puede ver que el cliente le solicita la creación de una ecuación a la back-end API, y ésta invoca las dos operaciones mencionadas del componente reutilizado. Si no hay errores, se guarda la nueva ecuación en la base de datos global.

4.6.2. Construcción de la ecuación

Un aspecto importante a resolver era construir las ecuaciones desde el front-end de manera sencilla, amigable y en notación matemática. Para esto se utilizó la librería JavaScript ngKatex[10]. En el caso de uso de crear ecuación, la construcción mantiene 3 versiones diferentes de la misma ecuación:

- Versión ngKatex: versión de la ecuación que se utiliza para mostrarla en lenguaje matemático con la librería ngKatex.
- Versión amigable: string que ingresa el docente en una notación definida por el equipo, basada en MathPapa, que se considera intuitiva, cómoda y fácil de utilizar.
- Versión inFija: versión de la ecuación que se envía a la back-end API, y contiene el formato requerido por las operaciones del repositorio central.

Para la versión amigable, el docente puede escribir ambos miembros de la ecuación con los siguientes símbolos (además de números y variables):

- Suma: $+$
- Resta: $-$
- División: **{ ... / ... }**
- Multiplicación: $*$
- Potencias: **^ 2, ^ 3, ^ {1/2}, ^ {1/2}, ^ {1/3}, ^ {-1}, ^ {-2}, ^ {-3}**
- Raíces: **raiz2{ ... }, raiz3{ ... }**
- Paréntesis: **(...)**

En el front-end, a medida que el docente escribe la ecuación, se transforma y genera automáticamente la expresión matemática de la ecuación en sintaxis Tex[22] con la ayuda de la librería ngKatex para que el docente la pueda visualizar en lenguaje matemático.

Esta transformación se da de la siguiente manera:

- **raiz2** se reemplaza por **\sqrt**
- **raiz3** se reemplaza por **\sqrt[3]**
- / se reemplaza por **\over**

Una vez reemplazados estos símbolos ngKatex es capaz de representar la ecuación en sintaxis Tex. Es definitiva, cada vez que se produce un cambio en la entrada de la ecuación, se genera nuevamente la versión ngKatex de la misma, para mostrarla correctamente en cada momento.

Luego que el docente confirma la ecuación deseada se realiza una nueva transformación para obtener la ecuación infija en un formato que el método *validarEcuacion* del repositorio central pueda interpretar y validar de manera adecuada. En caso de ser una ecuación válida para el repositorio central, se invoca al método *altaEcuacion* para generar la nueva ecuación a guardar en la base de datos junto con sus resultados.

A continuación se presentan los algoritmos que representan el proceso de creación y alta de una ecuación:

Algorithm 1 Crear ecuación en front-end

```
1: function CREAR_ECUACIÓN(ecuacion, e_katex)
2:     e_infija ← transformar_a_infija(ecuacion)
3:     resultado ← Backend.alta_ecuacion(e_infija,e_katex)   ▷ se invoca el servicio
4:     if resultado.success then
5:         return success(creacion_exitosa)
6:     else
7:         return error(ecuacion_invalida)
```

Algorithm 2 Alta ecuación en back-end

```
1: function ALTA_ECUACIÓN(e_infija, e_katex)          ▷ se invoca a validar_ecuacion
2:     valida ← RepositorioCentral.validar_ecuacion(e_infija)
3:     if valida then                                          ▷ ecuación válida
4:         e ← RepositorioCentral.alta_ecuacion(e_infija)
5:         crear_ecuacion_sin_clasificacion(e, e_katex)        ▷ se crea la ecuacion
6:         return success(creacion_exitosa)
7:     else
8:         return error(ecuacion_invalida)
```

4.7. Registro y borrado de grupo

Para el registro de los grupos de un docente, se debió atacar el problema de cómo generar el código de los mismos. En el relevamiento de requisitos, lo discutido con las profesoras derivó en que el código no debía ser muy complejo ni largo, para disminuir la probabilidad de que los alumnos se equivoquen al registrarse a un grupo desde la aplicación de escritorio. Esto tiene el problema de acotar la cantidad de códigos posibles, es decir de grupos posibles a registrar en el sistema, ya que cada código es único para cada grupo. El resultado intermedio fue construir códigos de la siguiente forma:

- Largo fijo de 6 caracteres.
- Los 3 primeros caracteres son letras mayúsculas.
- Los 3 caracteres siguientes son dígitos.

Un ejemplo de código de grupo sería **ZYY431**, y la cantidad de posibles grupos es: $26^3.10^3$ ya que son 26 letras del alfabeto inglés y 10 dígitos posibles. Esto da un resultado de 17576000 grupos posibles. Esta solución se considera buena porque cumple con una cantidad amplia de códigos posibles, y además no son códigos complejos para los alumnos.

4.7.1. Registro desde la aplicación web

Al crear un grupo, el sistema genera automáticamente un código y se lo asigna al mismo, asegurándose de que no exista otro grupo con ese nuevo código. Para generarlo se utiliza una función de Ruby llamada *sample*, que genera pseudo-aleatoriamente un caracter dentro de un rango especificado. El algoritmo de generación de código se presenta a continuación:

Algorithm 3 Generar código

```
1: function GENERAR_CODIGO(())                    ▷ Genera un código para un grupo
2:     c1 ← sample(A..Z)                  ▷ sample genera un caracter de forma aleatoria
3:     c2 ← sample(A..Z)
4:     c3 ← sample(A..Z)
5:     c4 ← sample(0.,9)
6:     c5 ← sample(0.,9)
7:     c6 ← sample(0.,9)
8:     return concat(c1, c2, c3, c4, c5, c6)   ▷ retorna la concatenación de los caracteres
   obtenidos
```

Para que el sistema no registre dos grupos con el mismo código, se tiene el siguiente algoritmo implementado en el controlador de grupos, que utiliza la generación de códigos:

Algorithm 4 Crear grupo

```
1: function CREAR_GRUPO(nombre, nivel, depto, anio, id_usuario)
2:     if existe_grupo(nombre, id_usuario) then      ▷ no se puede repetir el nombre
3:         return error(repite_nombre)
4:     else
5:         nuevo_cod ← GENERAR_CODIGO()
6:         while existe(nuevo_codigo) do          ▷ si ya existe el código, genera otro
7:             nuevo_cod ← GENERAR_CODIGO()
8:         Grupo.crear(nombre, nivel, depto, anio, id_usuario, nuevo_codigo)
9:         return success(creacion_exitosa)
```

Con este algoritmo el sistema se asegura no tener dos grupos diferentes con el mismo código. Se tiene el problema de que luego de un número muy alto de grupos registrados, la creación pueda demorar ampliamente, y empeora si se llega al tope de grupos posibles, generando un loop infinito. En el presente proyecto no se ataca este problema porque en un principio no se espera que el número de grupos creados sea tan alto, y además la función de eliminar grupos va liberando códigos, aumentando la cantidad disponible. De todas formas, si se diera el problema, una solución muy simple es aumentar el tamaño del código.

4.7.2. Registro desde la aplicación de escritorio

El alumno en la aplicación de escritorio puede registrarse a un grupo. Para ello el docente previamente debe haber creado el grupo, y debe saber el código generado para el mismo. El docente debe darle a sus alumnos el código para que éstos se registren en la aplicación de escritorio. El alumno debe estar previamente asociado a la aplicación,

y al hacerlo se genera en la base de datos local una nueva fila con el código del grupo. En ese entonces, mientras el alumno no sincronice la información de ese grupo, en la base de datos local sólo se conoce el código del grupo y no su nombre. Por cuestiones de simplicidad, el registro de grupos desde la aplicación de escritorio no repercute en la aplicación web. Es decir que el alumno puede incluso registrarse en un grupo con un código que ni siquiera existe. Lo que sucede en ese caso se explica en la sección 4.8 (Sincronización de información de alumno).

4.7.3. Borrado desde la aplicación web

La eliminación del grupo desde la aplicación web, implica destruirlo en el servidor central. Por lo tanto se debe tener en cuenta las relaciones del mismo (ver Figura 18), de manera de no dejar información inconsistente luego del borrado. De las mismas surge el algoritmo siguiente:

Algorithm 5 Eliminar grupo

```
procedure ELIMINAR_GRUPO(codigo)                ▷ Elimina el grupo del sistema
    p_g ← PaqueteGrupo(codigo)                  ▷ obtiene la relacion de paquetes-grupos
    for i in p_g do
        PaqueteGrupo.eliminar(i)
    a_g ← AlumnoGrupo(codigo)                   ▷ obtiene la relacion de alumnos-grupos
    for i in a_g do
        AlumnoGrupo.eliminar(i)
    Grupo.eliminar(codigo)                      ▷ elimina el grupo
```

Para la implementación de este algoritmo, se hace uso de las asociaciones de Active Record[23]. Active Record es un patrón de arquitectura utilizado en Ruby on rails que implementa la persistencia de objetos en el mapeo objeto-relacional (es decir, se encarga de los modelos de Ruby on rails). Las asociaciones que permite definir este patrón entre los distintos modelos se encargan de la consistencia del sistema luego de eliminar una instancia determinada. Por lo tanto al configurar correctamente el modelo de Grupo, definiendo que las relaciones con los paquetes y con los alumnos deben destruirse al destruir el grupo, el algoritmo presentado previamente es ejecutado de forma automática por Ruby on rails.

4.7.4. Borrado desde la aplicación de escritorio

En la aplicación de escritorio, el alumno asociado a la aplicación y registrado en uno o más grupos, puede des-registrarse de cualquiera de ellos. Además, si elige la opción de sincronizar información y el grupo está asociado a uno o más paquetes, en la aplicación de escritorio se relaciona cada ecuación de los paquetes con el grupo. Al borrar un grupo, dicha relación con las ecuaciones debe eliminarse también, por lo que se tiene un algoritmo similar al borrado desde la aplicación web:

Algorithm 6 Eliminar grupo

```
procedure ELIMINAR_GRUPO(codigo)          ▷ Elimina el grupo de la app de escritorio
    e_g ← EcuacionGrupo(codigo)           ▷ obtiene la relacion de ecuaciones-grupos
    for i in e_g do
        EcuacionGrupo.eliminar(i)
    Grupo.eliminar(codigo)                ▷ elimina el grupo
```

Por cuestiones de alcance y tamaño del trabajo, se definió que eliminar un grupo en la aplicación de escritorio no incide en la base de datos global. Sólo actúa en la base de datos local. Es decir que si un alumno está registrado en un grupo en la base de datos global y local, y luego se borra en la aplicación de escritorio, de todas formas para la aplicación web, el alumno sigue en el grupo y el docente puede ver el último avance registrado.

4.8. Sincronización de información de alumno

La implementación de la sincronización de información de alumno se llevó a cabo en la aplicación web y en la aplicación de escritorio, dado que es la funcionalidad que se encarga de comunicarlas. En esta sección se explica el algoritmo que implementa la solución del presente caso de uso, teniendo en cuenta los diferentes estados que pueden tener ambas aplicaciones a la hora de comunicarse. La ejecución de la sincronización comienza desde la aplicación de escritorio, teniendo al alumno asociado a la misma, y estando conectada a Internet. Si no hay problemas en la conexión, el resultado de este caso de uso repercute tanto en la computadora del alumno como en el servidor central. A continuación se muestra un diagrama con el flujo de la sincronización de información de alumno:

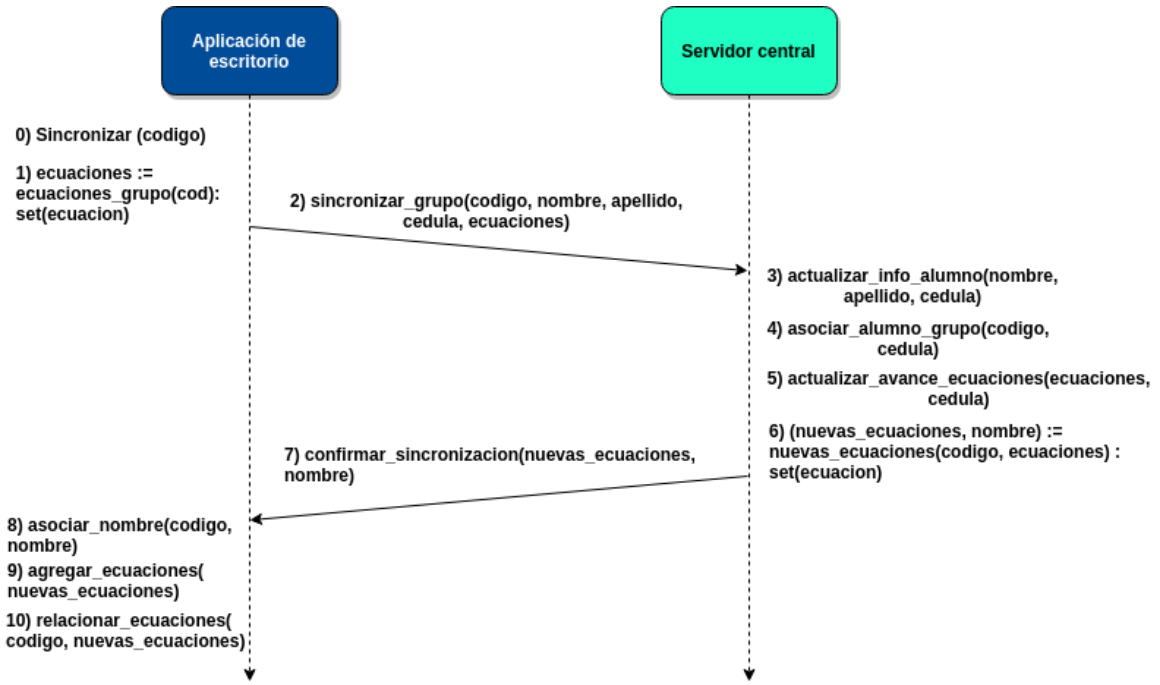


Figura 26: Flujo de sincronización de información de alumno.

En el diagrama presentado, el tiempo avanza hacia abajo como lo indican las flechas

discontinuas, mientras que en las continuas se representan los mensajes entre las aplicaciones. En la aplicación de escritorio, un alumno asociado a la misma y registrado en al menos un grupo, puede seleccionar la opción de sincronizar y confirmar eligiendo previamente el código del grupo deseado. En ese momento se está en el punto 0 del diagrama. A partir de este punto se explica la implementación paso a paso de la sincronización de información de alumno.

1. $ecuaciones := obtener\_ecuaciones(codigo) : set(ecuacion)$
   Lo primero que se hace en la solución es obtener las ecuaciones relacionadas a el grupo del *codigo* seleccionado, en la aplicación de escritorio. La interfaz gráfica no permite que se ingresen códigos no válidos, ni códigos de grupos no almacenados en la base de datos local. Esta función toma el código y obtiene las ecuaciones relacionadas al grupo. En caso de que el grupo no esté relacionado a ninguna ecuación en la base de datos local (por ejemplo si se está sincronizando al grupo por primera vez), simplemente se retorna el conjunto vacío. Es importante aclarar que en cada ecuación se envía el estado de resolución por parte del alumno.

2. $sincronizar\_grupo(codigo, nombre, apellido, cedula, ecuaciones)$
   Teniendo la computadora de escritorio conectada a Internet, la aplicación de escritorio invoca al repositorio central (ver Figura 8), y éste invoca a la back-end API enviando como parámetros el *nombre, apellido* y *cedula* del alumno asociado, el *codigo* del grupo seleccionado, y las *ecuaciones* relacionadas a éste último. Con dicha información, se invoca en la back-end API la operación de sincronizar grupo, cuyo algoritmo se presenta a continuación:

Algorithm 7 Sincronizar información de grupo

function SINCRONIZAR_GRUPO($codigo, nombre, apellido, cedula, ecuaciones$)
$actualizar\_info\_alumno(nombre, apellido, cedula)$
$asociar\_alumno\_grupo(codigo, cedula)$
$actualizar\_avance\_ecuaciones(cedula, ecuaciones)$
$(nuevas\_ecuaciones, nombre) \leftarrow nuevas\_ecuaciones(codigo, ecuaciones)$
return $(nuevas\_ecuaciones, nombre)$

   Una vez que la aplicación de escritorio envía el mensaje, se mide el tiempo. Si en 1 minuto no hay respuestas por parte del servidor central (ya sea porque se perdió lo enviado o se perdió la respuesta), se finaliza la sincronización y la aplicación le muestra un mensaje de error al alumno, solicitando que intente nuevamente. Las 4 siguientes operaciones explicadas, se ejecutan en el servidor central.

   Es importante aclarar que si el alumno intenta sincronizar la información de un grupo que registró en la aplicación de escritorio pero no existe en la base de datos global, entonces no se ejecutan las operaciones siguientes, y se devuelve un mensaje en la aplicación de escritorio indicando que el grupo no existe.

3. $actualizar\_info\_alumno(nombre, apellido, cedula)$
   Si el alumno nunca antes sincronizó información, se guardan sus datos en la base de datos global. Si ya existe el alumno con esa *cedula*, no se hace nada.

4. $asociar\_alumno\_grupo(codigo, cedula)$
   Si el alumno nunca antes sincronizó información para el grupo seleccionado, se

guarda en la base de datos global la relación entre el alumno y el grupo. Si ya estaban relacionados, no se hace nada.

5. *actualizar_avance_ecuaciones*(*cedula*, *ecuaciones*)
   Tomando las *ecuaciones* que envió la aplicación de escritorio (donde cada una posee el estado de resolución del alumno indicado con la *cedula*), se actualiza el estado de resolución de cada una de ellas. Se muestra este algoritmo a continuación:

Algorithm 8 Actualizar avance

```
procedure ACTUALIZAR_AVANCE(cedula, ecuaciones)
    for e in ecuaciones do
        estado ← estado(e)
        AlumnoEcuacion.actualizar_estado(estado, cedula)
```

Si el conjunto de ecuaciones es vacío, no se hace nada.

6. (*nuevas_ecuaciones*, *nombre*) ← *nuevas_ecuaciones*(*codigo*, *ecuaciones*)
   En esta función se retorna el nombre del grupo, y nuevas ecuaciones que hayan podido ser asignadas al grupo. A continuación se presenta el algoritmo:

Algorithm 9 Obtener nombre y nuevas ecuaciones

```
function NUEVAS_ECUACIONES(codigo, ecuaciones)
    nombre ← obtener_nombre(codigo)
    nuevas_ecuaciones = []                    ▷ Se inicializa el conjunto vacío
    paquetes ← PaqueteGrupo.paquetes(codigo)
    for p in paquetes do
        for ec in p do
            if not(ec ∈ ecuaciones) then
                nuevas_ecuaciones.agregar(ec)
    return (nuevas_ecuaciones, nombre)
```

Esta función retorna el nombre por la siguiente razón: si sincroniza por primera vez, la aplicación de escritorio toma el nombre retornado y lo asocia al código, dado que previamente sólo conoce el código. Por lo tanto el alumno podrá ver de ahí en más el nombre del grupo.

Por otro lado, las *nuevas_ecuaciones* son las ecuaciones que se asociaron al grupo desde la aplicación web (a través de paquetes) luego de su última sincronización, o antes de sincronizar por primera vez. En definitiva son las ecuaciones asociadas al grupo que en ese momento no pertenecen al conjunto del parámetro *ecuaciones*. A continuación se muestra un ejemplo para clarificar lo mencionado: El docente d_1 creó en la aplicación web el grupo g_1 con nombre n_1 al cuál le asignó un paquete que contiene las ecuaciones del sistema e_1, e_2 y e_3. Posteriormente, en la aplicación de escritorio, el alumno a_1 ya asociado a la misma, se registró en el grupo g_1 con su código, y seleccionó sincronizar su información en el mismo. Esto dió como resultado en *nombre* el valor n_1 y en *nuevas_ecuaciones* el conjunto con e_1, e_2 y e_3. Luego de la finalización de la sincronización del grupo, d_1 le asignó un nuevo paquete a g_1 que contiene las ecuaciones e_2 y e_5. Finalmente, el alumno seleccionó otra vez

sincronizar información de g_1 y en este caso se retornó en *nombre* el valor n_1 y en *nuevas_ecuaciones* el conjunto únicamente con e_5, ya que e_2 se encontraba como una ecuación previamente asignada.

7. $confirmar\_sincronizacion(nuevas\_ecuaciones, nombre)$
   Luego de las operaciones 3, 4, 5 y 6 de la back-end API (págs. 47 y 48), el repositorio central toma la respuesta de esta última y la reenvía a la aplicación de escritorio con el *nombre* y las *nuevas_ecuaciones* obtenidas. Es importante recordar que la aplicación de escritorio estaba esperando con un timeout en el punto 2 la respuesta de la aplicación web, manteniendo recordado el *codigo* del grupo en cuestión. Si dicha respuesta llega correctamente, la operación $confirmar\_sincronizacion$ ejecuta entonces tres operaciones para finalizar la sincronización, como se muestra a continuación:

Algorithm 10 Confirmar sincronización

procedure CONFIRMAR_SINCRONIZACION($nuevas\_ecuaciones, nombre$)
  $asociar\_nombre(codigo, nombre)$
  $agregar\_ecuaciones(nuevas\_ecuaciones)$
  $relacionar\_ecuaciones(codigo, nuevas\_ecuaciones)$

   Recordar que, si el mensaje de respuesta se pierde o llega luego de que se terminó el tiempo, la aplicación de escritorio devuelve un mensaje de error al alumno solicitando que intente nuevamente.

8. $asociar\_nombre(codigo, nombre)$
   Si se está sincronizando por primera vez, la aplicación de escritorio no conoce el nombre del grupo, por lo que en este caso le asocia dicho *nombre*. En otro caso no se hace nada.

9. $agregar\_ecuaciones(nuevas\_ecuaciones)$
   Se agregan a la base de datos local las nuevas ecuaciones en caso de que no existan. Para aquellas ecuaciones que ya existen en la base de datos local, ya sea por otras sincronizaciones, o por la descarga, no se hace nada.

10. $relacionar\_ecuaciones(codigo, nuevas\_ecuaciones)$
    Se asocian las *nuevas_ecuaciones* al grupo en cuestión a partir de su código y de las ecuaciones traídas desde el servidor.

Se puede afirmar que al sincronizar, en caso de que se pierda el mensaje de respuesta del servidor, puede quedar información inconsistente entre la aplicación web y la aplicación de escritorio. Por cuestiones de tiempo y de alcance se definió únicamente mostrar el mensaje de error mencionado previamente. Es importante aclarar que dada la naturaleza del algoritmo, si se genera alguna inconsistencia, esta última se soluciona cuando se ejecute correctamente una nueva sincronización. A continuación se presenta un diagrama que refleja el ejemplo del punto 6, para que se comprenda mejor la sincronización de información de alumno:

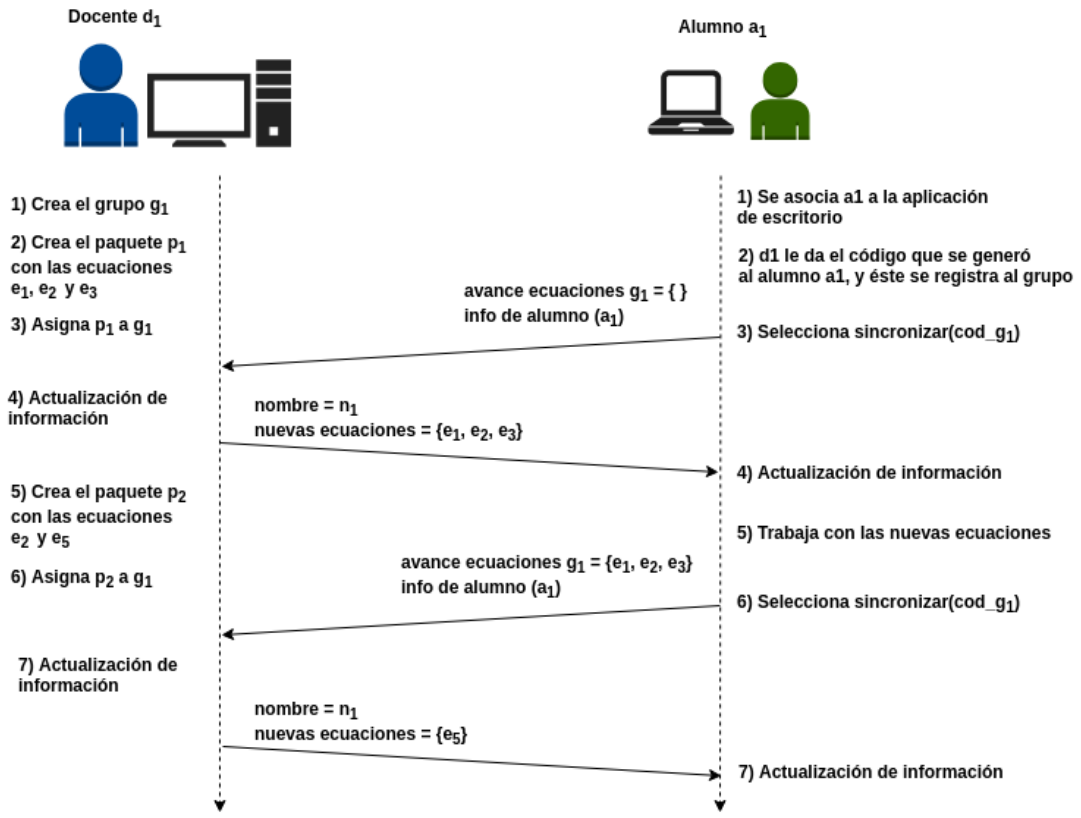


Figura 27: Ejemplo de sincronización.

4.9. Visualización de información en la aplicación web

En esta sección se explica cómo se visualiza la información desde la aplicación web, indicando los algoritmos y las herramientas involucradas. En general, cuando se muestran conjuntos de elementos, se utilizan tablas de Bootstrap. Si verticalmente la información a mostrar se excede de un tamaño definido, se utiliza la paginación. Si se excede horizontalmente, entonces se usan barras de desplazamiento. Se profundiza a continuación en la visualización del avance, y de las ecuaciones, que tiene aspectos particulares que se tuvieron que resolver.

4.9.1. Visualización del avance de los alumnos

Una funcionalidad principal de la aplicación web consiste en mostrar el avance de los alumnos en la resolución de las ecuaciones que les fueron asignadas a través de los paquetes. Para hacerla posible, se utilizó la directiva *ngFor* de Angular combinada con las tablas de Bootstrap. Dicha directiva permite desplegar de forma sencilla los elementos de un arreglo, y las tablas de Bootstrap se pueden generar dinámicamente, por lo que al combinarlas se pueden construir matrices dinámicas en HTML. Cuando un docente desea ver el avance de sus alumnos, debe elegir el grupo y el paquete asociado al mismo. Esto genera que desde el lado del servidor central, se invoque la operación de obtener avance de grupo, a partir del código del mismo y de un paquete que le fue asignado, cuyo algoritmo se describe a continuación:

Algorithm 11 Obtener avance de grupo

```
function OBTENER_AVANCE_GRUPO(codigo, id_paquete)
    a_g ← AlumnoGrupo.alumnos(codigo)          ▷ obtiene los alumnos del grupo
    e_p ← PaqueteEcuacion.ecuaciones(id_paquete)          ▷ obtiene las ecuaciones
    info_alumnos = []                                  ▷ inicializa en vacio
    for a in a_g do
        data_alumno.crear(a.nombre, a.apellido)          ▷ crea el data alumno
        for e in e_p do
            data_alumno.agregar(e.avance)
        info_alumnos.agregar(data_alumno)
    return (e_p, info_alumnos)
```

Esta función retorna dos conjuntos de datos: en *e_p* retorna las ecuaciones del paquete seleccionado, y en *info_alumnos*, un conjunto de *data_alumno* donde cada elemento del mismo contiene la información del alumno, y el avance de cada ecuación del paquete. Con estos dos conjuntos obtenidos, en el front-end se genera una matriz cuya primera columna corresponde a los nombres y apellidos de los alumnos, y el resto de las columnas corresponde a cada ecuación. En cuanto a las filas, cada fila corresponde a un alumno y al estado de su trabajo para cada ecuación. En definitiva, con dichos conjuntos se genera la matriz donde cada entrada **[i,j]** de la misma corresponde al estado de resolución de la ecuación **j** del paquete, realizada por el alumno **i** del grupo, como se describe en la siguiente imagen del front-end de la aplicación web:

Avance	$x+5=12$	$2(x+5)=12$	$\frac{150}{x+10}=30$	$25=x^2$	$(x+2)^2+10=26$	$\sqrt[3]{\frac{40}{x+1}}=2$	$20=84-x^3$
Roberto Medina	Sin descargar	Sin descargar	Trabajada	Sin descargar	Trabajada	Trabajada	Sin descargar
Selena Gutiérrez	Sin descargar	Sin descargar	Terminada	Sin descargar	Terminada	Terminada	Sin descargar
Joaquín Espino	Sin descargar	Sin descargar	Sin trabajar	Sin descargar	Terminada	Trabajada	Sin descargar
Betina Píriz	Sin descargar	Sin descargar	Sin trabajar	Sin descargar	Terminada	Terminada	Sin descargar
Alejandra Rosas	Sin descargar	Sin descargar	Terminada	Sin descargar	Trabajada	Sin trabajar	Sin descargar

Figura 28: Visualización de información de alumnos.

Como se puede ver en la parte inferior de la imagen, si la cantidad de ecuaciones supera el ancho de la pantalla, se genera automáticamente una barra de scroll, para poder moverse horizontalmente por la matriz y ver el avance de todas las ecuaciones del paquete. En cuanto a la cantidad de alumnos, el front-end pagina en grupos de a 10 alumnos, es decir que si hay más de 10 alumnos registrados en el grupo, se muestra una opción para paginar y mostrar la información de los siguientes. El docente puede

entonces visualizar claramente cómo es el avance de cada alumno del grupo, con respecto a las ecuaciones del paquete asignado.

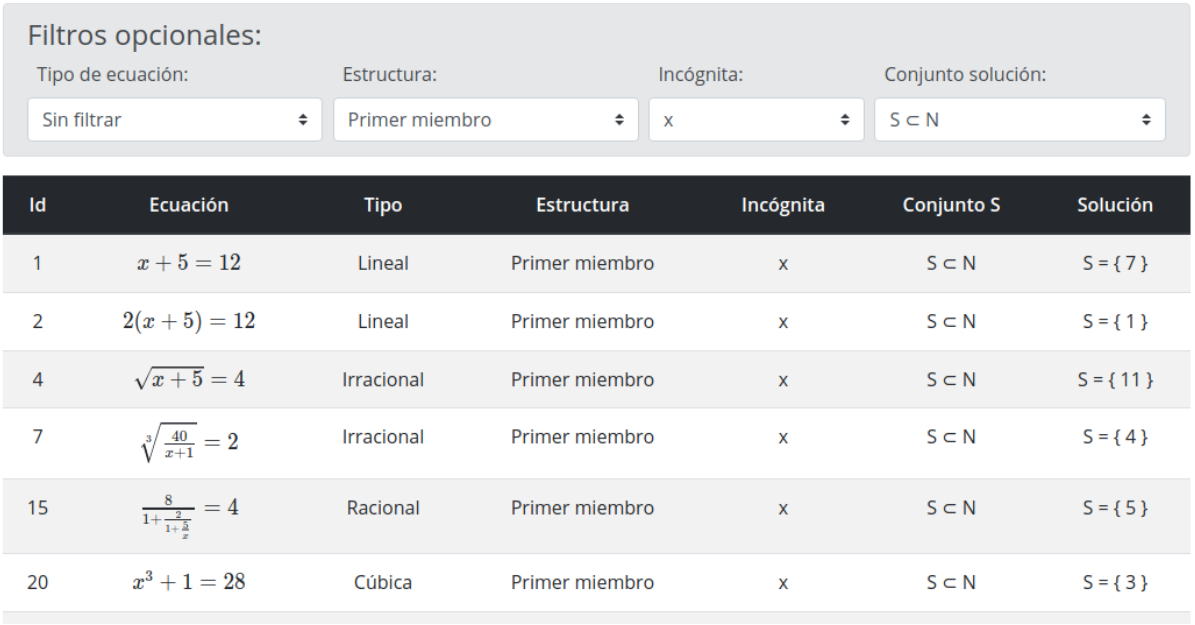
4.9.2. Visualización de ecuaciones

La clasificación de las ecuaciones tiene como principal utilidad asistir en la visualización de las mismas. Dado que al crear una ecuación se guarda automáticamente en el conjunto global de ecuaciones, accesible a todos los usuarios docentes de la aplicación web, al intentar ver dicho conjunto puede suceder que en cierto momento sea demasiado grande. Por lo tanto cuando se desee ver las ecuaciones, se puede aplicar filtros según las categorías definidas en los requisitos (ver Tabla 1). Dicho filtrado se realiza dinámicamente en el front-end, tanto en el caso de ecuaciones globales como el de ecuaciones creadas. Inicialmente la visualización no tiene filtros, es decir que todas las categorías tienen la opción *sin filtrar*. El algoritmo toma el conjunto de ecuaciones seleccionado y un valor para cada categoría. El mismo se presenta a continuación:

Algorithm 12 Filtrar ecuaciones

function FILTRAR_ECUACIONES($ecuaciones, tipo, estructura, incognita, conjunto\_sol$)
  if $tipo \neq sin\_filtrar$ **then**
    $ecuaciones \leftarrow filtrar(ecuaciones, tipo)$ ▷ filtra segun tipo
  if $estructura \neq sin\_filtrar$ **then**
    $ecuaciones \leftarrow filtrar(ecuaciones, estructura)$ ▷ filtra segun estructura
  if $incognita \neq sin\_filtrar$ **then**
    $ecuaciones \leftarrow filtrar(ecuaciones, incognita)$ ▷ filtra segun incognita
  if $conjunto\_sol \neq sin\_filtrar$ **then**
    $ecuaciones \leftarrow filtrar(ecuaciones, conjunto\_sol)$ ▷ filtra segun conj sol
  return ($ecuaciones$)

La función $filtrar$ retorna el conjunto de las ecuaciones que cumplen con el valor de la clasificación. Es decir que se aplica un filtrado en cascada, mostrando las ecuaciones que cumplen simultáneamente con los 4 filtros. A continuación se muestra una imagen de visualización de ecuaciones en la aplicación web, donde se eligen los filtros de: incógnita x, conjunto solución incluído en naturales, estructura de primer miembro, y sin filtrar según el tipo.



Id	Ecuación	Tipo	Estructura	Incógnita	Conjunto S	Solución
1	$x+5=12$	Lineal	Primer miembro	x	S ⊂ N	S = { 7 }
2	$2(x+5)=12$	Lineal	Primer miembro	x	S ⊂ N	S = { 1 }
4	$\sqrt{x+5}=4$	Irracional	Primer miembro	x	S ⊂ N	S = { 11 }
7	$\sqrt[3]{\frac{40}{x+1}}=2$	Irracional	Primer miembro	x	S ⊂ N	S = { 4 }
15	$\frac{8}{1+\frac{2}{1+\frac{3}{x}}}=4$	Racional	Primer miembro	x	S ⊂ N	S = { 5 }
20	$x^3+1=28$	Cúbica	Primer miembro	x	S ⊂ N	S = { 3 }

Figura 29: Ejemplo de filtrado en aplicación web.

4.10. Cifrado de la información entre el cliente y el servidor

Actualmente muchos de los sitios que no utilizan HTTPS como protocolo de transferencia están siendo "marcados" como no seguros, es decir, los usuarios que acceden a dichos sitios son notificados acerca de la inseguridad del mismo. Debido a esto y a que el sistema maneja información personal de los usuarios como lo son nombres, e-mails, contraseñas, fechas de nacimiento y cédulas de identidad de los alumnos, se decidió la utilización de este protocolo para la comunicación entre cliente y servidor web. Además, todas las solicitudes hacia el servidor son forzadas a redireccionarse por HTTPS en lugar de poder estar ambos protocolos disponibles, HTTP y HTTPS. Los pasos realizados son los descriptos a continuación:

En el back-end:

- Se editó un archivo de configuración de Ruby on rails para indicar el forzado de SSL, es decir, forzar el servidor a utilizar un certificado SSL/TLS.
- Se creó un certificado SSL para usar localmente desde la consola de comandos de Linux con el comando *openssl*, el cuál generó los archivos *localhost.key* y *localhost.crt*, clave y certificado de seguridad respectivamente.
- Se despliega el servidor escuchando en el puerto 443 (*puerto por defecto en servidores que utilizan SSL/TLS*) indicando la ruta de los archivos creados en el paso anterior.

Así queda cubierta la transferencia segura en el servidor web.

En el front-end:

- Se creó un certificado SSL para usar localmente desde la consola de comandos de Linux con el comando *openssl*, el cuál generó los archivos *localhost.key* y *localhost.crt*, clave y certificado de seguridad respectivamente.

- Se despliega la aplicación web habilitando SSL e indicando la ruta de los archivos creados en el paso anterior.

Con estos pasos se puede asegurar que la comunicación entre el cliente y el servidor web será cifrada. En la sección 5 (Testing) se incluyen las pruebas realizadas.

5. Testing

En esta sección se abordan los aspectos relevantes del testing del sistema. Según su definición, el *testing* es la investigación técnica de un producto con el objetivo de brindar información relativa a la calidad del software, a los diferentes actores involucrados en un proyecto. En este proyecto, el testing se realizó a través de las siguientes pruebas:

- Pruebas de back-end de la aplicación web automatizadas por cada caso de uso.
- Pruebas manuales desde el front-end de la aplicación web.
- Pruebas manuales desde el front-end de la aplicación de escritorio.
- Pruebas de usabilidad con usuarios finales.
- Pruebas de despliegue.
- Pruebas de concurrencia.
- Pruebas de seguridad con HTTPS y SSL/TLS.

5.1. Datos de prueba

Para la realización de testing tanto manual como automático, se definieron datos de prueba para poder generar pruebas significativas. Al tener datos de prueba creados, la ejecución de pruebas se hizo más sencilla y ahorró el tiempo de tener que cargarlos manualmente cada vez que era necesario. Se crearon datos de prueba sobre:

- Usuarios.
- Ecuaciones.
- Grupos.
- Paquetes.
- Alumnos.
- Relaciones entre los elementos de los puntos anteriores.

Se utilizaron los *fixtures*[24] de Ruby on rails, que son una forma de organizar los datos para el testing, en archivos de extensión yml, y se integran fácilmente con la herramienta de testing automático en el back-end. Los *fixtures* permiten además de definir los datos, relacionar a los mismos entre sí de forma sencilla a través de referencias. A continuación se muestra un ejemplo de datos definidos para la ejecución de pruebas:

Referencia	**e-mail**	**nombre**	**apellido**	**fecha nacimiento**	**contraseña**	**coordinador**
c1	coordinador@mail.com	Coordinador	Sistema	18/7/1980	Coord123	si
d1	usuario1@mail.es	Juan	Gómez	14/5/1990	JuanG1899	no
d2	usuario2@mail.uy	Sistema	Márquez	25/11/1985	aMarquez25	no

Tabla 2: Ejemplo de datos de prueba.

En este ejemplo de usuarios definidos para la aplicación web, se puede apreciar que se crearon referencias a los mismos, utilizadas por la herramienta *fixtures* para relacionar dichos datos con otros de otra naturaleza, por ejemplo relacionar al usuario d_2 con el grupo g_1 del sistema, definido en otro archivo. Al comenzar las pruebas automáticas, la herramienta carga los datos y ejecuta las pruebas. Se definió en función de los datos los resultados esperados de las pruebas.

5.2. Clases de equivalencia

Además de los datos de prueba, se definieron clases de equivalencia para poder crear tests significativos sobre el sistema. Las clases de equivalencia son conjuntos de datos de entrada, donde a cada conjunto le corresponde una única salida. Es decir que debe haber una única salida por cada clase definida. Las clases de equivalencia se definieron para cada caso de uso en el que tuviese sentido hacerlo, y se usaron tanto para las pruebas automáticas como para las manuales. Las mismas consistieron en ingresar los datos en cada caso de entrada y verificar la salida correspondiente. Se trató de cubrir la mayoría de las posibilidades de entradas de forma de generar robustez en las aplicaciones y corroborar casos borde. A continuación se muestra un ejemplo de clase de equivalencia, para el caso de uso de inicio de sesión:

- e-mail correcto:
  - No tiene restricciones sobre caracteres alfanuméricos, y
  - contiene un solo @, y
  - corresponde a un usuario registrado del sistema.
- e-mail incorrecto:
  - Contiene espacios, o
  - no contiene @, o
  - contiene más de un @, o
  - no corresponde a un usuario registrado del sistema, o
  - es vacío.
- contraseña correcta:
  - Corresponde al e-mail ingresado.
- contraseña incorrecta:
  - No corresponde al e-mail ingresado, o
  - es vacía.

Para este ejemplo, y las posibles entradas, las clases de equivalencias quedaron definidas como:

Clases	e-mail	contraseña	resultado o salida
1	correcto: usuario1@mail.es	correcta: JuanG1899	Inicio de sesión exitoso
2	correcto: usuario1@mail.es	incorrecta: no corresponde al e-mail	e-mail y contraseña no coinciden
3	incorrecto: e-mail no registrado	correcta o incorrecta	e-mail no registrado
4	incorrecto: no respeta formato	correcta o incorrecta	Formato incorrecto
5	incorrecto: e-mail vacío	correcta o incorrecta	Debe ingresar el e-mail
6	correcto o incorrecto	incorrecta: contraseña vacía	Debe ingresar la contraseña

Tabla 3: Ejemplo de clases de equivalencia.

A partir de este ejemplo, se puede decir que se tienen 6 casos de prueba significativos para el sistema, ya que para cada clase se define un caso. Para los demás casos de uso en los que tiene sentido hacerlo, se definieron clases de equivalencia similares a las del ejemplo, que permitieron encontrar y corregir fallas en las funcionalidades de ambas aplicaciones.

5.3. Pruebas de back-end automatizadas

Estas pruebas se realizaron con la herramienta RSpec[25] que permite especificar pruebas para código desarrollado en lenguaje Ruby. Para cada caso de uso planteado en los requerimientos funcionales se realizaron distintas pruebas que verifican su comportamiento esperado dada una entrada de datos determinada, llegando a un total de 71 pruebas. Las mismas recorren todo el back-end, desde los puntos de entrada de la back-end API hasta los modelos que acceden a la base de datos. Es decir, se verifica que con un conjunto de datos previamente especificado, al invocar un método de la back-end API que implementa el caso de uso en cuestión, el mismo se ejecuta con éxito y los resultados a nivel de la respuesta y del nuevo estado de base de datos son los esperados. Para estas pruebas se utilizaron los *fixtures* mencionados en la sección 5.1 (datos de prueba). Además, una vez terminado un caso de uso, se crearon las pruebas automatizadas para el mismo antes de integrar los cambios en el repositorio de Git. Con esto el equipo se aseguró que las nuevas funcionalidades no impacten negativamente en otras previamente desarrolladas.

5.4. Pruebas desde el front-end

En esta sección se habla de las pruebas desde el front-end tanto de la aplicación de escritorio como de la aplicación web. Éstas se realizaron utilizando los datos previamente establecidos, al final de cada iteración, para verificar el funcionamiento de todos los casos de uso. Cada prueba corresponde a una clase de equivalencia, es decir que se ingresa la entrada de dicha clase y se corrobora la salida correspondiente. El objetivo de las mismas fue buscar fallas, más que nada visuales y de comunicación con el back-end, y verificar las salidas obtenidas con los resultados esperados. Cada vez que se detectaron errores en la aplicación, los mismos fueron reportados y reparados en la misma iteración.

5.5. Pruebas de usabilidad con usuarios finales

Una vez obtenido un conjunto de funcionalidades implementadas bastante amplio se precedió a realizar pruebas de usabilidad con las docentes Teresa Pérez y Nora Ravaioli. Para las mismas se desarrolló un documento indicando el procedimiento de las distintas pruebas a realizar. Para estudiar los resultados se pidió a las docentes que realicen ciertas

acciones sin recibir ayuda de parte del equipo, y de esta forma se pudieron detectar fallas en aspectos como la facilidad de uso, interpretaciones de mensajes e interfaces, así como también analizar que tan útil y eficaz puede ser la plataforma para los usuarios. Además de esto, se les otorgó una encuesta de satisfacción en la cuál expresaron sus comentarios y calificaron distintos aspectos del sistema. Luego de realizar los cambios que surgieron de las pruebas, las profesoras se mostraron satisfechas.

5.6. Pruebas de despliegue

Se realizaron pruebas de despliegue tanto de la aplicación desarrollada en Angular 5 como de la API construida en Ruby on rails. Para la API se utilizó la plataforma Heroku[26] y la misma se publicó de manera satisfactoria en la siguiente url:

`https://tapaybusca.herokuapp.com/`

Por otro lado, el sitio construido con Angular 5 se publicó en la plataforma Firebase[27] y se encuentra disponible en la siguiente url:

`https://tapaybusca-2fe5d.firebaseapp.com/`

Con el fin de testear el sistema en su totalidad, también se desplegó el web service del repositorio central en Heroku, ahora disponible en la siguiente url:

`https://repositoriocentral.herokuapp.com/`

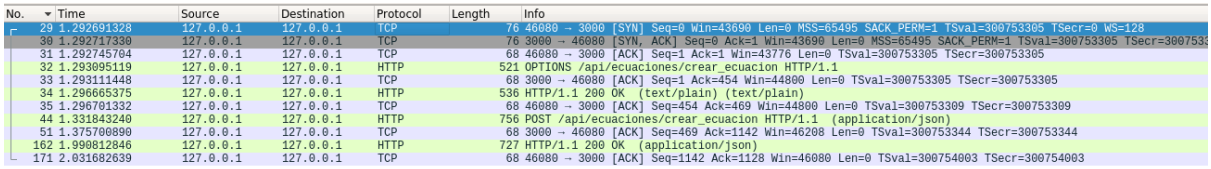
Una vez desplegados todos los sistemas, se realizaron pruebas para verificar el correcto funcionamiento de todos los casos de uso, desde la aplicación de escritorio hasta la aplicación web. Se ejecutaron todas las pruebas manuales desde front-end definidas previamente, con flujos en los cuales interactúan todos los sistemas: front-end, back-end, repositorio central y aplicación de escritorio, todos ellos alojados en las plataformas mencionadas.

5.7. Pruebas de concurrencia

El servidor web construido en Ruby on rails es ejecutado con la ayuda de Puma[28], un servidor simple, rápido, con soporte para *threads*, y altamente concurrente, para aplicaciones Ruby. Puma atiende cada solicitud en un thread independiente obtenido de un *thread pool* interno. El servidor se dejó por defecto con un máximo de 5 threads, valor que coincide con el tamaño de threads de Active Record (patrón de arquitectura utilizado para la persistencia de objetos en Rails). A su vez, Puma permite el "clustered mode"que bifurca en *workers* el proceso maestro, donde cada proceso hijo tiene su propio *thread pool*. El valor de workers establecido fue 2, y con esto se tiene que la concurrencia de la aplicación es de 2*5 = 10. Por otro lado, el número de solicitudes atendidas por minuto depende, entre otras cosas, de la potencia del servidor y del ancho de banda. Las pruebas de concurrencia consistieron en utilizar el sistema desplegado con más de dos usuarios a la vez, ejecutando al mismo tiempo los mismos casos de uso. Los resultados fueron satisfactorios.

5.8. Pruebas de seguridad con HTTPS y SSL/TLS

Para verificar la comunicación sobre el protocolo HTTPS entre el cliente y el servidor web se realizaron pruebas con el software Wireshark[29]. Este software es un analizador de protocolos de red y permite, entre otras cosas, realizar capturas de tráfico de ciertas interfaces de red. Las primeras pruebas se realizaron sin HTTPS y los resultados se presentan a continuación:



No.	Time	Source	Destination	Protocol	Length	Info
29	1.292691328	127.0.0.1	127.0.0.1	TCP	76	46080 → 3000 [SYN] Seq=0 Win=43690 Len=0 MSS=65495 SACK_PERM=1 TSval=300753305 TSecr=0 WS=128
30	1.292717330	127.0.0.1	127.0.0.1	TCP	76	3000 → 46080 [SYN, ACK] Seq=0 Ack=1 Win=43690 Len=0 MSS=65495 SACK_PERM=1 TSval=300753305 TSecr=300753
31	1.292745704	127.0.0.1	127.0.0.1	TCP	68	46080 → 3000 [ACK] Seq=1 Ack=1 Win=43776 Len=0 TSval=300753305 TSecr=300753305
32	1.293095119	127.0.0.1	127.0.0.1	HTTP	521	OPTIONS /api/ecuaciones/crear_ecuacion HTTP/1.1
33	1.293111448	127.0.0.1	127.0.0.1	TCP	68	3000 → 46080 [ACK] Seq=1 Ack=454 Win=44800 Len=0 TSval=300753305 TSecr=300753305
34	1.296665375	127.0.0.1	127.0.0.1	HTTP	536	HTTP/1.1 200 OK (text/plain) (text/plain)
35	1.296701332	127.0.0.1	127.0.0.1	TCP	68	46080 → 3000 [ACK] Seq=454 Ack=469 Win=44800 Len=0 TSval=300753309 TSecr=300753309
44	1.331843240	127.0.0.1	127.0.0.1	HTTP	756	POST /api/ecuaciones/crear_ecuacion HTTP/1.1 (application/json)
51	1.375700890	127.0.0.1	127.0.0.1	TCP	68	3000 → 46080 [ACK] Seq=469 Ack=1142 Win=46208 Len=0 TSval=300753344 TSecr=300753344
162	1.990812846	127.0.0.1	127.0.0.1	HTTP	727	HTTP/1.1 200 OK (application/json)
171	2.031682639	127.0.0.1	127.0.0.1	TCP	68	46080 → 3000 [ACK] Seq=1142 Ack=1128 Win=46080 Len=0 TSval=300754003 TSecr=300754003

Figura 30: Tráfico de mensajes sin HTTPS

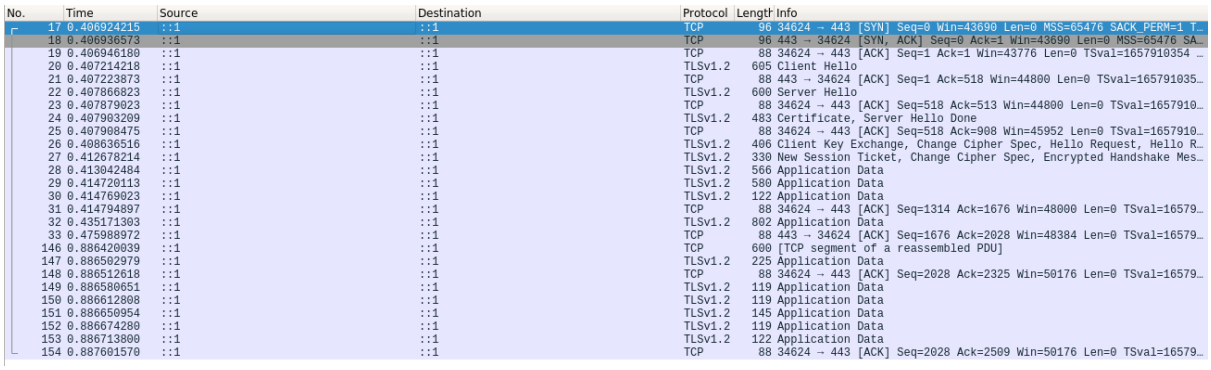
```
POST /api/ecuaciones/crear_ecuacion HTTP/1.1
Host: localhost:3000
Connection: keep-alive
Content-Length: 113
Origin: http://localhost:4200
access-token: wdQaH4MCXqbi2swpdJd67Q
client: nKnqc845k1SBEEunSWax7g
Content-Type: application/json
Accept: application/json
expiry: 1541212432
uid: franco19ps@gmail.com
token-type: Bearer
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/67.0.3396.62 Safari/
537.36
Referer: http://localhost:4200/crear-ecuacion
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9

{
  "ecuacion": "(x+pot1_3(3))=2",
  "ecuacion_katex": "(x+3^{1\\over 3})=2",
  "email": "franco19ps@gmail.com"
}HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, OPTIONS, DELETE, PUT
Access-Control-Expose-Headers: access-token, expiry, token-type, uid, client
Access-Control-Max-Age: 1728000
Content-Type: application/json; charset=utf-8
access-token: wdQaH4MCXqbi2swpdJd67Q
token-type: Bearer
client: nKnqc845k1SBEEunSWax7g
expiry: 1541212432
uid: franco19ps@gmail.com
Vary: Origin
ETag: W/"e156c1de751f0fc41f3ce5edefd256d2"
Cache-Control: max-age=0, private, must-revalidate
X-Request-Id: 70b0dba2-4781-4a32-821f-7f228a191e4a
X-Runtime: 0.658330
Transfer-Encoding: chunked

{"success":true,"errors":[]}
```

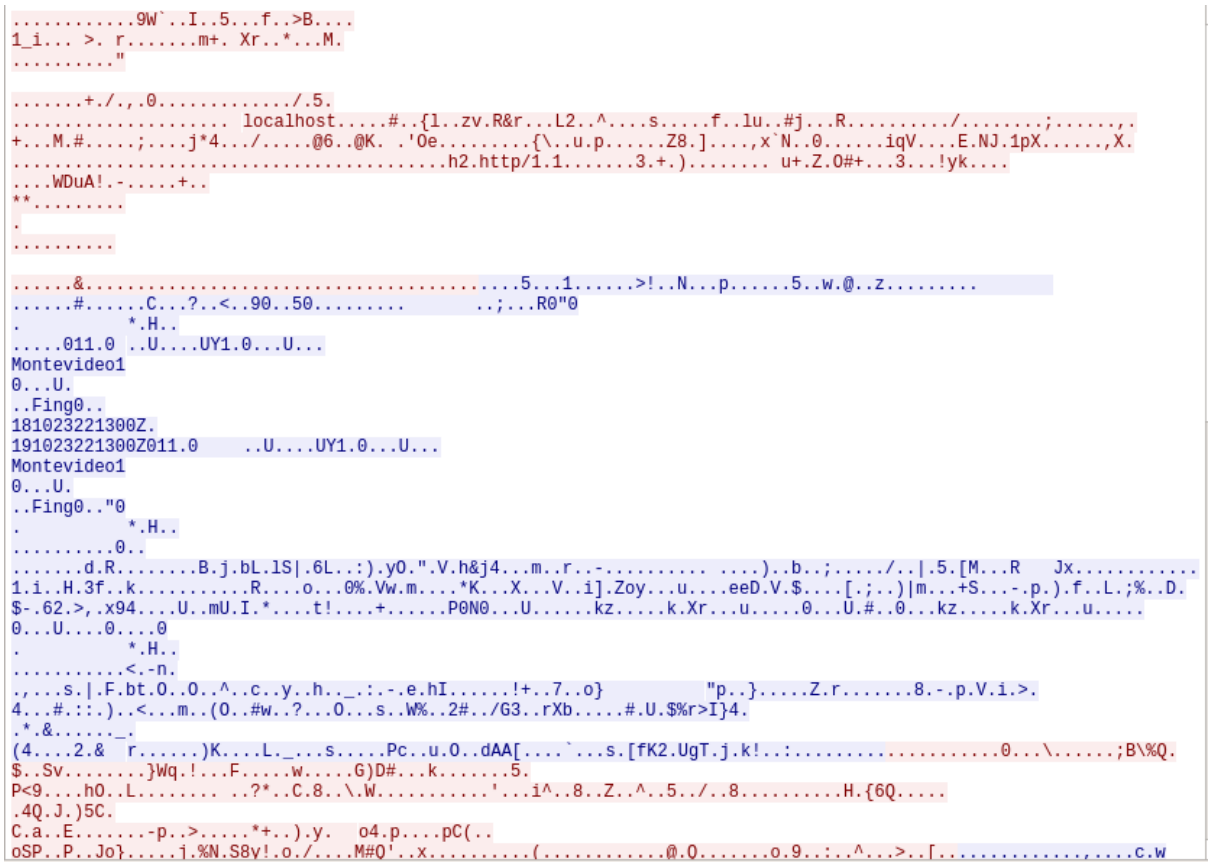
Figura 31: Mensajes sin HTTPS

Como se puede apreciar en la primera imagen, las solicitudes desde el cliente llegan por HTTP al puerto 3000 de 127.0.0.1 (localhost). En la segunda imagen se puede visualizar como un POST para crear una ecuación y su respuesta viajan totalmente en texto plano, siendo un grave problema de seguridad. Por otro lado, se realizó la misma prueba con HTTPS y los resultados fueron los siguientes:



No.	Time	Source	Destination	Protocol	Length	Info
17	0.406924215	::1	::1	TCP	96	34624 → 443 [SYN] Seq=0 Win=43690 Len=0 MSS=65476 SACK_PERM=1 T…
18	0.406936573	::1	::1	TCP	96	443 → 34624 [SYN, ACK] Seq=0 Ack=1 Win=43690 Len=0 MSS=65476 SA…
19	0.406946180	::1	::1	TCP	88	34624 → 443 [ACK] Seq=1 Ack=1 Win=43776 Len=0 TSval=1657910354 …
20	0.407214218	::1	::1	TLSv1.2	605	Client Hello
21	0.407223873	::1	::1	TCP	88	443 → 34624 [ACK] Seq=1 Ack=518 Win=44800 Len=0 TSval=165791035…
22	0.407866823	::1	::1	TLSv1.2	600	Server Hello
23	0.407879023	::1	::1	TCP	88	34624 → 443 [ACK] Seq=518 Ack=513 Win=44800 Len=0 TSval=1657910…
24	0.407903209	::1	::1	TLSv1.2	483	Certificate, Server Hello Done
25	0.407908475	::1	::1	TCP	88	34624 → 443 [ACK] Seq=518 Ack=908 Win=45952 Len=0 TSval=1657910…
26	0.408636516	::1	::1	TLSv1.2	406	Client Key Exchange, Change Cipher Spec, Hello Request, Hello R…
27	0.412678214	::1	::1	TLSv1.2	330	New Session Ticket, Change Cipher Spec, Encrypted Handshake Mes…
28	0.413042484	::1	::1	TLSv1.2	566	Application Data
29	0.414720113	::1	::1	TLSv1.2	580	Application Data
30	0.414769023	::1	::1	TLSv1.2	122	Application Data
31	0.414794897	::1	::1	TCP	88	34624 → 443 [ACK] Seq=1314 Ack=1676 Win=48000 Len=0 TSval=16579…
32	0.435171303	::1	::1	TLSv1.2	802	Application Data
33	0.475988972	::1	::1	TCP	88	443 → 34624 [ACK] Seq=1676 Ack=2028 Win=48384 Len=0 TSval=16579…
146	0.886420039	::1	::1	TCP	600	[TCP segment of a reassembled PDU]
147	0.886502979	::1	::1	TLSv1.2	225	Application Data
148	0.886512618	::1	::1	TCP	88	34624 → 443 [ACK] Seq=2028 Ack=2325 Win=50176 Len=0 TSval=16579…
149	0.886580651	::1	::1	TLSv1.2	119	Application Data
150	0.886612808	::1	::1	TLSv1.2	119	Application Data
151	0.886650954	::1	::1	TLSv1.2	145	Application Data
152	0.886674280	::1	::1	TLSv1.2	119	Application Data
153	0.886713800	::1	::1	TLSv1.2	122	Application Data
154	0.887601570	::1	::1	TCP	88	34624 → 443 [ACK] Seq=2028 Ack=2509 Win=50176 Len=0 TSval=16579…

Figura 32: Tráfico de mensajes a través de HTTPS



```
...........9W`..I..5...f..>B....
1_i... >. r.......m+. Xr..*...M.
.........."

.......+./.,.0.............../.5.
..................... localhost.....#..{l..zv.R&r...L2..^....s.....f..lu..#j...R.........../........;......,.
+...M.#.....;....j*4.../.....@6..@K. .'Oe.........{\..u.p......Z8.]....,x`N..0......iqV....E.NJ.1pX......,X.
.........................................h2.http/1.1.......3.+.)........ u+.Z.O#+...3...!yk....
....WDuA!.-.....+..
**.........
.
..........

......&.....................................5...1......>!..N...p......5..w.@..z.........
......#......C...?..<..90..50.........       ..;...R0"0
.           *.H..
.....011.0 ..U....UY1.0...U...
Montevideo1
0...U.
..Fing0..
181023221300Z.
191023221300Z011.0     ..U....UY1.0...U...
Montevideo1
0...U.
..Fing0.."0
.           *.H..
..........0..
.......d.R........B.j.bL.lS|.6L..:).yO.".V.h&j4...m..r..-.......... ....)..b..;...../..|.5.[M...R   Jx............
1.i..H.3f..k...........R....o...0%.Vw.m....*K...X...V..i].Zoy...u....eeD.V.$....[.;..)|m...+S...-.p.).f..L.;%..D.
$-.62.>,.x94....U..mU.I.*....t!....+......P0N0...U......kz.....k.Xr...u.....0...U.#..0...kz.....k.Xr...u.....
0...U....0....0
.           *.H..
..........<.-n.
.,...s.|.F.bt.O..O..^..c..y..h.._.:.-.e.hI......!+..7..o}          "p..}.....Z.r.......8.-.p.V.i.>.
4...#.::.)..<...m..(O..#w..?...O...s..W%..2#../G3..rXb.....#.U.$%r>I}4.
.*.&......_.
(4....2.&  r......)K....L._...s.....Pc..u.O..dAA[....`...s.[fK2.UgT.j.k!..:.....................0...\......;B\%Q.
$..Sv........}Wq.!...F.....w.....G)D#...k.......5.
P<9....hO..L........ ..?*..C.8..\.W..........'...i^..8..Z..^..5../..8..........H.{6Q.....
.4Q.J.)5C.
C.a..E.......-p..>.....*+..).y.   o4.p....pC(..
oSP..P..Jo}.....j.%N.S8y!.o./....M#Q'..x..........(............@.Q.......o.9..:..^...>..[..............,....c.w
```

Figura 33: Mensajes a través de HTTPS

En la primera imagen se visualiza, entre otras cosas, el intercambio de claves (*Handshake Protocol*) y los mensajes de datos de aplicación (*Application Data*) sobre el protocolo TLSv1.2. Esto nos indica que las solicitudes entre el cliente y el servidor web están viajando de manera segura sobre este protocolo, lo cual se puede apreciar en la segunda imagen con un mensaje de datos de aplicación encriptado.

6. Conclusiones y trabajo futuro

El objetivo general del proyecto es la construcción de una aplicación web que permite a docentes de matemática de educación media, organizar, gestionar y visualizar el avance del aprendizaje de sus alumnos en la resolución de ecuaciones por el método "Cover Up" o "la tapadita". Los objetivos específicos son, por un lado, resolver la comunicación entre la aplicación web y la aplicación de escritorio, que es la que utilizan los estudiantes para resolver las ecuaciones, y por otro lado crear y añadir nuevas funcionalidades a la aplicación de escritorio que son significativas para el seguimiento de los docentes.

Tanto el objetivo general como los específicos fueron alcanzados satisfactoriamente, respetando los plazos descritos en la planificación, lo que lleva al equipo a concluir que la especificación y el alcance fueron adecuados para un proyecto de grado. Cabe agregar que las profesoras Ravaioli y Pérez, consideran que sus expectativas fueron satisfechas.

Desde el punto de vista de la formación del equipo del proyecto, se puede afirmar que en el transcurso del mismo se aprendieron a utilizar nuevas técnicas y herramientas, como lo son lenguajes de programación, librerías y frameworks. Asimismo se profundizó en conocimientos previamente adquiridos, por ejemplo en el diseño de arquitectura de software, en la buena reutilización de las funcionalidades provistas en el proyecto anterior, así como también en uniformizar la comunicación y el dominio de datos entre sistemas con características diferentes.

La naturaleza del proyecto mantuvo al equipo motivado. Se considera que la comunicación entre los integrantes, los tutores y las profesoras (clientes) fue fluida, facilitando el seguimiento y la evaluación de las diferentes partes, hasta el resultado final.

El problema planteado y el diálogo con las profesoras dio lugar a un conjunto de requisitos que fue acotado para no exceder el alcance del proyecto. Se establecieron prioridades y se construyó el documento de especificación de requisitos con los que se consideraron más importantes y que conformaban un conjunto adecuado. Aquellos que no quedaron en ese conjunto se describen a continuación, como ideas para futuros proyectos.

- En relación a la aplicación de escritorio:
  - agregar la entidad Paquete, y construir casos de uso como por ejemplo: visualizar las ecuaciones que se asignaron a un paquete, visualizar a qué grupos fue asignado un paquete, y sincronizar la información seleccionando (además del grupo) un paquete,
  - añadir la posibilidad de que varios alumnos se encuentren asociados a la aplicación de manera simultánea, en caso de que varias personas usen la misma computadora. Esto permitiría la incorporación de casos de uso como por ejemplo: sincronizar información dependiendo no sólo del grupo sino también del alumno,
  - durante el testing de la creación de ecuaciones se detectó un error en la aplicación de escritorio que consiste en no procesar correctamente ecuaciones con números extremadamente grandes, de muchas cifras, devolviendo "error de división entre 0". Se podría probar de forma exhaustiva el conjunto de ecuaciones soportadas, con el objetivo de robustecer la creación de ecuaciones en la aplicación de escritorio y con ello la creación en la aplicación web.
- En relación al sistema web:

- introducir la eliminación de ecuaciones del conjunto global, y la eliminación de paquetes,
- añadir opciones del perfil de los docentes y coordinadores, por ejemplo subir una foto y modificar sus datos,
- crear un foro de mensajes para facilitar la comunicación entre docentes y coordinadores, así como compartir paquetes de ecuaciones entre docentes,
- mejorar la interfaz de creación de ecuaciones, logrando una interacción similar a la de la aplicación de escritorio en donde se disminuye la posibilidad de errores de tipeo que podría ocasionar el docente,
- verificar que los docentes registrados en el sistema sean realmente docentes de matemática habilitados,
- en cuanto a seguridad, se debería establecer conexiones bajo el protocolo HTTPS entre el servidor web y el web service del repositorio central para que los mensajes viajen de manera encriptada,
- construir una aplicación web con las mismas funcionalidades que la aplicación de escritorio, es decir un sistema web donde los alumnos resuelvan ecuaciones utilizando un navegador de Internet sin la necesidad de instalar la aplicación en sus computadoras.

7. Referencias

[1] Moodle: Impulsado por la comunidad, soportado globalmente.
https://moodle.org/.

[2] ThatQuiz: tu examen de matemáticas - tu test de álgebra - tu ejercicio de geometría - se hacen aquí.
https://www.thatquiz.org/es/.

[3] MathPapa, step-by-step algebra help.
https://www.mathpapa.com/.

[4] Angular. One framework. Mobile and desktop.
https://angular.io/.

[5] React, A JavaScript library for building user interfaces.
https://reactjs.org/.

[6] Ruby on Rails, A web-application framework that includes everything.
https://rubyonrails.org/.

[7] .Net: Free, Cross-platform, Open source, A developer platform for building mobile and web app.
https://www.microsoft.com/net.

[8] TypeScript - JavaScript that scales.
https://www.typescriptlang.org/.

[9] Mathjax, beautiful math in all browsers.
https://www.mathjax.org/.

[10] Angular module to write beautiful math expressions in TeX syntax.
https://github.com/garciparedes/ng-katex.

[11] ChartJS, Simple yet flexible JavaScript charting for designers and developers.
https://www.chartjs.org/.

[12] Bootstrap, The most popular HTML, CSS, and JS library in the world.
https://getbootstrap.com/.

[13] HTTPS - Hyper Text Transfer Protocol Secure.
https://www.instantssl.com/ssl-certificate-products/https.html.

[14] Capa de conexión segura SSL.
https://www.digicert.com/es/ssl.htm.

[15] Launch4j - Cross-platform Java executable wrapper.
http://launch4j.sourceforge.net/.

[16] InnoSetup - Installer for Windows programs.
http://www.jrsoftware.org/isinfo.php.

[17] Bitbucket — The Git solution for professional teams.
https://bitbucket.org/.

[18] Devise - Flexible authentication solution for Rails based on Warden.
https://github.com/plataformatec/devise.

[19] DeviseTokenAuth - Token-based authentication for Rails.
https://github.com/lynndylanhurley/devise_token_auth.

[20] Angular2Token - Token-based authentication service for Angular.
https://www.npmjs.com/package/angular2-token.

[21] SendGrid - Email Delivery Service.
https://sendgrid.com/.

[22] TeX - Sistema de tipografía escrito por Donald E. Knuth.
http://www.tug.org/.

[23] Rails Guides - Active Record Basics.
https://guides.rubyonrails.org/active_record_basics.html.

[24] Fixtures of Ruby on rails.
https://api.rubyonrails.org/v3.1/classes/ActiveRecord/Fixtures.html.

[25] RSpec: Behaviour Driven Development for Ruby.
http://rspec.info/.

[26] Heroku: Cloud Application Platform.
https://www.heroku.com/.

[27] Firebase: Herramienta de Apps — Olvídate de la infraestructura.
https://firebase.google.com/.

[28] Puma: A Modern, Concurrent Web Server for Ruby.
http://puma.io/.

[29] Wireshark - Go Deep.
https://www.wireshark.org/.

8. Anexos

Índice de figuras

Índice de tablas