DISEÑO E IMPLEMENTACIÓN DE UNA APLICACIÓN WEB PARA LA ENSEÑANZA DE VOLUMETRÍAS DE NEUTRALIZACIÓN EN ANÁLISIS QUÍMICO CUANTITATIVO

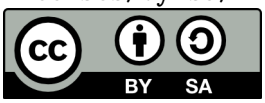
Harold Sánchez Ospina

Trabajo de grado presentado como requisito
para optar al título de
Ingeniero de Sistemas y Computación

Director
César Augusto Jaramillo Acevedo

UNIVERSIDAD TECNOLÓGICA DE PEREIRA
Programa de Ingeniería Sistemas y Computación.
Pereira, Septiembre de 2019

DISEÑO E IMPLEMENTACIÓN DE UNA APLICACIÓN WEB PARA LA ENSEÑANZA DE VOLUMETRÍAS DE NEUTRALIZACIÓN EN ANÁLISIS QUÍMICO CUANTITATIVO



Harold Sánchez Ospina

Pereira, Septiembre de 2019
Programa de Ingeniería de Sistemas y Computación.
Universidad Tecnológica de Pereira
La Julita. Pereira(Colombia)
TEL: (+57)(6)3137122
www.utp.edu.co
Versión web disponible en: *http://recursosbiblioteca.utp.edu.co/tesisd/index.html*

Agradecimientos

Agradezco principalmente a mi familia por el gran apoyo brindado durante mi proceso de formación profesional. A la Universidad Tecnológica de Pereira por ser la institución en la cual se adquirieron los conocimientos para hoy estar optando a este título. A mis docentes por el acompañamiento en el proceso y brindar de manera abierta sus conocimientos.

Tabla de Contenido

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

La volumetría de neutralización es una técnica clásica para análisis químico cuantitativo, su propósito es hallar el contenido de un analito problema con base en una reacción de neutralización mediante el punto de equivalencia en una titulación volumétrica.

En la actualidad el curso de química analítica que se imparte en la Universidad Tecnológica de Pereira a estudiantes de Tecnología Química y Química Industrial es teórico-práctico, la mayor parte corresponde a diferentes métodos del análisis químico cuantitativo y en este se estudian las volumetrías de neutralización usando las técnicas de Warder y Winkler para el estudio de mezclas de álcalis.

Durante la clase se aborda el fundamento teórico con sus cálculos estequiométricos y en el laboratorio se plantean problemas experimentales, donde el estudiante debe aplicar los conceptos teóricos y realizar una titulación volumétrica del analito problema con un estándar de concentración en presencia de un indicador (cambio de color), para observar el punto final de la reacción. Finalmente el estudiante debe reportar el contenido del analito problema con base en la reacción de neutralización (ácido-base) y los volúmenes gastados en la titulación.

Capítulo 2

Definición del problema

El desarrollo de la práctica de laboratorio se basa en una guía impresa, que detalla los procedimientos a seguir, tanto para la estandarización como para cada una de las técnicas.

Esta guía cumple su propósito esencial de dar los pasos al estudiante para la ejecución de sus experimentos; pero la práctica al contar con: un análisis inicial, mezclas, observaciones y cálculos matemáticos; puede conducir a errores conceptuales o de ejecución.

Actualmente en la universidad no se cuenta con herramientas tecnológicas que guíen al estudiante de forma interactiva durante el desarrollo de sus prácticas. También los estudiantes al encontrarse limitados a las prácticas presenciales en el laboratorio, ya que se trata de una área experimental, no cuentan con herramientas alternativas tecnológicas para simular previamente los experimentos y afianzar los conocimientos teóricos previos.

Se busca construir una aplicación web para la enseñanza de volumetrías de neutralización en análisis químico cuantitativo; que permita simular las prácticas de laboratorio de manera interactiva y ubicua. Además permitirá al docente ampliar el contenido y actualizarlo a las necesidades actuales.

Capítulo 3

Justificación

La tecnología actualmente nos permite administrar y aprovechar la inmensa cantidad de datos disponibles y su uso ha ido en aumento desde principios de los años 90 con la llamada «era de la información». A su vez la educación ha sufrido una progresiva transformación. Un proceso en el que se abandona el papel y el lápiz en favor de las pantallas digitales; y con ello, la estática metodología tradicional cede paso al dinamismo, la creatividad y la modularidad. Características comúnmente ausentes en los entornos educativos actuales; por esto, surge la necesidad de brindar a estudiantes y docentes herramientas que permitan explorar de nuevas formas sus actividades académicas.

Actualmente los laboratorios que se imparten en la Escuela de Química, se desarrollan haciendo uso de guías impresas y estas por su carácter estático, tienen la desventaja de no poder ser actualizadas después de impresas y fomentan un uso inadecuado del papel. También por sus características inherentes, no permite la apreciación clara de colores y variaciones; puntos clave al momento de desarrollar las prácticas.

Este proyecto tiene como objetivo el desarrollo de una aplicación web que permita la representación digital de las guías y apoyarlas con recursos multimedia que faciliten al estudiante la comprensión de fenómenos clave, como las sutiles variaciones de color al momento de titular una muestra.

Capítulo 4

Objetivos Generales y específicos

4.1. Objetivo General

Diseñar e implementar una aplicación web para la enseñanza de volumetrías de neutralización en análisis químico cuantitativo.

4.2. Objetivos Específicos

- Levantamiento de requerimientos por medio de historias de usuario
- Utilizar herramientas de desarrollo apoyándose en el uso de metodologías ágiles
- Diseñar los módulos de estudiante y docente
- Desarrollar el prototipo y publicarlo

Capítulo 5

Marco Referencial

5.1. Marco de Antecedentes

El papel ha tenido una influencia importante en el desarrollo de la sociedad, permitiendo plasmar y transferir el conocimiento. Las guías de laboratorio especialmente diseñadas para estudiantes de química, se presentan impresas en papel y habitualmente cada estudiante debe tener una copia para desarrollar sus experimentos. Algunas veces el proceso de copiado puede inducir a pérdida de calidad y la representación en este medio limita la observación de fenómenos que dinámicos. También es común que no se cuente con el texto original digitalizado, lo que obliga a que una posible modificación, requiera una transcripción completa de la guía y esto ha llevado a que el texto haya entregado sin variaciones cada semestre.

Actualmente se cuenta con aplicativos que por separado pueden cumplir ciertos requerimientos planteados como son los editores de texto en tiempo real como Google Docs u Office 365 y estas aplicaciones permitirían redactar las guías y tener acceso a estas en cualquier lugar en su versión actualizada. También se pueden tener en cuenta aplicaciones para transmisión de videos por internet como YouTube o Vimeo, en las que podrían publicarse los ejemplos de los experimentos y apreciar los resultados o cambios de color de las reacciones.

Se hace necesario el desarrollo de una aplicación que integre diferentes recursos que mejoren la interacción con la guía de laboratorio, añadiendo facilidad en su actualización, recursos multimedia y que a su vez oriente al estudiante durante el desarrollo de

las prácticas, como un complemento directo a la guía.

5.2. Marco Teórico

5.2.1. Método de Warder

Este método puede llevarse a cabo sobre una porción única de muestra o sobre dos porciones iguales de ella. Cualquiera sea el caso, permite conocer qué componentes se encuentran presentes y la concentración de cada uno de ellos. A continuación se hace un análisis sencillo de cada uno de los casos posibles.

- **Método de Warder sobre una alícuota:** Se realizan dos titulaciones sucesivas usando indicadores que poseen distintos rangos de viraje (fenolftaleína y naranja de metilo). Se llamará VF al volumen de ácido normalizado necesario para producir la decoloración de la fenolftaleína y VV al volumen de ácido (contando a partir de VF) necesario para producir el cambio de color del indicador naranja de metilo.
- **Método de Warder sobre dos alícuotas:** Se realiza trabajando sobre dos porciones iguales de la muestra. Se llamará VF al volumen de ácido normalizado necesario para producir la decoloración de la fenolftaleína y VV al volumen de ácido necesario para producir el cambio de color del indicador naranja de metilo. Ambos volúmenes de ácido se medirán desde el cero de la bureta.

5.2.2. Método de Winkler

En este método se debe trabajar sobre dos porciones iguales de la misma muestra. En una de ellas, se valora con un ácido fuerte normalizado hasta viraje del indicador naranja de metilo. Esta operación se denomina determinación de la alcalinidad total. En la segunda alícuota se precipitan los iones CO32- (ya sean propios de la muestra o producidos por una reacción química conveniente) con BaCl2 neutro y se valoran los iones OH- presentes (propios de la muestra o los remanentes de un exceso agregado ex profeso), usando fenolftaleína como indicador.

5.2.3. Estandarización de ácidos y bases

La estandarización es un examen crítico de algunas sustancias sugeridas como tipos primarios para las valoraciones ácido-base,con vistas a realizar recomendaciones

concernientes a sus aplicaciones prácticas.

5.3. Marco Conceptual

5.3.1. Guía de laboratorio

Es documento que contiene la descripción y los procedimientos detallados, para guiar al estudiante en el desarrollo de los experimentos que serán ejecutados en el laboratorio.

5.3.2. Volumetría de neutralización

La volumetría de neutralización es una técnica clásica para análisis químico cuantitativo, su propósito es hallar el contenido de un analito problema con base en una reacción de neutralización mediante el punto de equivalencia en una titulación volumétrica.

5.3.3. Práctica de laboratorio

Es una actividad llevada a cabo por los estudiantes, donde se les plantea un problema y a través de métodos experimentales, apoyándose en la guía de laboratorio, encontrar la solución al planteamiento y entregar los resultados.

5.3.4. Alícuota

Es una parte que se toma en representación de un volumen o masa experimental, para su estudio individual en el laboratorio.

5.3.5. Intervalo de Viraje

Se trata del rango de valores de pH en el que un indicador manifiesta un cambio de color.

5.3.6. Titulación

Es un método químico cuantitativo que a partir de un reactivo con concentración conocida, permite hallar una concentración desconocida.

5.3.7. Analito

Es un término usado utilizado en química, y hace referencia a una sustancia, íon o un compuesto.

5.3.8. Reacción química

Se define como un proceso termodinámico de un sistema cerrado en el que las masas de sus componentes cambian unas a expensas de las otras de acuerdo con las leyes de la Química. [1]

5.3.9. Álcali

Compuesto que en disolución acuosa se comporta como una base fuerte. [2]

[1] Diccionario de la RAE https://dle.rae.es/?w=%C3%A1lcali
[2] Gayé, Jesús Biel (1997). Formalismo y métodos de la termodinámica

Capítulo 6

Diseño metodológico

6.1. Hipótesis

¿Es posible desarrollar una aplicación web que permita guiar al estudiante en el desarrollo de sus prácticas experimentales, incluyendo la guía y recursos multimedia?.

6.2. Tipo de investigación

Esta investigación tomará un enfoque cualitativo.

6.3. Población

Estudiantes y docentes de la escuela de química de la Universidad Tecnológica de Pereira

Capítulo 7

Diseño, implementación y pruebas del sistema

7.1. Análisis

Durante la primera etapa del desarrollo se realizó el levantamiento de requerimientos. Este levantamiento se hizo a través de historias de usuario, que pueden ser visualizadas en el conjunto de tablas.

Historia:	Ingreso al módulo de docente
ID:	1
Rol:	Docente - Administrador
Descripción	▪ Se debe poder ingresar al aplicativo por medio de una contraseña al módulo docente. ▪ Se debe poder cerrar la sesión iniciada.

Tabla 7.1: Historia 1. Ingreso al módulo de docente. Referencia: autores.

Historia:	Modificar Contraseña
ID:	2
Rol:	Docente
Descripción	A través del módulo docente, se debe poder cambiar la contraseña del módulo.

Tabla 7.2: Historia 2. Modificar Contraseña. Referencia: autores.

Historia:	Agregar o modificar contenido de las guías de laboratorio
ID:	3
Rol:	Docente
Descripción	A través del módulo docente, se debe permitir redactar las guías de laboratorio correspondientes a cada método experimentar. También se debe permitir añadir contenido multimedia.

Tabla 7.3: Historia 3. Agregar o modificar contenido de las guías de laboratorio. Referencia: autores.

7.2. Diseño del sistema

El diseño de la aplicación se modeló usando UML 2.0. A continuación se pueden observar los casos de uso, en la figura **??** su diagrama asociado.

De estos casos de uso se diseñaron 7 módulos, que componen las partes esenciales del sistema. Se comienza con los módulos base: Docente y Estudiante.

El Módulo de Docente tiene como función la redacción de las guías de laboratorio, por lo que hace uso del módulo Redactar guía.

En el Módulo de Estudiante se desarrollan las prácticas experimentales, por lo que se conecta a dos módulos principales, que se ejecutan normalmente en orden:

1. Estandarización

2. Método

Ambos módulos ejecutan el módulo central Reacción, que a partir de una reacción ejecutarán el experimento y guiarán al estudiante a través de sus pasos. Este módulo

Historia:	Ingreso al módulo de estudiante
ID:	4
Rol:	Estudiante
Descripción	El ingreso al módulo de estudiante se dará al abrir el aplicativo (sin contraseña)

Tabla 7.4: Historia 4. Ingreso al módulo de estudiante. Referencia: autores.

Historia:	Ver guía
ID:	5
Rol:	Estudiante
Descripción	▪ Se debe permitir consultar la guía general (todos los métodos) ▪ Se debe permitir consultar la guía de cada método experimental individualmente

Tabla 7.5: Historia 5. Ver guía. Referencia: autores.

hace uso de Ver Guía, que se encarga de mostrar la guía de laboratorio.

El Diagrama de Componentes asociado a los módulos se puede apreciar en la figura 7.2 y su Diagrama de Secuencia correspondiente en la figura 7.3.

Historia:	Estandarización
ID:	6
Rol:	Estudiante
Descripción	▪ Se debe tener la posibilidad de estandarizar ácidos y bases ▪ Se debe poder seleccionar el compuesto que desea estandarizar ▪ Se debe balancear la reacción correspondiente al compuesto seleccionado ▪ Se debe introducir los resultados experimentales obtenidos para cada una de las alícuotas ▪ El sistema debe calcular el resultado de cada alícuota y hacer un promedio de todas las alícuotas

Tabla 7.6: Historia 6. Estandarización. Referencia: autores.

Historia:	Aplicación de métodos experimentales
ID:	7
Rol:	Estudiante
Descripción	▪ Se debe poder seleccionar qué método desea aplicar, teniendo como opciones: Warder y Winkler ▪ Dentro de cada método, se debe poder seleccionar las reacciones correspondientes a cada tipo de práctica experimental ▪ El sistema debe guiar el desarrollo de cada método experimental y permitir introducir los datos obtenidos en cada etapa ▪ Cada reacción nueva que se presente, debe ser balanceada, antes de poder avanzar el en procedimiento ▪ Se debe introducir los resultados experimentales obtenidos para cada una de las alícuotas ▪ El sistema debe calcular el resultado de cada alícuota y hacer un promedio de todas las alícuotas

Tabla 7.7: Historia 7. Aplicación de métodos experimentales. Referencia: autores.

Historia:	Seguridad
ID:	8
Rol:	Sistema
Descripción	El sistema debe garantizar la seguridad de las contraseñas y el acceso al módulo docente

Tabla 7.8: Historia 8. Seguridad. Referencia: autores.

Caso de uso:	Ingreso al módulo de docente
Actor:	Docente
Flujo básico	1. El usuario entra a la URL y selecciona en el menú superior la opción "Docente" 2. El usuario ingresa su contraseña y selecciona el botón"Docente" 3. El sistema compara la contraseña y permite el acceso 4. El sistema carga las guías de laboratorio y las muestra en pantalla
Flujo alterno	Contraseña incorrecta 2. El usuario ingresa una contraseña incorrecta 3. El sistema verifica la contraseña, y arroja un mensaje de error
Pre-condiciones	
Post-condiciones	El usuario abrirá el módulo de usuario, donde se presentarán las guías para ser modificadas
Requisitos	Tener la contraseña de docente
Requerimientos	1

Tabla 7.9: Caso de uso 1. Ingreso al módulo de docente. Referencia: autores.

Caso de uso:	Salida del módulo docente
Actor:	Docente
Flujo básico	1. El usuario presiona el botón salir, ubicado en la esquina superior derecha
Flujo alterno	
Pre-condiciones	Estar dentro del módulo docente
Post-condiciones	El sistema regresa a la pantalla de inicio de sesión
Requisitos	Tener la contraseña de docente
Requerimientos	1

Tabla 7.10: Caso de uso 2. Salida del módulo docente. Referencia: autores.

Caso de uso:	Modificar contraseña
Actor:	Docente
Flujo básico	1. El usuario presiona la opción para cambiar de la contraseña 2. El sistema despliega un diálogo con dos campos de texto 3. El usuario introduce la nueva contraseña dos veces 4. El usuario presiona el botón de guardado
Flujo alterno	Contraseña incorrecta 3. Las contraseñas introducidas no coinciden 4. Se muestra un mensaje de error
Pre-condiciones	Estar dentro del módulo docente
Post-condiciones	Cambio de contraseña exitoso
Requisitos	
Requerimientos	2

Tabla 7.11: Caso de uso 3. Modificar contraseña. Referencia: autores.

Caso de uso:	Agregar o modificar contenido de las guías de laboratorio
Actor:	Docente
Flujo básico	1. El usuario selecciona alguno de los métodos del menú izquierdo 2. El sistema despliega las reacciones posibles al lado derecho, cada una de estas con un editor de texto con capacidades multimedia 3. El usuario introduce el contenido de la guía
Flujo alterno	
Pre-condiciones	Estar dentro del módulo docente
Post-condiciones	El sistema guarda automáticamente el texto introducido en cada uno de los editores
Requisitos	
Requerimientos	3

Tabla 7.12: Caso de uso 4. Agregar o modificar contenido de las guías de laboratorio. Referencia: autores.

Caso de uso:	Ingreso al módulo de estudiante
Actor:	Estudiante
Flujo básico	1. El usuario entra en la aplicación o en su URL base
Flujo alterno	Contraseña incorrecta 2. El usuario ha salido (docente) y se encuentra en la pantalla de inicio de sesión 3. El usuario presiona el botón “Estudiante”
Pre-condiciones	
Post-condiciones	El sistema entra en el módulo de estudiante
Requisitos	
Requerimientos	4

Tabla 7.13: Caso de uso 5. Ingreso al módulo de estudiante. Referencia: autores.

Caso de uso:	Ver guía general
Actor:	Estudiante
Flujo básico	1. El usuario presiona el botón “Guía completa” ubicado en una esquina de la pantalla
Flujo alterno	
Pre-condiciones	
Post-condiciones	El sistema despliega un diálogo en el que se pueden revisar todas las guías programadas en el sistema
Requisitos	Tener la contraseña de docente
Requerimientos	5

Tabla 7.14: Caso de uso 6. Ver guía general. Referencia: autores.

Caso de uso:	Estandarización
Actor:	Estudiante
Flujo básico	1. El usuario selecciona un tipo de estandarización 2. El sistema presenta la reacción correspondiente, con sus coeficientes acompañados de un selector asignado a un número aleatorio 3. El usuario selecciona los coeficientes correctos y balancea la reacción 4. El sistema desbloquea el procedimiento experimental, teniendo 3 alícuotas por defecto 5. El usuario llena los valores correspondientes a los resultados obtenidos para cada alícuota
Flujo alterno	
Pre-condiciones	
Post-condiciones	1. El sistema calcula los valores de cada alícuota 2. El sistema promedia los valores de todas las alícuotas y lo presenta en tiempo real, acompañado de sus unidades
Requisitos	Tener la contraseña de docente
Requerimientos	6

Tabla 7.15: Caso de uso 7. Estandarización. Referencia: autores.

Caso de uso:	Aplicación de métodos experimentales
Actor:	Estudiante
Flujo básico	1. El usuario selecciona un método experimental 2. El usuario selecciona una reacción del listado presentado 3. El sistema presenta la reacción correspondiente, con sus coeficientes acompañados de un selector asignado a un número aleatorio 4. El usuario selecciona los coeficientes correctos y balancea la reacción 5. El sistema desbloquea el procedimiento experimental, teniendo 3 alícuotas por defecto 6. El usuario llena los valores correspondientes a los resultados obtenidos para cada alícuota 7. Si el método lo amerita, se repetirán los pasos 3 - 6 para diferentes reacciones
Flujo alterno	
Pre-condiciones	
Post-condiciones	3. El sistema calcula los valores de cada alícuota 4. El sistema promedia los valores de todas las alícuotas y lo presenta en tiempo real, acompañado de sus unidades
Requisitos	Tener la contraseña de docente
Requerimientos	7

Tabla 7.16: Caso de uso 8. Aplicación de métodos experimentales. Referencia: autores.

Caso de uso:	Ver guía específica a una reacción
Actor:	Estudiante
Flujo básico	1. Frente a la reacción seleccionada se habilitará el botón "Guía" 2. El estudiante presiona el botón "Guía"
Flujo alterno	
Pre-condiciones	
Post-condiciones	El sistema despliega un diálogo en el que se puede revisar la guía específica a dicha reacción, aplicando el método experimental correspondiente
Requisitos	6, 7
Requerimientos	5

Tabla 7.17: Caso de uso 9. Ver guía específica a una reacción. Referencia: autores.

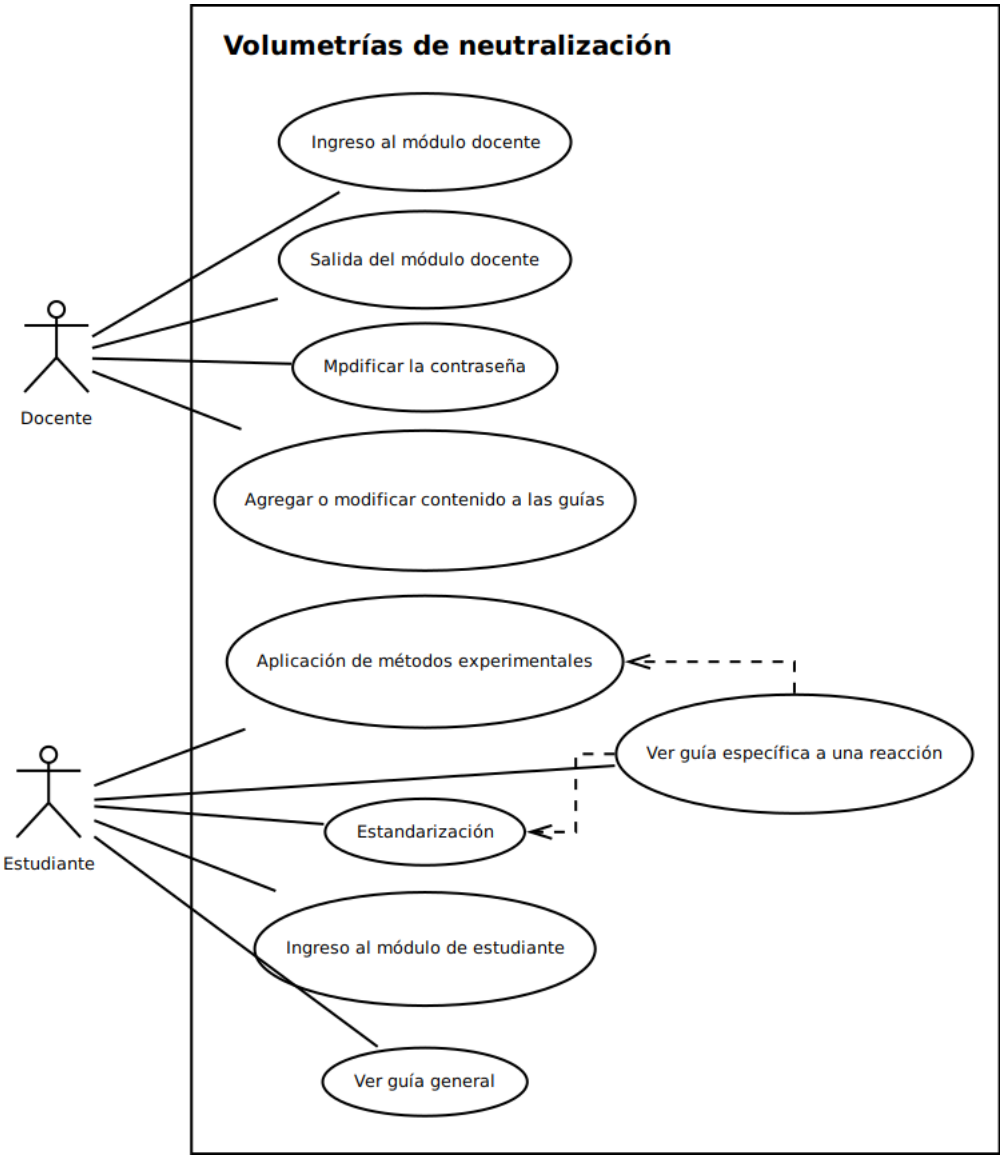


Figura 7.1: Diagrama de casos de uso. Referencia: autores.

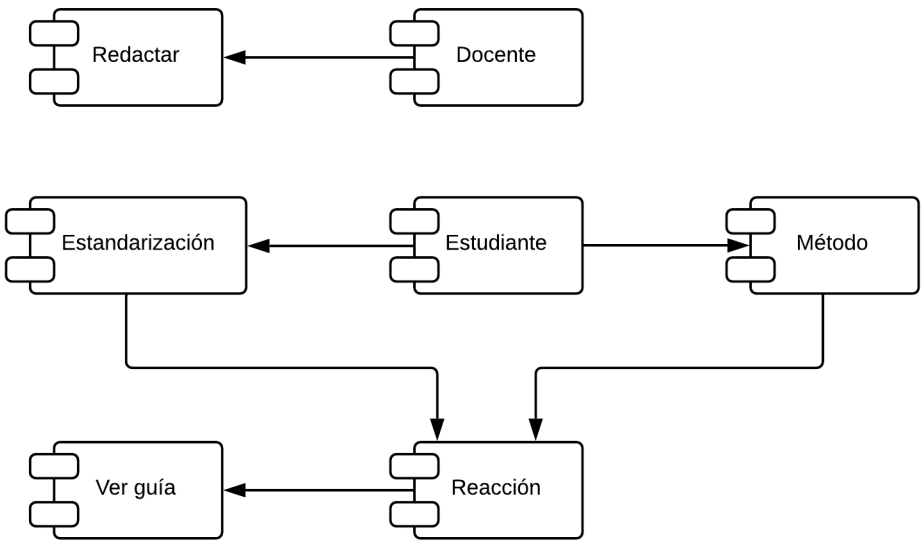


Figura 7.2: Diagrama de componentes. Referencia: autores.

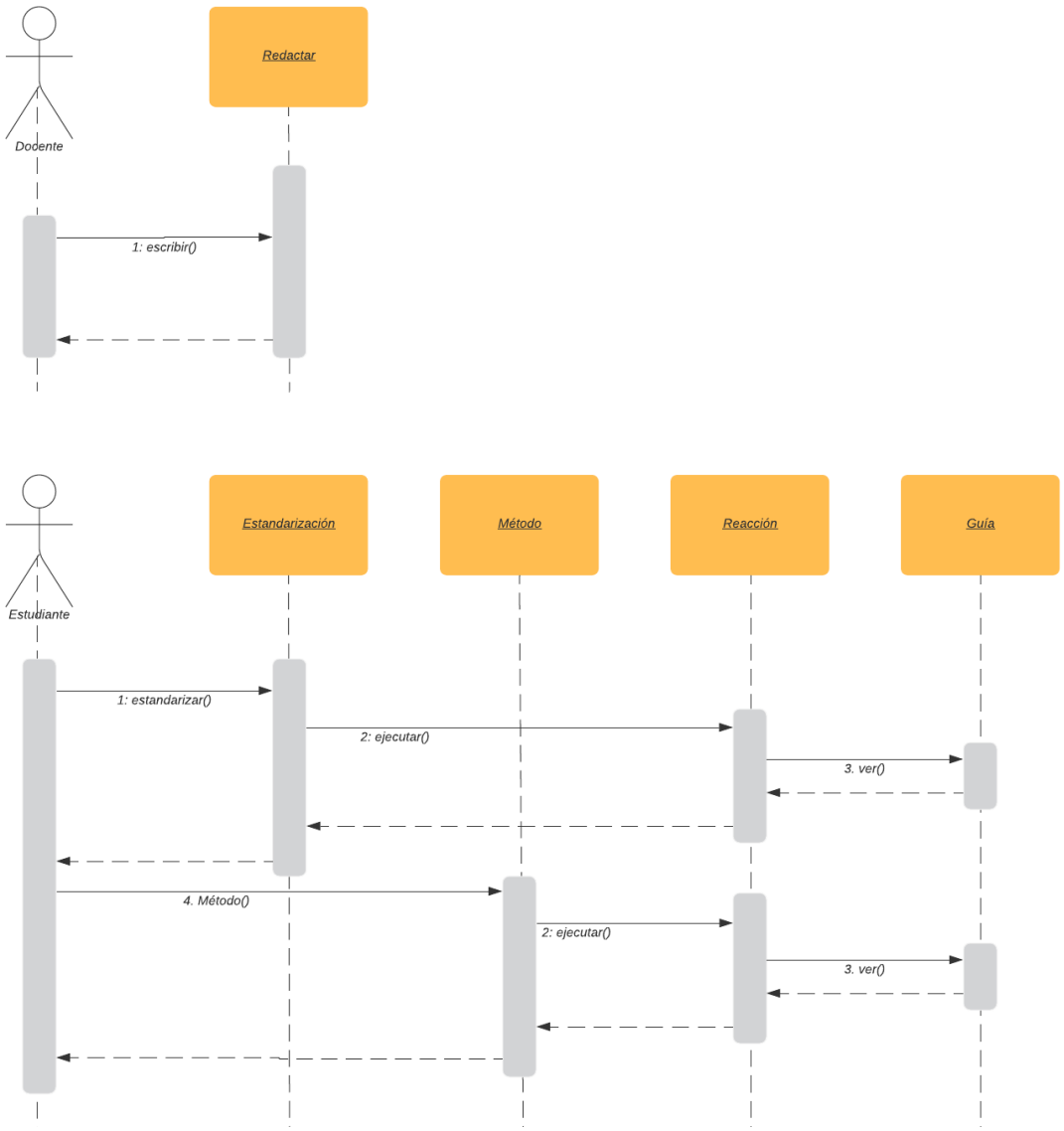


Figura 7.3: Diagrama de secuencia. Referencia: autores.

7.3. Implementación

Hasta el momento se expusieron los conceptos generales del diseño de la aplicación que soluciona el problema expuesto al inicio del documento. En este capítulo se relata cómo se desarrolló el prototipo de la aplicación, tecnologías, decisiones y mejoras aplicadas durante el proceso.

7.4. Elección de tecnologías

Se eligió desarrollar una aplicación web, ya que es una tecnología ampliamente usada en la actualidad. Esta permite que la misma aplicación pueda adaptarse a distintos tipos de dispositivo, ser accedida desde cualquier lugar y que cada usuario pueda tener la versión de los datos más reciente en todo momento.

Se optó por desarrollar una aplicación de una sola página (SPA); este es un método que realiza una única carga del sitio y a partir de este punto el sitio se ejecuta, cambia de ventanas y obtiene datos y sin tener que recargar la pantalla. Este enfoque de aplicación es muy útil para la aplicación que se desea desarrollar, ya que esta cuenta solamente con dos pantallas principales: Docente y Estudiante; y la mayor parte de las interacciones ocurren dentro del navegador. El tener la página en una sola carga reduce las peticiones al servidor y permite que la aplicación se pueda distribuir como una progressive web app (PWA), y pueda ser publicada como una aplicación para dispositivos móviles, usando la misma base de código.

Se usó la librería gráfica ReactJS para toda la representación gráfica de la aplicación. Esta librería cuenta con excelentes buenas características de rendimiento, modularización, escalabilidad y un uso muy extendido a nivel mundial, lo que permitirá posibles futuros desarrollos sobre la misma base de código.

La información de las guías de laboratorio se guarda en Firebase, que es un servicio de Google para bases de datos en tiempo real NoSQL. Es un registro de llave valor y es ideal para el tipo de información que se quiere registrar. Esta tecnología permite acceder a las actualizaciones en tiempo real, desde cualquier lugar y de manera eficiente.

7.5. Arquitectura general

Se plantea una arquitectura general, que consta de los módulos principales Estudiante y Docente. Estos dos módulos representan las pantallas de la aplicación y a su vez, el tipo de usuario hará uso de estos. En la figura 7.2 se encuentra un bosquejo de dicha arquitectura.

7.5.1. Módulo docente

Es el módulo de administración del sistema y solamente el docente tiene acceso a través de una contraseña.

Estando dentro, es posible realizar el cambio de contraseña, además de poder redactar las prácticas de laboratorio.

7.5.2. Módulo estudiante

Es el módulo principal de la aplicación y presenta un entorno que guía al estudiante a través de la estandarización y los métodos de Warder y Winkler. También permite consultar las guías de laboratorio.

7.5.3. Módulo redactar

Permite al docente la redacción de las guías de laboratorio. Estas se redactan basadas en las reacciones correspondientes a cada método, y para la introducción de la información se cuenta con un editor que permite agregar texto y contenido multimedia como fotos y videos. Esta información se guarda en tiempo real y será consultada en su versión actualizada cuando el estudiante requiera revisar las guías de laboratorio.

7.5.4. Módulo estandarización

Es el primer módulo presentado al estudiante. En este se guía en el proceso de la obtención de un estándar para el reactivo que se usará durante la práctica.

Permite elegir entre estandarizar un ácido o una base, después el estudiante introducirá los resultados experimentales obtenidos en los campos correspondientes de cada

alícuota y al final le presentará un cálculo promedio que será usado durante la práctica posterior.

7.5.5. Módulo método

Es el módulo de entrada al desarrollo de su práctica de laboratorio. Permite elegir entre los métodos de Warder o Winkler y luego presenta las posibles reacciones que pueden ser trabajadas con cada uno.

7.5.6. Módulo reacción

Guía al estudiante a través de una práctica específica de laboratorio. Consta normalmente de los pasos base de balancear una reacción y recolectar información de alícuotas, que pueden presentarse varias veces, dependiendo de las necesidades del método y de la reacción en sí.

Su función principal es presentar durante cada etapa de la práctica las alícuotas y los datos que se espera obtener para cada una. Después se introducirán los datos experimentales, la aplicación hará el cálculo independiente para cada alícuota y presentará un promedio de todas.

7.5.7. Módulo ver guía

Es un módulo transversal y permite consultar la guía de laboratorio. Esta puede presentarse de forma general o para reacciones específicas. Se presenta como una ventana emergente y muestra al estudiante el procedimiento descrito por el docente, junto con los recursos multimedia que lo acompañen.

Capítulo 8

Resultado de la implementación

Se llevó a cabo la implementación de la solución, haciendo uso de las tecnologías propuestas. Esto permite que como resultado se tenga una aplicación web desarrollada en ReactJS y base de datos Firebase.

La aplicación cuenta con un diseño responsive, que implica que esta se puede adaptar a diferentes tamaños de pantalla de forma automática. Esto significa que puede ser usada tanto en computadores, como en dispositivos móviles; permitiendo que sea óptima para ser usada durante el desarrollo de las prácticas de laboratorio.

La aplicación es del tipo conocido como serverless (sin servidor). Este tipo de aplicaciones se hospedan en un servidor web, cuyo uso es ser consultado únicamente para lanzar la aplicación. Al tener esta característica, la aplicación tiene unos requisitos de hardware muy bajos, pudiendo ser hospedada en servicios públicos especializados en aplicaciones de este tipo.

Se cuenta con un menú de usuario, que guía en el uso de cada uno de los módulos, haciendo uso de capturas de pantalla y descripciones apropiadas para cada sección.

8.1. Futuros proyectos que se pueden derivar

Se propone que la implementación actual pueda ser extendida a través de dos enfoques que pueden llevarse a cabo de forma complementaria:

- **Adaptación a una aplicación móvil nativa:** Esto permitirá que la aplicación se ejecute potencialmente con mayor eficiencia y compatibilidad con los dispositivos. Además de agregarle la capacidad de ser consultada sin conexión y actualizando su información a través de internet, cuando este se encuentre disponible.

  Esta adaptación permitirá además dotar a la aplicación de características propias de los dispositivos móviles como: cámara, GPS, etc. Elementos que pueden enriquecer en gran medida la experiencia de uso de la aplicación.

- **Implementación de otros métodos experimentales:** Es conocido que durante la asignatura de Química Analítica se tratan métodos adicionales a las volumetrías de neutralización, como puede ser el análisis gravimétrico.

  La arquitectura de la aplicación permite la adición de nuevos métodos experimentales, que pueden ampliar el espectro de prácticas aplicables y de esta manera, en un futuro permitir unificar las prácticas y guías de la asignatura.

Capítulo 9

Bibliografía

[1]Schwaber, K., Sutherland, E. La Guía de Scrum. 2016.

[2]Miles, R., Hamilton, K. Learning UML 2.0. 2006.

[3]Lockhart, J. Modern PHP: new features and good practices. 2016.

[4]Fain, Y., Moiseev, A. Angular 2 development with TypeScript. 2017.

[5]Skoog, D. A., West, D. M., Crouch, S. R., Holler, F. J. Fundamentos de quimica analitica. Mexico: Cengage Learning. 2010.

[6]FACULTAD DE CIENCIAS AGRARIAS – U.N.C. Cátedra de Química Analítica.

[7]Ospina, G. A., García, J. J., Martínez, P. N. Gravimetría y volumetríaca. 2010.

[8]Laitinen, H. A., Harris, W.E. Análisis Químico. 1982.

Apéndice A

Anexo: Código fuente

Algoritmo A.1: index.js

```
import React from 'react';
import Router from './components/Router';
import { render } from 'react-dom';
import { library } from '@fortawesome/fontawesome-svg-core';
import {
  faCheck,
  faLongArrowAltRight
} from '@fortawesome/free-solid-svg-icons';
import { faSquare as faSquareR } from '@fortawesome/free-
   regular-svg-icons';
import 'bootstrap/dist/css/bootstrap.min.css';
import './css/index.css';
import 'react-dropdown/style.css';
import * as serviceWorker from './serviceWorker';

library.add(faCheck, faLongArrowAltRight, faSquareR);

render(<Router />, document.getElementById('root'));
serviceWorker.unregister();
```

Algoritmo A.2: components/Router.js

```
import React, { useState } from 'react';
```

```
import { BrowserRouter, Route, Switch, Redirect } from 'react-
    router-dom';
import App from './App';
import Admin from './Admin';
import NotFound from './NotFound';
import Navbar from './Navbar';
import Login from './Login';

const UserContext = React.createContext();

function Router() {
  const [logged, setLogged] = useState(false);

  return (
    <BrowserRouter>
      <div>
        <UserContext.Provider value={logged}>
          <Navbar setLogged={setLogged} />
          <Switch>
            <Route exact path="/" component={App} />
            <PrivateRoute
              exact
              path="/admin"
              logged={logged}
              component={Admin}
            />
            <Route
              exact
              path="/login"
              component={() => <Login setLogged={setLogged}
                  />}
            />
            <Route component={NotFound} />
          </Switch>
        </UserContext.Provider>
      </div>
```

```
    </BrowserRouter>
  );
}

function PrivateRoute({ component: Component, logged, ...rest
   }) {
  return (
    <Route
      {...rest}
      render={props =>
        logged ? (
          <Component {...props} />
        ) : (
          <Redirect
            to={{
              pathname: '/login',
              state: { from: props.location }
            }}
          />
        )
      }
    />
  );
}

export default Router;
export { UserContext };
```

Algoritmo A.3: components/Admin.js

```
/* eslint-disable jsx-a11y/anchor-is-valid */
import React from 'react';
import Guide from './guide/Guide';
import '../css/Admin.css';

export default function Admin() {
  return (
    <div className="container ">
```

```
      <Guide />
    </div>
  );
}
```

Algoritmo A.4: components/App.js

```
import React, { useState, useEffect } from 'react';
import Standardization from './standardization/Standardization
  ';
import '../css/App.css';
import Warder from './warder/Warder';
import Winkler from './winkler/Winkler';
import base from '../base';
import ShowGuide from './guide/ShowGuide';
import ShowReaction from './guide/ShowReaction';
import { Collapse } from 'reactstrap';

export default function App() {
  const [chapters, setChapters] = useState({});
  const [reaction, setReaction] = useState({});
  const [openGuide, setOpenGuide] = useState(false);
  const [openReaction, setOpenReaction] = useState(false);
  const [cards, setCards] = useState({
    acid: false,
    base: false,
    warder: false,
    winkler: false
  });

  const toggleGuide = () => {
    setOpenGuide(open => !open);
  };

  const toggleCard = card_id => {
    setCards(cards => {
      const newCards = { ...cards };
      Object.keys(newCards).forEach(k => {
```

```
      newCards[k] = k === card_id ? !newCards[k] : false;
    });
    return newCards;
  });
};

const setReactionId = (chapter_id, reaction_id) => {
  setReaction(
    Object.values(chapters[chapter_id].reactions).find(
      r => r.id === reaction_id
    )
  );
  toggleReaction();
};

const toggleReaction = () => {
  setOpenReaction(open => !open);
};

useEffect(() => {
  base
    .ref('/chapters')
    .once('value')
    .then(function(data) {
      setChapters(data.val());
    });
}, []);

return (
  <div className="container">
    <ShowGuide chapters={chapters} toggle={toggleGuide} open
       ={openGuide} />
    <ShowReaction
      reaction={reaction}
      toggle={toggleReaction}
      open={openReaction}
```

```
/>
<button
  style={{
    position: 'fixed',
    right: '1em',
    bottom: '1em',
    zIndex: 100
  }}
  className="btn btn-outline btn-primary"
  onClick={toggleGuide}
>
  Gu a completa
</button>
<h2>Estandarizaci n</h2>
<Standardization
  cards={cards}
  toggleCard={toggleCard}
  setReactionId={setReactionId}
/>
<h2 className="mt-5">M todos</h2>
<div className="card">
  <div
    className="card-header"
    style={{ cursor: 'pointer', fontWeight: 600 }}
    onClick={() => toggleCard('warder')}
  >
    <span>Warder</span>
  </div>
  <Collapse isOpen={cards.warder}>
    <div className="card-body">
      <Warder setReactionId={setReactionId} />
    </div>
  </Collapse>
</div>

<div className="card">
```

```
          <div
            className="card-header"
            style={{ cursor: 'pointer', fontWeight: 600 }}
            onClick={() => toggleCard('winkler')}
          >
            <span>Winkler</span>
          </div>
          <Collapse isOpen={cards.winkler}>
            <div className="card-body">
              <Winkler setReactionId={setReactionId} />
            </div>
          </Collapse>
        </div>
      </div>
    );
}
```

Algoritmo A.5: components/Balancer.js

```
import React, { useState, useEffect } from 'react';
import { splitReaction, random } from '../helpers';
import { FontAwesomeIcon } from '@fortawesome/react-
    fontawesome';
import Molecule from './render/Molecule';

function NumberChooser({ pair, changeNumber, balanced }) {
  const [number, setNumber] = useState(pair[1]);

  const numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9];

  useEffect(
    function() {
      changeNumber(pair, number);
    },
    [number]
  );

  return (
```

```
    <select
      className="form-control form-control-sm"
      disabled={balanced}
      value={number}
      style={{ width: '50px', display: 'inline-block' }}
      onChange={event => setNumber(parseInt(event.target.value
        ))}
    >
      {numbers.map(n => (
        <option key={n}>{n}</option>
      ))}
    </select>
  );
}

export default function ReactionBalancer(props) {
  const [pairs, setPairs] = useState([]);
  const [reaction, setReaction] = useState([]);
  const is_test = false;

  useEffect(() => {
    setPairs([]);
    setReaction(splitReaction(props.reaction));
  }, [props.reaction]);

  useEffect(() => {
    let pairs = [];
    reaction.forEach(side => {
      side.forEach(molecule => {
        let pair;
        if (is_test) {
          pair = [1, 1];
          if (Number.isInteger(molecule[0]))
            pair[0] = pair[1] = molecule.splice(0, 1)[0];
        } else {
          pair = [1, random(1, 10)];
```

```
        if (Number.isInteger(molecule[0])) pair[0] =
          molecule.splice(0, 1)[0];
      }
      pairs.push(pair);
    });
  });
  setPairs(pairs);
}, [is_test, reaction]);

useEffect(() => {
  props.balance(!pairs.find(p => p[0] !== p[1]));
}, [pairs, props]);

const changeNumber = (pair, number) => {
  setPairs(pairs => {
    const newPairs = [...pairs];
    newPairs[pairs.indexOf(pair)][1] = number;
    return newPairs;
  });
};

return (
  <>
    <span style={{ fontSize: '1rem' }} className="mr-2">
      {reaction.map((side, j, reaction) => (
        <React.Fragment key={j}>
          {side.map((molecule, i) => (
            <React.Fragment key={i}>
              <span style={{ marginRight: '5px' }}>
                {pairs.length && (
                  <NumberChooser
                    balanced={props.balanced}
                    pair={pairs[j * reaction.length + i]}
                    changeNumber={changeNumber}
                  />
                )}
```

```
                </span>

                <Molecule molecule={molecule} />
                {i < side.length - 1 && <span className="mx-2"
                  >+</span>}
              </React.Fragment>
            ))}
            {j < reaction.length - 1 && (
              <FontAwesomeIcon
                icon="long-arrow-alt-right"
                className="fa-fw mx-2"
              />
            )}
          </React.Fragment>
        ))}
      </span>
    </>
  );
}
```

Algoritmo A.6: components/Login.js

```
import React, { useState } from 'react';
import '../css/Login.css';
import logo from '../images/logo.png';
import { Link } from 'react-router-dom';
import sha256 from 'js-sha256';
import base from '../base';
import { withRouter } from 'react-router-dom';

const Login = withRouter(({ history, setLogged }) => {
  const [password, setPassword] = useState('');

  const login = () => {
    const user_password = sha256(password);
    base
      .ref('password')
      .once('value')
```

```
    .then(data => {
      const server_password = data.val();
      if (server_password) {
        if (server_password === user_password) {
          setLogged(true);
          history.push('/admin');
        } else {
          alert('Contrase a incorrecta');
        }
      } else {
        base.ref('password').set(user_password);
      }
    });
};

return (
  <div className="text-center body">
    <div className="form-signin">
      <img className="mb-4" src={logo} alt="" height="256"
        />
      <h1 className="h3 mb-3 font-weight-normal">QAC-VN</h1>
      <label htmlFor="inputPassword" className="sr-only">
        Contrase a
      </label>
      <input
        type="password"
        id="inputPassword"
        className="form-control"
        placeholder="Contrase a"
        value={password}
        onChange={event => setPassword(event.target.value)}
      />
      <button
        className="btn btn-lg btn-secondary btn-block"
        onClick={login}
        disabled={!password}
```

```
          >
            Docente
          </button>
          <br />
          <Link to="/" className="btn btn-lg btn-primary btn-
              block">
            Estudiante
          </Link>
        </div>
      </div>
    );
});

export default Login;
```

Algoritmo A.7: components/Mean.js

```
/* eslint-disable jsx-a11y/anchor-is-valid */
import React, { useState, useEffect } from 'react';
import Reaction from './render/Reaction';
import { units } from '../helpers';
import nanoid from 'nanoid';

export default function Mean(props) {
  const [mean, setMean] = useState(0);
  const [measurements, setMeasurements] = useState({});
  const [measurements_t, setMeasurementsT] = useState({});

  const getNew = function() {
    const object = {};
    props.measures.forEach(measure => {
      object[measure[0]] = 0;
    });
    return object;
  };

  useEffect(() => {
    for (let i = 0; i < 3; i++) {
```

```
    addMeasurement();
  }
  // eslint-disable-next-line react-hooks/exhaustive-deps
}, []);

const addMeasurement = () => {
  setMeasurements(m => ({ ...m, [nanoid()]: getNew() }));
};

const removeMeasurement = k => {
  setMeasurements(m => {
    const newM = { ...m };
    delete newM[k];
    return newM;
  });
};

const setMeasurement = (k, field, event) => {
  const value = event.target.value ? parseFloat(event.target
    .value) : 0;
  setMeasurements(m => {
    const measurement = { ...m[k] };
    measurement[field] = value;
    return { ...m, [k]: measurement };
  });
};

const getMean = () => {
  let m = 0;
  const newMeasurements = {};
  Object.entries(measurements).forEach(([k, measurement]) =>
    {
    const val = props.formula(measurement);
    m += val;

    newMeasurements[k] = val;
```

```
    });

    setMeasurementsT(newMeasurements);

    const length = Object.values(measurements).filter(
      measurement => Object.values(measurement).indexOf(0) < 0
    ).length;
    if (length) setMean(m / length);
    if (props.q) props.q(Object.values(measurements).length >=
        3);
    if (props.setMean) props.setMean(mean);
  };

  useEffect(getMean, [props.external]);

  useEffect(getMean, [measurements]);

  return (
    <>
      <div className="row mt-4">
        <div className="col">
          <h5>
            <b>Promedio:</b> {mean.toFixed(4)} {props.result
                [0]} de{' '}
            <Reaction reaction={props.result[1]} />
          </h5>
        </div>
      </div>
      <br />
      <div className="row mt-2">
        {Object.entries(measurements).map(([k, m], j) => (
          <div className="col-md-6 mb-3" key={k}>
            <div className="card">
              <div className="card-header">
                <button
                  type="button"
```

```
        className="close"
        onClick={() => removeMeasurement(k)}
      >
        <span aria-hidden="true">&times;</span>
      </button>
      <b>Al cuota {j + 1}</b>
    </div>
    <div className="card-body">
      {props.measures.map(([v, unit, mol], i) => (
        <div className="row mb-1" key={i}>
          <div className="col-md-6 font-weight-bold"
              >
            {units[unit].prefix} de <Reaction
                reaction={mol} />{' '}
            {units[unit].units && ` (${units[unit].
                units})`}
          </div>
          <div className="col-md-6 mb-2">
            <input
              type="number"
              className="form-control form-control-
                  sm"
              value={m[v]}
              onChange={event => setMeasurement(k, v
                  , event)}
            />
          </div>
        </div>
      ))}
    </div>
    <div className="card-footer">
      {(measurements_t[k] ? measurements_t[k] : 0).
          toFixed(4)}{' '}
      {props.result[0]} de <Reaction reaction={props
          .result[1]} />
    </div>
```

```
            </div>
          </div>
        ))}
        <div className="col-sm-6 align-self-center">
          <button
            className="btn btn-secondary btn-block"
            onClick={addMeasurement}
          >
            Agregar medici n
          </button>
        </div>
      </div>
    </>
  );
}
```

Algoritmo A.8: components/Navbar.js

```
import React, { useContext } from 'react';
import { UserContext } from './Router';
import { NavLink, withRouter } from 'react-router-dom';

const Navbar = withRouter(({ history, setLogged }) => {
  const logged = useContext(UserContext);

  const logout = () => {
    setLogged(false);
    history.push('/login');
  };

  return (
    <nav className="navbar navbar-expand-lg navbar-light bg-
        light mb-4">
      <span className="navbar-brand mb-0 h1">QAC-VN</span>
      <div className="collapse navbar-collapse" id="
          navbarSupportedContent">
        {logged && (
          <ul className="navbar-nav mr-auto">
```

```
            <li className="nav-item">
              <NavLink exact to="/" className="nav-link">
                Estudiante
              </NavLink>
            </li>
            <li className="nav-item">
              <NavLink exact to="/admin" className="nav-link">
                Docente
              </NavLink>
            </li>
          </ul>
        )}
      </div>
      <ul className="navbar-nav mr-auto">
        <li className="nav-item">
          <a href="#" onClick={logout}>
            Salir
          </a>
        </li>
      </ul>
    </nav>
  );
});

export default Navbar;
```

Algoritmo A.9: components/NotFound.js

```
import React from 'react';

const NotFound = () => (
  <div>
    <h2>Not Found!!!</h2>
  </div>
);

export default NotFound;
```

Algoritmo A.10: base.js

```
import { initializeApp } from 'firebase';

const firebaseApp = initializeApp({
  apiKey: '',
  authDomain: 'analisis-cuantitativo.firebaseapp.com',
  databaseURL: 'https://analisis-cuantitativo.firebaseio.com'
});

const base = firebaseApp.database();

export { firebaseApp };

export default base;
```

Algoritmo A.11: helpers.js

```
export const units = {
  vol: {
    prefix: 'Vol.',
    units: 'L'
  },
  pm: {
    prefix: 'PM'
  },
  peso: {
    prefix: 'Gramos'
  },
  moles: {
    prefix: 'moles'
  },
  molaridad: {
    prefix: 'M'
  }
};

export function formatNumber(number) {
  return number.toFixed(4);
```

```
}

export function getNumber(number) {
  return number && number > 1 && number;
}

export function capitalize(str) {
  return str.charAt(0).toUpperCase() + str.slice(1);
}

export function isLowerCase(char) {
  return char && char.toLowerCase() === char;
}

export function isNumber(string) {
  return string && Number.isInteger(parseInt(string));
}

export function splitReaction(string) {
  return string.split('=').map(side => side.split('+').map(
     splitMolecule));
}

export function random(min, max) {
  return Math.floor(Math.random() * (max - min)) + min;
}

export function splitMolecule(string) {
  const ret = [];
  const array = string.split('');
  let temp = null,
    i = 0,
    length = array.length;

  const add = (e, bool) => {
    if (bool) {
```

```
      temp += e;
    } else {
      ret.push(isNumber(temp) ? parseInt(temp) : temp);
      temp = e;
    }
  };

  while (i <= length) {
    let e = array[i];
    if (temp) {
      isNumber(e) ? add(e, isNumber(temp)) : add(e,
        isLowerCase(e));
    } else {
      temp = e;
    }
    i++;
  }
  return ret;
}
```

Algoritmo A.12: components/guide/Chapter.js

```
import React, { useState, useEffect } from 'react';
import CKEditor from '@ckeditor/ckeditor5-react';
import ClassicEditor from '@ckeditor/ckeditor5-build-classic';
import Reaction from '../render/Reaction';
import { Collapse } from 'reactstrap';

export default function Chapter(props) {
  const [chapter, setChapter] = useState(null);

  useEffect(() => {
    setChapter({ ...props.chapter });
  }, [props.chapter]);

  const toggleReaction = reaction_id => {
    const newChapter = { ...chapter };
    Object.entries(chapter.reactions).forEach(([id, reaction])
```

```
      => {
    reaction.is_open = id === reaction_id ? !reaction.
      is_open : false;
  });

  setChapter(newChapter);
};

const setReaction = (value, reaction_id) => {
  const reaction = { ...chapter.reactions[reaction_id] };

  if (reaction.guide === value) return;

  reaction.guide = value;

  props.setChapter(props.chapter_id, {
    ...chapter,
    reactions: { ...chapter.reactions, [reaction_id]:
      reaction }
  });
};

return (
  <>
    {chapter &&
      Object.entries(chapter.reactions)
        .sort((a, b) => a[1].order - b[1].order)
        .map(([key, reaction]) => (
          <div className="card" key={key}>
            <div className="card-header">
              <button
                className="btn btn-link"
                onClick={() => toggleReaction(key)}
              >
                <Reaction reaction={reaction.label} />
              </button>
```

```
              </div>
              <Collapse isOpen={reaction.is_open}>
                <div className="card-body">
                  <CKEditor
                    editor={ClassicEditor}
                    config={{
                      mediaEmbed: {
                        previewsInData: true
                      }
                    }}
                    data={reaction.guide}
                    onChange={(_event, editor) =>
                      setReaction(editor.getData(), key)
                    }
                  />
                </div>
              </Collapse>
            </div>
          ))}
    </>
  );
}
```

Algoritmo A.13: components/guide/Guide.js

```
import React, { useState, useEffect } from 'react';
import Chapter from './Chapter';
import base from '../../base';
import nanoid from 'nanoid';

export default function Guide() {
  const [chapter_id, setChapterId] = useState();
  const [chapters, setChapters] = useState(null);

  const getDefaultChapters = () => {
    const default_chapters = {
      standard: {
        label: 'Estandarizaci n',
```

```
    reactions: [
      {
        id: 'acid',
        label: ' cido '
      },
      {
        id: 'base',
        label: 'Base'
      }
    ]
  },
  warder: {
    label: 'Warder',
    reactions: [
      {
        id: 'naoh',
        label: 'NaOH'
      },
      {
        id: 'nahco3',
        label: 'NaHCO3'
      },
      {
        id: 'na2co3',
        label: 'Na2CO3'
      },
      {
        id: 'naohpna2co3',
        label: 'NaOH+Na2CO3'
      },
      {
        id: 'nahco3pna2co3',
        label: 'NaHCO3+Na2CO3'
      }
    ]
  },
```

```
    winkler: {
      label: 'Winkler',
      reactions: [
        {
          id: 'naohpna2co3',
          label: 'NaOH+Na2CO3'
        },
        {
          id: 'nahco3pna2co3',
          label: 'NaHCO3+Na2CO3'
        }
      ]
    }
  };

  Object.values(default_chapters).forEach(chapter => {
    let reactions = {};
    chapter.reactions.forEach((reaction, order) => {
      reactions[nanoid()] = { ...reaction, guide: '', order
        };
    });

    chapter.reactions = reactions;
  });

  return default_chapters;
};

useEffect(() => {
  base
    .ref('chapters')
    .once('value')
    .then(snapshot => {
      if (snapshot.val()) setChapters(snapshot.val());
      else setChapters(getDefaultChapters());
    });
```

```
}, []);

const setChapter = (chapter_id, chapter) => {
  setChapters(chapters => {
    const new_chapters = { ...chapters, [chapter_id]:
       chapter };
    base.ref('chapters').set(new_chapters);
    return new_chapters;
  });
};

return (
  <>
    <div className="row">
      <div className="col-4">
        <div className="list-group">
          {chapters &&
            Object.entries(chapters).map(([key, chapter]) =>
              (
              <button
                key={key}
                type="button"
                className={
                  'list-group-item list-group-item-action '
                    +
                  (key === chapter_id && 'active')
                }
                onClick={() => setChapterId(key)}
              >
                {chapter.label}
              </button>
            ))}
        </div>
      </div>
      <div className="col-8">
        {chapter_id && (
```

```
            <Chapter
              chapter_id={chapter_id}
              chapter={chapters[chapter_id]}
              setChapter={setChapter}
            />
          )}
        </div>
      </div>
    </>
  );
}
```

Algoritmo A.14: components/guide/ShowGuide.js

```
import React, { useState, useEffect } from 'react';
import { Button, Modal, ModalHeader, ModalBody, ModalFooter }
   from 'reactstrap';
import Reaction from '../render/Reaction';
import { Collapse } from 'reactstrap';

export default function ShowGuide(props) {
  const [chapters, setChapters] = useState();

  useEffect(() => {
    setChapters({ ...props.chapters });
  }, [props.chapters]);

  const toggleReaction = (chapter_id, reaction_id) => {
    const newChapter = { ...chapters[chapter_id] };
    Object.entries(newChapter.reactions).forEach(([id,
       reaction]) => {
      reaction.is_open = id === reaction_id ? !reaction.
         is_open : false;
    });

    setChapters({ ...chapters, [chapter_id]: newChapter });
  };
```

```
return (
  <div>
    <Modal isOpen={props.open} toggle={props.toggle} size="
        lg">
      <ModalHeader toggle={props.toggle}>Gu a</ModalHeader>
      <ModalBody>
        {chapters &&
          Object.entries(chapters).map(([chapter_id, chapter
              ]) => (
            <React.Fragment key={chapter_id}>
              <h3>
                <Reaction reaction={chapter.label} />
              </h3>
              <br />
              {chapter.reactions &&
                Object.entries(chapter.reactions)
                  .sort((a, b) => a[1].order - b[1].order)
                  .map(([key, reaction]) => (
                    <div className="card" key={key}>
                      <div
                        className="card-header"
                        onClick={() => toggleReaction(
                            chapter_id, key)}
                        style={{ cursor: 'pointer',
                            fontWeight: 600 }}
                      >
                        <Reaction reaction={reaction.label}
                            />
                      </div>
                      <Collapse isOpen={reaction.is_open}>
                        <div
                          className="card-body"
                          dangerouslySetInnerHTML={{ __html:
                              reaction.guide }}
                        />
                      </Collapse>
```

```
                    </div>
                  ))}
                <br />
                <br />
              </React.Fragment>
            ))}
        </ModalBody>
        <ModalFooter>
          <Button color="secondary" onClick={props.toggle}>
            Cerrar
          </Button>
        </ModalFooter>
      </Modal>
    </div>
  );
}
```

Algoritmo A.15: components/guide/ShowReaction.js

```
import React from 'react';
import { Button, Modal, ModalHeader, ModalBody, ModalFooter }
   from 'reactstrap';
import Reaction from '../render/Reaction';

export default function ShowReaction(props) {
  return (
    <div>
      <Modal isOpen={props.open} toggle={props.toggle} size="
          lg">
        <ModalHeader toggle={props.toggle}>
          <Reaction reaction={props.reaction.label} />
        </ModalHeader>
        <ModalBody>
          <div
            className="card-body"
            dangerouslySetInnerHTML={{ __html: props.reaction.
                guide }}
          />
```

```
        </ModalBody>
        <ModalFooter>
          <Button color="secondary" onClick={props.toggle}>
            Cerrar
          </Button>
        </ModalFooter>
      </Modal>
    </div>
  );
}
```

Algoritmo A.16: components/render/Molecule.js

```
import React from 'react';
import { splitMolecule } from '../../helpers';
import { FontAwesomeIcon } from '@fortawesome/react-
    fontawesome';

export default function Molecule(props) {
  return (
    <>
      {(typeof props.molecule === 'string'
        ? splitMolecule(props.molecule)
        : props.molecule
      ).map((e, i) => (
        <React.Fragment key={i}>
          {i === 0 && props.hideC && (
            <FontAwesomeIcon
              className="text-danger fa-fw"
              icon={['far', 'square']}
            />
          )}
          {Number.isInteger(e) && i !== 0 ? <sub>{e}</sub> : <
              span>{e}</span>}
        </React.Fragment>
      ))}
    </>
  );
```

```
}
```

Algoritmo A.17: components/render/Reaction.js

```
import React from 'react';
import Molecule from './Molecule';
import { FontAwesomeIcon } from '@fortawesome/react-
   fontawesome';
import { splitReaction } from '../../helpers';

export default function Reaction(props) {
  return (
    <>
      {splitReaction(props.reaction).map((side, j, reaction)
         => (
        <React.Fragment key={j}>
          {side.map((molecule, i) => (
            <React.Fragment key={i}>
              <Molecule molecule={molecule} hideC={props.hideC
                 } />
              {i < side.length - 1 && <span className="mx-2"
                 >+</span>}
            </React.Fragment>
          ))}
          {j < reaction.length - 1 && (
            <FontAwesomeIcon
              icon="long-arrow-alt-right"
              className="fa-fw mx-1"
            />
          )}
        </React.Fragment>
      ))}
    </>
  );
}
```

Algoritmo A.18: components/standardization/Acid.js

```
import React from 'react';
```

```
import Balancer from '../Balancer';
import Mean from '../Mean';

export default function Acid(props) {
  const measures = [
    ['p', 'peso', 'Na2CO3'],
    ['pm', 'pm', 'Na2CO3'],
    ['v', 'vol', 'HCl']
  ];
  const formula = ({ p, pm, v }) => {
    if (p && pm && v) return (2 * (p / pm)) / v;
    return 0;
  };
  return (
    <>
      <div className="row">
        <div className="col-12">
          <Balancer {...props} />
        </div>
      </div>
      {props.balanced && (
        <Mean result={['M', 'HCl']} measures={measures}
          formula={formula} />
      )}
    </>
  );
}
```

Algoritmo A.19: components/standardization/Base.js

```
import React from 'react';
import Balancer from '../Balancer';
import Mean from '../Mean';

export default function Base(props) {
  const measures = [
    ['p', 'peso', 'C8H5KO4'],
    ['pm', 'pm', 'C8H5KO4'],
```

```
    ['v', 'vol', 'NaOH']
  ];
  const formula = ({ p, pm, v }) => {
    if (p && pm && v) return p / pm / v;
    return 0;
  };
  return (
    <>
      <div className="row">
        <div className="col-12">
          <Balancer {...props} />
        </div>
      </div>
      {props.balanced && (
        <Mean result={['M', 'NaOH']} measures={measures}
            formula={formula} />
      )}
    </>
  );
}
```

Algoritmo A.20: components/standardization/Standardization.js

```
import React, { useState } from 'react';
import Acid from './Acid';
import Base from './Base';
import Reaction from '../render/Reaction';
import { Collapse } from 'reactstrap';

export default function Standardization(props) {
  const [acid_balanced, setAcidBalanced] = useState(false);
  const [base_balanced, setBaseBalanced] = useState(false);

  const acid = 'Na2CO3+2HCl=2NaCl+CO2+H2O';
  const base = 'C8H5KO4+NaOH=NaC8H4KO4+H2O';

  return (
    <>
```

```
<div className="card border-danger">
  <div
    className="card-header bg-danger"
    style={{ cursor: 'pointer', fontWeight: 600 }}
    onClick={() => props.toggleCard('acid')}
  >
    <span className="text-white"> cido </span>
  </div>
  <Collapse isOpen={props.cards.acid}>
    <div className="card-body">
      <div className="row">
        <div className="col">
          <button
            className="btn btn-outline-secondary pull-
                right"
            onClick={() => props.setReactionId('standard
                ', 'acid')}
          >
            Gu a
          </button>
        </div>
      </div>
      <br />
      <Acid
        reaction={acid}
        balanced={acid_balanced}
        balance={setAcidBalanced}
      />
    </div>
  </Collapse>
</div>
<div className="card border-info">
  <div
    className="card-header bg-info"
    style={{ cursor: 'pointer', fontWeight: 600 }}
    onClick={() => props.toggleCard('base')}
```

```
          >
            <span className="text-white">Base</span>
          </div>
          <Collapse isOpen={props.cards.base}>
            <div className="card-body">
              <div className="row">
                <div className="col">
                  <button
                    className="btn btn-outline-secondary pull-
                       right"
                    onClick={() => props.setReactionId('standard
                       ', 'base')}
                  >
                    Gu a
                  </button>
                </div>
              </div>
              <br />
              <Base
                reaction={base}
                balanced={base_balanced}
                balance={setBaseBalanced}
              />
            </div>
          </Collapse>
        </div>
      </>
    );
}
```

Algoritmo A.21: components/warder/Na2CO3.js

```
import React from 'react';
import Mean from '../Mean';

export default function Na2CO3(props) {
  const measures = [
    ['v', 'vol', 'HCl'],
```

```
    ['m', 'molaridad', 'HCl'],
    ['vl', 'vol', 'Muestra']
  ];
  const formula = ({ v, m, vm }) => {
    if (v && m && vm) return (v * m) / 2 / vm;
    return 0;
  };

  return (
    <Mean
      result={['M', 'Na2CO3']}
      measures={measures}
      formula={formula}
      q={props.q}
    />
  );
}
```

Algoritmo A.22: components/warder/NaHCO3.js

```
import React from 'react';
import Mean from '../Mean';

export default function NaHCO3(props) {
  const measures = [
    ['v', 'vol', 'HCl'],
    ['m', 'molaridad', 'HCl'],
    ['vm', 'vol', 'Muestra']
  ];
  const formula = ({ v, m, vm }) => {
    if (v && m && vm) return (v * m) / vm;
    return 0;
  };

  return (
    <Mean
      result={['M', 'NaOH']}
      measures={measures}
```

```
      formula={formula}
      q={props.q}
    />
  );
}
```

Algoritmo A.23: components/warder/NaOH.js

```
import React from 'react';
import Mean from '../Mean';

export default function NaOH(props) {
  const measures = [
    ['v', 'vol', 'HCl'],
    ['m', 'molaridad', 'HCl'],
    ['vm', 'vol', 'Muestra']
  ];
  const formula = ({ v, m, vm }) => {
    if (v && m && vm) return (v * m) / vm;
    return 0;
  };

  let result = 'M';

  if (props.label)
    result = (
      <span>
        M <sub>totales</sub>
      </span>
    );

  return (
    <Mean
      result={[result, 'NaOH']}
      measures={measures}
      formula={formula}
      q={props.q}
      setMean={props.setMean}
```

```
    />
  );
}
```

Algoritmo A.24: components/warder/Warder.js

```
import React, { useState, useEffect } from 'react';
import Reaction from '../render/Reaction';
import Dropdown from 'react-dropdown';
import ReactionBalancer from '../Balancer';
import NaOH from './NaOH';
import NaHCO3 from './NaHCO3';
import Na2CO3 from './Na2CO3';

export default function Warder(props) {
  const warder = [
    ['naoh', 'NaOH', ['NaOH+HCl=NaCl+H2O']],
    ['nahco3', 'NaHCO3', ['NaHCO3+HCl=NaCl+CO2+H2O']],
    ['na2co3', 'Na2CO3', ['Na2CO3+2HCl=2NaCl+CO2+H2O']],
    [
      'naohpna2co3',
      'NaOH+Na2CO3',
      ['NaOH+HCl=NaCl+H2O', 'Na2CO3+2HCl=2NaCl+CO2+H2O']
    ],
    [
      'nahco3pna2co3',
      'NaHCO3+Na2CO3',
      ['NaHCO3+HCl=NaCl+CO2+H2O', 'Na2CO3+2HCl=2NaCl+CO2+H2O']
    ]
  ].map((r, i) => ({
    value: i,
    reactions: r[2].map(r => ({ reaction: r, balanced: false
      })),
    id: r[0],
    label: <Reaction reaction={r[1]} />
  }));

  const [reaction, setReaction] = useState(null);
```

```
const [reactions, setReactions] = useState([]);
const [reactions_balanced, setReactionsBalanced] = useState(
    false);

useEffect(() => {
  if (!reaction) return;
  setReactions(warder[reaction.value].reactions);
}, [reaction]);

useEffect(() => {
  setReactionsBalanced(reactions.length && !reactions.find(r
      => !r.balanced));
}, [reactions]);

const balance = (i, balanced) => {
  const r = [...reactions];
  r[i].balanced = balanced;
  setReactions(r);
};

const solveReaction = function() {
  if (!reactions_balanced) return null;
  switch (reaction.value) {
    case 0:
      return <NaOH />;
    case 1:
      return <NaHCO3 />;
    case 2:
      return <Na2CO3 />;
    case 3:
      return (
        <div className="row">
          <div className="col-lg">
            <NaOH />
          </div>
          <div className="col-lg">
```

```
            <Na2CO3 />
          </div>
        </div>
      );
    case 4:
      return (
        <div className="row">
          <div className="col-lg">
            <NaHCO3 />
          </div>
          <div className="col-lg">
            <Na2CO3 />
          </div>
        </div>
      );
    default:
      return;
  }
};

return (
  <div className="form-group">
    <label>Reacciones</label>
    <div className="input-group">
      <Dropdown
        className="react-dropdown"
        options={warder}
        value={reaction}
        onChange={setReaction}
        placeholder="Seleccionar..."
      />
      <div className="input-group-append">
        <button
          disabled={reaction === null}
          className="btn btn-outline-secondary"
          onClick={() =>
```

```
                props.setReactionId('warder', warder[reaction.
                    value].id)
              }
            >
              Gu a
            </button>
          </div>
        </div>
        <br />
        {reactions.map((r, i) => (
          <div className="mb-3" key={i}>
            <ReactionBalancer
              reaction={r.reaction}
              balanced={r.balanced}
              i={i}
              balance={balance.bind(this, i)}
            />
            <br />
          </div>
        ))}
        {solveReaction()}
      </div>
    );
}
```

Algoritmo A.25: components/winkler/NaHCO3pNa2CO3/Measurement.js

```
import React from 'react';
import { units } from '../../../helpers';
import Reaction from '../../render/Reaction';

export default function Measurement(props) {
  const measure = props.measures.find(m => m[0] === props.v);

  return (
    <div className="row mb-1">
      <div className="col-md-6 font-weight-bold">
        {units[measure[1]].prefix} de <Reaction reaction={
```

```
            measure[2]} />{' '}
          {units[measure[1]].units && ` (${units[measure[1]].
            units})`}
        </div>
        <div className="col-md-6 mb-4">
          <input
            type="number"
            className="form-control form-control-sm"
            value={props.m[props.v]}
            onChange={event => props.setMeasurement(props.k,
              props.v, event)}
          />
        </div>
      </div>
  );
}
```

Algoritmo A.26: components/winkler/NaHCO3pNa2CO3/Na2HCO3.js

```
import React, { useState, useEffect } from 'react';
import nanoid from 'nanoid';
import Reaction from '../../render/Reaction';
import Measurement from './Measurement';

export default function Na2HCO3(props) {
  const [mean, setMean] = useState(0);
  const measures = [
    ['v1', 'vol', 'HCl'],
    ['v2', 'vol', 'HCl'],
    ['vm', 'vol', 'Muestra'],
    ['moles3', 'moles', 'Na2CO3']
  ];
  const [measurements, setMeasurements] = useState({});
  const [measurements_t, setMeasurementsT] = useState({});

  const getNew = function() {
    const object = {};
    measures.forEach(measure => {
```

```
    object[measure[0]] = 0;
  });
  return object;
};

useEffect(() => {
  for (let i = 0; i < 3; i++) {
    addMeasurement();
  }
}, []);

const addMeasurement = () => {
  setMeasurements(m => ({ ...m, [nanoid()]: getNew() }));
};

const removeMeasurement = k => {
  setMeasurements(m => {
    const newM = { ...m };
    delete newM[k];
    return newM;
  });
};

const setMeasurement = (k, field, event) => {
  const value = event.target.value ? parseFloat(event.target
    .value) : 0;
  setMeasurements(m => {
    const measurement = { ...m[k] };
    measurement[field] = value;
    return { ...m, [k]: measurement };
  });
};

const getMean = () => {
  setMeasurementsT(() => {
    const newMeasurements = {};
```

```
    let mean = 0;

    Object.entries(measurements).forEach(([k, m]) => {
      let newM = {
        moles1: m.v1 * props.m1,
        moles2: m.v2 * props.m2
      };

      newM.m3 = m.vm ? m.moles3 / m.vm : 0;

      mean += newM.m3;

      newMeasurements[k] = newM;
    });

    mean = mean / Object.values(newMeasurements).length;

    setMean(mean);
    return newMeasurements;
  });
};

useEffect(getMean, [measurements, props.m1, props.m2]);

const getTotal = (k, v) => {
  return measurements_t[k] && measurements_t[k][v]
    ? measurements_t[k][v].toFixed(4)
    : 0;
};

const args = {
  measures,
  setMeasurement
};

return (
```

```
<>
  <div className="row mt-4">
    {Object.entries(measurements).map(([k, m], j) => (
      <div className="col-md-6 mb-4" key={k}>
        <div className="card">
          <div className="card-header">
            <button
              type="button"
              className="close"
              onClick={() => removeMeasurement(k)}
            >
              <span aria-hidden="true">&times;</span>
            </button>
            <b>Al cuota {j + 1}</b>
          </div>
          <div className="card-body">
            <Measurement k={k} {...args} v="vm" m={m} />
            <div className="row">
              <div className="col-md-6 offset-md-6">
                <span>
                  {getTotal(k, 'moles1')} moles <sub>
                    totales</sub> de{' '}
                  <Reaction reaction="HCl" />
                </span>
              </div>
            </div>
            <Measurement k={k} {...args} v="v1" m={m} />
            <div className="row">
              <div className="col-md-6 offset-md-6">
                <span>
                  {getTotal(k, 'moles2')} moles <sub>
                    reaccionan</sub> de{' '}
                  <Reaction reaction="HCl" />
                </span>
              </div>
            </div>
```

```
                <Measurement k={k} {...args} v="v2" m={m} />
                <Measurement k={k} {...args} v="moles3" m={m}
                    />
              </div>
              <div className="card-footer">
                <b>
                  {getTotal(k, 'm3')} M de <Reaction reaction=
                      "Na2CO3" />
                </b>
              </div>
            </div>
            <br />
          </div>
        ))}
        <div className="col-md-6 mb-4 align-self-center">
          <button
            className="btn btn-secondary btn-block"
            onClick={addMeasurement}
          >
            Agregar medici n
          </button>
        </div>
      </div>
      <h5>
        Promedio: {mean.toFixed(4)} M de <Reaction reaction="
            NaHCO3" />
      </h5>
      <br />
    </>
  );
}
```

Algoritmo A.27: components/winkler/NaHCO3pNa2CO3/NaHCO3.js

```
import React, { useState, useEffect } from 'react';
import nanoid from 'nanoid';
import Reaction from '../../render/Reaction';
import Measurement from './Measurement';
```

```
export default function NaHCO3(props) {
  const [mean, setMean] = useState(0);
  const measures = [
    ['v1', 'vol', 'NaOH'],
    ['vm', 'vol', 'Muestra'],
    ['v2', 'vol', 'HCl'],
    ['moles3', 'moles', 'NaHCO3']
  ];
  const [measurements, setMeasurements] = useState({});
  const [measurements_t, setMeasurementsT] = useState({});

  const getNew = function() {
    const object = {};
    measures.forEach(measure => {
      object[measure[0]] = 0;
    });
    return object;
  };

  useEffect(() => {
    for (let i = 0; i < 3; i++) {
      addMeasurement();
    }
  }, []);

  const addMeasurement = () => {
    setMeasurements(m => ({ ...m, [nanoid()]: getNew() }));
  };

  const removeMeasurement = k => {
    setMeasurements(m => {
      const newM = { ...m };
      delete newM[k];
      return newM;
    });
```

```
};

const setMeasurement = (k, field, event) => {
  const value = event.target.value ? parseFloat(event.target
    .value) : 0;
  setMeasurements(m => {
    const measurement = { ...m[k] };
    measurement[field] = value;
    return { ...m, [k]: measurement };
  });
};

const getMean = () => {
  setMeasurementsT(() => {
    const newMeasurements = {};
    let mean = 0;

    Object.entries(measurements).forEach(([k, m]) => {
      let newM = {
        moles1: m.v1 * props.m1,
        moles2: m.v2 * props.m2
      };

      newM.m3 = m.vm ? m.moles3 / m.vm : 0;

      mean += newM.m3;

      newMeasurements[k] = newM;
    });

    mean = mean / Object.values(newMeasurements).length;

    setMean(mean);
    return newMeasurements;
  });
};
```

```
useEffect(getMean, [measurements, props.m1, props.m2]);

const getTotal = (k, v) => {
  return measurements_t[k] && measurements_t[k][v]
    ? measurements_t[k][v].toFixed(4)
    : 0;
};

const args = {
  measures,
  setMeasurement
};

return (
  <>
    <div className="row mt-4">
      {Object.entries(measurements).map(([k, m], j) => (
        <div className="col-md-6 mb-4" key={k}>
          <div className="card">
            <div className="card-header">
              <button
                type="button"
                className="close"
                onClick={() => removeMeasurement(k)}
              >
                <span aria-hidden="true">&times;</span>
              </button>
              <b>Al cuota {j + 1}</b>
            </div>
            <div className="card-body">
              <Measurement k={k} {...args} v="vm" m={m} />
              <div className="row">
                <div className="col-md-6 offset-md-6">
                  <span>
                    {getTotal(k, 'moles1')} moles <sub>
```

```
                totales</sub> de{'␣'}
              <Reaction reaction="NaOH" />
            </span>
          </div>
        </div>

        <Measurement k={k} {...args} v="v1" m={m} />
        <div className="row">
          <div className="col-md-6␣offset-md-6">
            <span>
              {getTotal(k, 'moles2')} moles <sub>
                 exceso</sub> de{'␣'}
              <Reaction reaction="NaOH" />
            </span>
          </div>
        </div>
        <Measurement k={k} {...args} v="v2" m={m} />
        <Measurement k={k} {...args} v="moles3" m={m}
           />
      </div>
      <div className="card-footer">
        <b>
          {getTotal(k, 'm3')} M de <Reaction reaction=
             "NaHCO3" />
        </b>
      </div>
    </div>
    <br />
  </div>
))}
<div className="col-md-6␣mb-4␣align-self-center">
  <button
    className="btn␣btn-secondary␣btn-block"
    onClick={addMeasurement}
  >
    Agregar medici n
```

```
            </button>
          </div>
        </div>
        <h5>
          Promedio: {mean.toFixed(4)} M de <Reaction reaction="
             NaHCO3" />
        </h5>
        <br />
      </>
    );
}
```

Algoritmo A.28: components/winkler/NaHCO3pNa2CO3/NaHCO3pNa2CO3.js

```
import React, { useState } from 'react';
import Balancer from '../../Balancer';
import Reaction from '../../render/Reaction';

import NaHCO3 from './NaHCO3';
import Na2HCO3 from './Na2HCO3';

export default function NaHCO3pNa2CO3() {
  const [balanced, setBalanced] = useState({ 0: false, 1:
     false, 2: false });
  const [m1, setM1] = useState(0);
  const [m2, setM2] = useState(0);

  return (
    <>
      <h4>Paso 1</h4>
      <div className="mt-2">
        <Balancer
          reaction="NaHCO3+NaOH=Na2CO3+H2O"
          balance={s => setBalanced(balanced => ({ ...balanced
             , 0: s }))}
          balanced={balanced[0]}
        />
      </div>
```

```
<div className="mt-2">
  <Balancer
    reaction="Na2CO3+BaCl2=BaCO3+2NaCl"
    balance={s => setBalanced(balanced => ({ ...balanced
      , 1: s }))}
    balanced={balanced[1]}
  />
</div>
<div className="mt-2">
  <Balancer
    reaction="NaOH+HCl=NaCl+H2O"
    balance={s => setBalanced(balanced => ({ ...balanced
      , 2: s }))}
    balanced={balanced[2]}
  />
</div>
<br />
{!Object.values(balanced).filter(b => !b).length && (
  <>
    <div className="row">
      <div className="col-md-6">
        <div className="row mb-1">
          <div className="col-md-6">
            M de <Reaction reaction="NaOH" />
          </div>
          <div className="col-md-6">
            <input
              type="number"
              className="form-control form-control-sm"
              value={m1}
              onChange={event => setM1(event.target.
                value)}
            />
          </div>
        </div>
      </div>
```

```
            <div className="col-md-6">
              <div className="form-row mb-1">
                <div className="col-md-6">
                  M de <Reaction reaction="HCl" />
                </div>
                <div className="col-md-6">
                  <input
                    type="number"
                    className="form-control form-control-sm"
                    value={m2}
                    onChange={event => setM2(event.target.
                        value)}
                  />
                </div>
              </div>
            </div>
          </div>
          <br />
          <NaHCO3 m1={m1} m2={m2} />

          <div className="mt-3">
            <div className="mt-4">
              <h4>Paso 2: Neutralizaci n total</h4>
              <Reaction reaction="Na2CO3+2HCl=2NaCl+CO2+H2O"
                  />
              <br />
              <Reaction reaction="NaHCO3+HCl=NaCl+CO2+H2O" />
              <br />
              <br />
              <Na2HCO3 m1={m1} m2={m2} />
            </div>
          </div>
        </>
      )}
    </>
  );
```

```
}
```

Algoritmo A.29: components/winkler/NaOHpNa2CO3/NaOHpNa2CO3.js

```
import React, { useState } from 'react';
import Balancer from '../../Balancer';
import Reaction from '../../render/Reaction';
import Mean from '../../Mean';
import NaOH from './NaOH';

export default function NaOHpNa2CO3() {
  const [balanced, setBalanced] = useState({ 0: false, 1:
     false });
  const [step13_q, setStep13Q] = useState(false);

  const measures = [
    ['m', 'molaridad', 'HCl'],
    ['v', 'vol', 'HCl'],
    ['vm', 'vol', 'Muestra']
  ];
  const formula = ({ m, v, vm }) => {
    if (m && v) return (v * m * (1 / 2)) / vm;
    return 0;
  };

  return (
    <div>
      <h4>Paso 1</h4>
      <div className="mt-2">
        <Balancer
          reaction="Na2CO3+BaCl2=BaCO3+2NaCl"
          balance={s => setBalanced(balanced => ({ ...balanced
             , 0: s }))}
          balanced={balanced[0]}
        />
      </div>
      <div className="mt-2">
        <Balancer
```

```
            reaction="NaOH+HCl=NaCl+H2O"
            balance={s => setBalanced(balanced => ({ ...balanced
               , 1: s }))}
            balanced={balanced[1]}
          />
        </div>
        {!Object.values(balanced).filter(b => !b).length && (
          <div className="mt-3">
            <NaOH q={setStep13Q} />
          </div>
        )}
        {step13_q && (
          <div className="mt-4">
            <h4>Paso 2: Neutralizaci n total</h4>
            <Reaction reaction="Na2CO3+2HCl=2NaCl+CO2+H2O" />
            <br />
            <Reaction reaction="NaOH+HCl=NaCl+H2O" />
            <br />
            <br />
            <Mean
              result={['M', 'Na2CO3']}
              measures={measures}
              formula={formula}
            />
          </div>
        )}
      </div>
    );
}
```

Algoritmo A.30: components/winkler/NaOHpNa2CO3/NaOH.js

```
import React from 'react';
import Mean from '../../Mean';

export default function NaOH(props) {
  const measures = [
    ['v', 'vol', 'HCl'],
```

```
    ['m', 'molaridad', 'HCl'],
    ['vm', 'vol', 'Muestra']
  ];
  const formula = ({ v, m, vm }) => {
    if (v && m && vm) return (v * m) / vm;
    return 0;
  };

  let result = 'M';

  if (props.label)
    result = (
      <span>
        M <sub>totales</sub>
      </span>
    );

  return (
    <Mean
      result={[result, 'NaOH']}
      measures={measures}
      formula={formula}
      q={props.q}
      setMean={props.setMean}
    />
  );
}
```

Algoritmo A.31: components/winkler/Winkler.js

```
import React, { useState } from 'react';
import Reaction from '../render/Reaction';
import Dropdown from 'react-dropdown';
import NaOHpNa2CO3 from './NaOHpNa2CO3/NaOHpNa2CO3';
import NaHCO3pNa2CO3 from './NaHCO3pNa2CO3/NaHCO3pNa2CO3';

export default function Winkler(props) {
  const winkler = [
```

```
    ['naohpna2co3', 'NaOH+Na2CO3'],
    ['nahco3pna2co3', 'NaHCO3+Na2CO3']
  ].map((r, i) => ({
    value: i,
    id: r[0],
    label: <Reaction reaction={r[1]} />
  }));

  const [reaction, setReaction] = useState(null);

  const solveReaction = function() {
    if (!reaction) return null;
    switch (reaction.value) {
      case 0:
        return <NaOHpNa2CO3 />;
      case 1:
        return <NaHCO3pNa2CO3 />;
      default:
        return;
    }
  };

  return (
    <div className="form-group">
      <label>Reacciones</label>
      <div className="input-group">
        <Dropdown
          className="react-dropdown"
          options={winkler}
          value={reaction}
          onChange={setReaction}
          placeholder="Seleccionar..."
        />
        <div className="input-group-append">
          <button
            disabled={reaction === null}
```

```
                    className="btn btn-outline-secondary"
                    onClick={() =>
                      props.setReactionId('winkler', winkler[reaction.
                          value].id)
                    }
                  >
                    Gu a
                  </button>
                </div>
              </div>
              <br />
              {solveReaction()}
            </div>
          );
        }
```