Instituto de Computación
Facultad de Ingeniería
Universidad de la República

# Desarrollo de una interfaz gráfica para herramientas de calidad de agua

Nieves, Santiago
Noguera, Santiago
Ramírez, Santiago

Tesis presentada en cumplimiento de los requerimientos para la obtención del título de
*Ingeniero en Computación*

Director de tesis:
Kruk, Carla
Sabiguero Yawelak, Ariel
Segura, Ángel

Tribunal:
Meerhoff, Daniel
Pizard, Sebastián
Viera, Omar

Diciembre 2018
Montevideo, Uruguay

## Resumen

La aparición de floraciones de microalgas es cada vez más frecuente, a pesar de que existen métodos para analizar la calidad del agua, estos son realizados manualmente por profesionales e implican varios cálculos de cierta complejidad, lo que insume mucho tiempo y está sujeto a errores. El grupo MAREN (Modelización y Análisis de Recursos Naturales) y el grupo de Ecología Funcional Acuática ya contaban con ciertos algoritmos en lenguaje R para automatizar estos cálculos, pero estos no eran amigables ni de fácil uso para terceros.

En este contexto surge la iniciativa en la que se centra este proyecto de grado, la construcción de una plataforma web de acceso público que permita publicar algoritmos en lenguaje R, para que cualquier individuo pueda hacer uso de los mismos online y compartir los resultados con la comunidad.

El modelo de la solución construida permite almacenar información a medida que los algoritmos son ejecutados. Los resultados de todas las ejecuciones existentes pueden ser visualizados en la plataforma web, que cuenta con un mapa interactivo indicando la ubicación de donde provienen los datos utilizados para la ejecución.

Para el desarrollo de esta plataforma web primero se realizó un estudio de alternativas existentes para integrar R, que es el lenguaje elegido por los directores de tesis para sus algoritmos, con tecnologías que permiten construir una plataforma web. Finalmente se optó por utilizar Java para construir la plataforma web y Rserve para ejecutar los algoritmos en R utilizando Java.

La plataforma web cuenta con usuarios de rol administrativos que pueden subir y editar algoritmos, y usuarios de rol normal que ejecutan dichos algoritmos, la información de las ejecuciones es geográficamente referenciada y acompañada de otros datos pertinentes, como por ejemplo, el instituto que realiza la ejecución del algoritmo o una referencia al lugar de dónde provienen las muestras.

# 1. Introducción

## 1.1 Estructura del documento

En el primer capítulo, se describe el equipo de trabajo, las motivaciones que dieron lugar a su desarrollo, el proyecto, principales objetivos y los directores del mismo. En el segundo capítulo se presenta el resultado del relevamiento sobre proyectos similares o antecedentes, se comenta las herramientas utilizadas para construir el proyecto, una pequeña introducción a R, relevamiento de información sobre R y las aplicaciones web. El tercer capítulo se centra en el desarrollo del proyecto, desde el análisis de requerimientos hasta el diseño del mismo pasando por la arquitectura, modelado del problema y modelado de los datos. También se encuentra la integración con R, comentarios sobre los prototipos desarrollados y la descripción de la implementación. El cuarto capítulo se centra en las consideración de seguridad que se tuvo en cuenta para desarrollar la plataforma web. El quinto capítulo trata sobre las pruebas realizadas, tanto para Renjin como Rserve, teniendo en cuenta un algoritmo simple escrito en R y como segunda instancia un algoritmo que utiliza un paquete externo y Rdata, por último se hace un apartado especial para pruebas sobre Renjin. El sexto capítulo es una mención al cierre del proyecto junto con los directores del mismo El séptimo capítulo expone los resultados obtenidos al culminar el proyecto, exponiendo su planificación, ejecución y conclusiones. El octavo capítulo trata sobre el posible trabajo a futuro, discutido con los directores del proyecto. Al final se exponen los anexos y referencias respectivamente.

## 1.2 Equipo de trabajo

El equipo de trabajo de este proyecto, está compuesto por los siguientes integrantes:

- Nieves, Santiago.
- Noguera, Santiago.
- Ramírez, Santiago.

Los tres integrantes, son estudiantes de la carrera Ingeniería en Computación en la Facultad de Ingeniería (FING) de la Universidad de la República (UdelaR). El presente informe corresponde a la materia Proyecto de Grado, que es la tesis correspondiente a la carrera en curso.

## 1.3 Motivación

La solución final a la que se llega en este proyecto permite ejecutar cualquier clase de algoritmo en R que cumpla con la clase de equivalencia definida, más allá de esto, los primeros algoritmos usados en la aplicación están relacionados con el análisis de la calidad del agua, debido al contexto de trabajo de los directores del proyecto en MAREN.

A continuación se describe el contexto de trabajo en MAREN y la motivación del proyecto.

### 1.3.1 Contexto de trabajo en MAREN

Las floraciones de fitoplancton nocivas son cada vez más frecuentes en Uruguay y el mundo. El fitoplancton es un conjunto de microorganismos (microalgas) que viven en el agua, hacen fotosíntesis y cuando crecen mucho pueden ser tóxicos.

Estas floraciones generan mal olor, sabor y algunas son tóxicas para los animales y el ser humano. Las mareas rojas en el este o las manchas verdosas en la costa de Montevideo son ejemplos concretos y bien conocidos.

Existen varios casos muy conocidos de los problemas que han generado en Santa Lucía, Montevideo y Laguna del Sauce, Maldonado, pero son un problema frecuente en todo el territorio uruguayo, incluyendo las playas.

Actualmente, los análisis de calidad de agua para detectar floraciones requiere de profesionales especializados que deben realizar numerosos pasos de cálculo manuales. Esto genera que el tiempo requerido para el análisis sea grande e incrementa la probabilidad de cometer errores. Ambas características son indeseables si se pretende generar avisos confiables a la población en tiempos reducidos. En este contexto, el desarrollo de herramientas semi-automatizadas para la predicción, estudio y análisis es crucial.

### 1.3.2 Motivación del proyecto

Los directores, que son parte de los usuarios finales de la plataforma web, y quienes impulsaron la idea inicial de este proyecto, cuentan con ciertos algoritmos para realizar análisis

sobre la calidad del agua, pero no tienen una forma de compartirlos con la comunidad para que sean usados por terceros, y sus procesos no están completamente automatizados.

La principal motivación del equipo de trabajo es brindarles a los usuarios finales una forma sencilla y automatizada de publicar, usar y mantener cualquier algoritmo en lenguaje R. De forma que cualquier individuo con datos y acceso a internet pueda volcar los mismos en la plataforma web y ejecutar el algoritmo necesario.

Se busca conectar a la comunidad científica, investigadores y otros usuarios. Se quiere lograr que quien tenga algoritmos o datos los pueda compartir y que paulatinamente se vaya almacenando información útil en la plataforma. Otro fin a destacar es el ahorro de tiempo y la mitigación de errores vinculados a la manipulación humana.

### 1.4 Problema planteado

En el proceso de análisis los técnicos toman muestras de agua, cuentan cuántos organismos hay y estiman su biomasa. Eso usualmente se hace de una forma muy primitiva, manualmente o utilizando planillas de excel, es una tarea muy minuciosa y da lugar a muchos errores.

Desde el grupo MAREN y el grupo de Ecología Funcional Acuática se está trabajando hace años en el desarrollo de herramientas semi-automatizadas que sean fácilmente aplicables por uruguayos y extranjeros, tanto para los organismos encargados del manejo, como para investigadores. Se desarrollaron algoritmos, códigos funcionales y varios prototipos, los cuales no son amigables para el uso de terceros, por lo cual surge la propuesta de construir una plataforma web pública con una interfaz gráfica amigable y disponible en línea, que provea herramientas para el análisis de la calidad del agua.

### 1.5 Objetivo

El objetivo del proyecto es crear una plataforma web que facilite a los usuarios la realización de análisis del agua en base a datos relacionados con el fitoplancton.

De esta plataforma se espera que el usuario introduzca sus datos brutos y recoja la información ya procesada, incluyendo:

- Estimación de abundancia y biovolumen.
- Clasificación en grupos taxonómicos, morfológicos y estrategias de vida.
- Comparación de los valores con las normativas de calidad de agua nacionales e internacionales.
- Evaluación ecológica preliminar (si además incluye datos de variables ambientales).
- Comparación con valores en otros lugares del país y del mundo, incluyendo un mapa que muestre en tiempo real los resultados que se van generando.

Además se espera que la interfaz de la aplicación web sea amigable e intuitiva de manera que sea fácil de usar y no desmotive a nuevos usuarios potenciales.

### 1.6 Los directores

MAREN es un grupo interdisciplinario compuesto por Matemáticos, Biólogos y MSc en Economía y Turismo, que tiene como temática general de investigación la modelización matemática en Ciencias Ambientales, entendiendo como ambiente al conjunto “sistemas naturales-hombre”.

A continuación se mencionan algunas áreas de trabajo y líneas específicas:

- Ecología acuática
- Turismo
- Desarrollo de herramientas estadísticas

Su trabajo incluye modelización y predicción de problemas ambientales, generación de modelos teóricos y creación de nuevas herramientas estadísticas.

El objetivo final de su trabajo es aportar herramientas objetivas para la gestión en Ciencias Ambientales. El grupo MAREN apuesta a la transversalidad y es por esto que además de las investigaciones que se desarrollan al interior de grupo, les es de interés la interacción con otros

grupos de investigación que trabajen en distintas áreas de la Biología y de la Gestión Ambiental a modo de poder aportar sobre el aspecto de modelización Estadística.

En particular, los impulsores del proyecto y con quienes se validaron los requerimientos, son Dr. Ángel Segura y Dra. Carla Kruk.

### 1.7 Conocimientos generales

Para comprender el siguiente informe y el sistema construido en su totalidad el lector debe comprender los conceptos básicos de los lenguajes Java, R y SQL, además de patrones de diseño como MVC y conocimientos sobre el servidor de aplicaciones Wildfly. También se debe contar con cierto entendimiento del Framework Spring para Java.

# 2. Estado del arte

Como fue mencionado en el capítulo anterior, los algoritmos que se deben integrar en la plataforma web están escritos en código R, es por ello que se realizó un estudio sobre este lenguaje y las posibles formas de integrarlo con Java y Ruby. Más adelante en esta sección , se explica porqué se restringió la búsqueda a estos dos lenguajes. En esta sección se presentan los resultados de dicho estudio, además de un poco de historia sobre el lenguaje y algunas de sus principales características.

## 2.1 Antecedentes

Antes de comenzar el desarrollo del proyecto se indagaron otras plataformas web que contaran con alguno de los requerimientos del mismo.

El foco de la búsqueda fueron aplicaciones web en las cuales se pudieran clasificar microorganismos, calcular el volumen de los mismos o realizar algún tipo de análisis online, con datos relacionados a las microalgas, brindados por el usuario en algún formato dado.

Entre los sitios web encontrados se hayan EQAT - www.planktonforum.eu (External Quality Assessment Trials Phytoplankton), AlgaTerra - www.algaterra.org/, planktonportal - www.planktonportal.org.

El primero de los antecedentes nombrados, EQAT, permite la estimación de bio volumen de manera individual, mientras que en AlgaTerra se presentan algunos grupos morfológicos. Finalmente en planktonportal se puede encontrar información sobre el fitoplancton, en este portal los usuarios ayudan a identificar diferentes especies de fitoplancton en imágenes brindadas por el sitio.

Además de los sitios nombrados anteriormente, también existen numerosos portales en donde se puede encontrar información sobre las microalgas o el fitoplancton y cuestiones relacionadas, pero estas páginas no son interactivas y no permiten que el usuario ingrese sus datos para obtener algún resultado preprocesado. En complemento a los sitios nombrados

existen diversos recursos online que permiten y asisten en la ejecución de cálculos aislados, como el cálculo del volumen.

En resumen, no se encontró otra aplicación web que tuviera características similares a la construida en el marco de este proyecto. Los directores tampoco conocen otros sitios web en los que se encuentren las mismas funcionalidades.

## 2.2 Herramientas usadas para construir el proyecto

A continuación se presentan las herramientas, lenguajes y frameworks elegidos para construir la plataforma web. Todas ellas fueron validadas en conjunto con el tutor del proyecto y el grupo de trabajo MAREN.

### 2.2.1 Java y Java EE

Java es un lenguaje de programación de propósito general, orientado a objetos y concurrente, creado por Sun Microsystems. Fue diseñado tomando como patrón el lenguaje de programación C y pensado para ser portable en diversas plataformas. La mayor parte del proyecto está construido sobre este lenguaje, por lo cual es un conocimiento fundamental para poder entender en su totalidad este informe. Por otra parte, Java EE (Enterprise Edition) es una plataforma de programación para el lenguaje Java. Se basa en componentes modulares que permiten utilizar arquitecturas de varias capas distribuidas.

La razón principal de la elección de esta plataforma es que permite a los desarrolladores concentrarse en la lógica del negocio en lugar de tareas de bajo nivel. También la elección se debe a la afinidad del equipo y del tutor al lenguaje, sin dejar del lado de que hoy en día sigue siendo un lenguaje muy usado.

### 2.3.2 Framework Spring [3]

Spring es un framework, alternativo al stack de tecnologías estándares en aplicaciones Java EE. El mismo nació en una época en la que las tecnologías estándar Java EE tenían muchas aristas por pulir, los servidores de aplicaciones consumían muchos recursos y los EJB eran pesados, inflexibles y era demasiado complejo trabajar con ellos. En este contexto, spring

popularizó ideas como la inyección de dependencias, permitiendo un desarrollo más sencillo y rápido.

Spring pasó rápidamente de ser un framework inicialmente diseñado para la capa de negocios a ser un completo stack de tecnologías para todas las capas de aplicación.

**Módulos de Spring.**

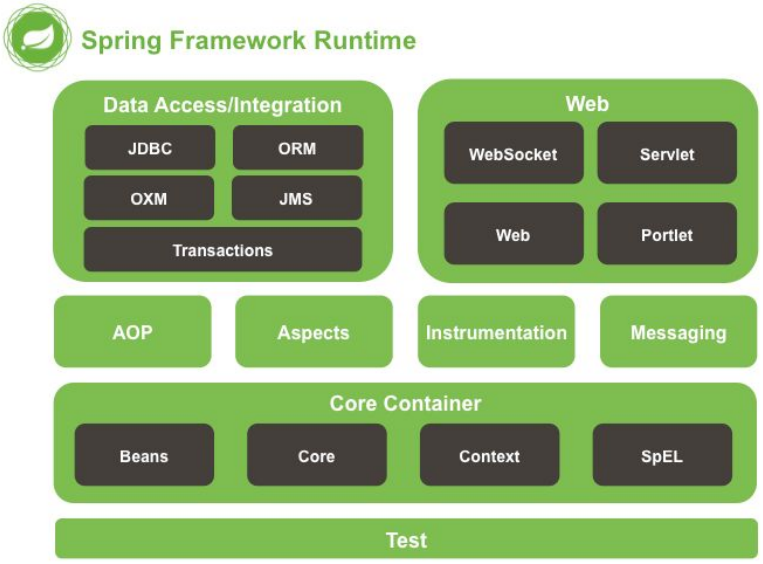



Figura 1

Dentro de la batería de herramientas de Spring Framework, se utilizaron para este proyecto los siguientes módulos:

- “Data Access” Se utiliza JDBC
- “Web” se utiliza Spring MVC
- “Core Container” Se utilizan los Beans, todo lo relativo a su invocación y manipulación, también se utiliza el context.

La principal razón por la cual se decidió utilizar este framework fue que el equipo tenía buen conocimiento del mismo y se podía aprovechar la productividad para abarcar más requerimientos del proyecto.

Desde un punto de vista técnico podríamos destacar lo siguiente:

Ventajas de Spring:

Flexibilidad para integración con otras herramientas (Ej: Hibernate, Maven, Renjin, Rserve), prolijidad del código, soporte de notaciones (patrón de inyección de dependencias) permitiendo un desacoplamiento del código.

Desventajas de Spring:

Dentro de las posibilidades que nos brinda el framework, existe una desventaja en la versión utilizada, que es la necesidad de realizar una tediosa configuración del proyecto mediante numerosos archivos XML, esto ya no es un problema en las futuras versiones de Spring.

### 2.2.3 WildFly.

WildFly anteriormente conocido como JBoss, es un servidor de aplicaciones de Java EE implementado en Java y de código abierto, su elección fue realizada por ser el servidor de aplicaciones que actualmente utiliza la mayoría del equipo en otros proyectos.

### 2.2.4 MySQL y MariaDB

MySQL está considerada como una de las bases de datos de código abierto más populares junto con Oracle y Microsoft SQL Server, el hecho de su popularidad y fácil acceso a documentación y soporte encontrado en la web, determinó el uso de esta tecnología durante el desarrollo del proyecto.

MariaDB es un sistema de gestión de base de datos derivado directamente de MySQL creado por el mismo creador de MySQL, el mismo surgió luego de la compra de MySQL por parte de Oracle, intentando mantener los mismos lineamientos y seguir siendo Open Source, es por esto que es totalmente compatible con algunas versiones de MySQL, ésta es la alternativa usada en los servidores del CURE, donde se encuentra desplegada la plataforma web.

### 2.2.6 OAuth 2 [15]

OAuth2 es un framework de autorización que permite a las aplicaciones obtener acceso limitado a cuentas de usuario en un servicio HTTP, como Facebook y Google. Delega la autenticación del usuario al servicio que aloja la cuenta del mismo y autoriza a las aplicaciones de terceros el acceso a dicha cuenta de usuario. OAuth 2 proporciona flujos de autorización para aplicaciones web y de escritorio; como también dispositivos móviles.

Teniendo en cuenta que actualmente la mayoría de personas tienen a su disponibilidad una cuenta en Google o Facebook y un smartphone o tablet, se optó por usar esta forma de autenticación.

## 2.3 Lenguaje R

R es un lenguaje de programación especialmente orientado al análisis estadístico y a la representación gráfica de los resultados obtenidos. Es un proyecto GNU. Por lo tanto, los usuarios son libres de modificarlo y extenderlo.

### 2.3.1 Historia [1]

R fue creado en 1992 en Nueva Zelanda por Ross Ihaka y Robert Gentleman. La intención inicial, era hacer un lenguaje didáctico, con el fin de ser utilizado en el curso de Introducción a la Estadística de la Universidad de Nueva Zelanda. Se decidió adoptar la sintaxis del lenguaje S, como consecuencia, actualmente la sintaxis de R es similar a la de este lenguaje. Sin embargo la semántica es muy diferente, sobre todo en los detalles más específicos de la programación.

El nombre “R” viene dado por las iniciales de los creadores del mismo, que en un comienzo bromeaban con llamarlo de esa manera. El nombre quedó y desde entonces así se le conoce en la comunidad. A continuación se expone una reseña histórica del lenguaje.

S fue desarrollado en 1976 por John Chambers en Laboratorios Bell, actualmente Lucent Tecnologies. Originalmente fue codificado e implementado como bibliotecas de FORTRAN.

En 1988 por razones de eficiencia, fue reescrito en lenguaje C, surgiendo el sistema estadístico S, versión 3. En búsqueda de propulsar comercialmente a S, Bell Laboratories dio a StatSci en 1993, una licencia exclusiva para desarrollar y vender el lenguaje S. En 1998, S ganó el premio de la Association for Computing Machinery a los Sistemas de Software , y se liberó la versión 4, la cual es prácticamente la versión actual.

El lenguaje tuvo tanto éxito que en 2004 Insghtful decide comprar el lenguaje a Lucent por la suma de 2 millones de dólares. Desde entonces, Insightful vende su implementación del lenguaje S bajo el nombre de S-PLUS. En el año 2008, TIBCO compra Insightful por 25 millones de dólares y se continúa vendiendo S-PLUS, sin modificaciones. R, que define su sintaxis a partir de esa versión de S, no ha sufrido en lo fundamental ningún cambio dramático desde 1998.

El primer anuncio al público del software R se da en 1993. En 1995 Martin Mächler, de la Escuela Politécnica Federal de Zúrich, convence a Ross y Robert a usar la Licencia GNU para hacer de R un software libre. En consecuencia, a partir de 1997, R forma parte del proyecto GNU. En 1996 se crea una lista pública de correos, el propósito era crear soporte para el lenguaje; esto no fue suficiente debido al gran éxito de R, los creadores se vieron desbordados por la continua llegada de correos. Por este motivo en 1997 tuvieron que crear dos listas de correos que son las que actualmente se usan para responder las dudas que los usuarios proponen en asuntos relativos al lenguaje. También se consolida el grupo núcleo de R, donde se involucran personas asociadas con S-PLUS, con el fin de administrar el código fuente de R.

### 2.3.2 Aspectos generales. [14]

La página principal del proyecto "R − project" es http://www.r−project.org , allí se puede conseguir gratuitamente el software en su última versión, o cualquiera de las anteriores, así como también manuales, bibliotecas, packages y demás elementos que forman parte de R. Se debe tener en consideración que R es un proyecto vivo y sus capacidades no son exactamente las mismas que S . A menudo el lenguaje S es el vínculo escogido por investigadores que utilizan la metodología estadística, y R es la ruta de código abierto para la participación en esa actividad, los usuarios pueden contribuir al proyecto implementando, creando modificaciones

de datos y funciones o bibliotecas. R reúne buenas condiciones de madurez, cantidad de recursos y manejabilidad, además ha tenido una gran implantación en la comunidad cientıfica en los últimos años,

Entre otras caracterısticas dispone de:

- Almacenamiento y manipulación de datos.
- Operadores para cálculo sobre variables indexadas (Arrays), en particular matrices.
- Herramientas para análisis de datos.
- Posibilidades gráficas para análisis de datos

El término entorno lo caracteriza como un sistema completamente diseñado y coherente de análisis de datos. Es por lo tanto dinámico y las versiones no son siempre totalmente compatibles con las anteriores. Generalmente se utiliza R como un sistema estadístico, sin embargo la descripción más acertada es la de un entorno en el que se han implementado muchas técnicas estadísticas. Algunas están incluidas en el entorno base de R y otras se encuentran en forma de bibliotecas.

Una diferencia fundamental de la filosofía de R, con el resto del software estadístico es el uso de “objetos” (variables, variables indexadas, cadenas de caracteres, funciones, etc.) como entidad básica. Cualquier expresión evaluada por R se realiza en una serie de pasos, con unos resultados intermedios que se van almacenando en objetos, para ser analizados posteriormente, de manera que se puede hacer un análisis sin necesidad de mostrar su resultado inmediatamente.

Cada objeto pertenece a una clase, esto lleva a que las funciones puedan tener comportamientos diferentes en función de la clase a la que pertenece su objeto argumento. Por ejemplo, una función no se comporta igual cuando su argumento es un vector que cuando es un fichero de datos.

Junto con R se incluyen ocho bibliotecas o paquetes, llamadas bibliotecas estándar, pero hay muchos más disponibles a través de Internet en (*http://www.r-project.org*). Actualmente se

encuentran disponibles 13517 bibliotecas desarrolladas en R, que cubren diversos campos, desde aplicaciones Bayesianas, financieras, graficación de mapas, wavelets, análisis de datos espaciales, etc.

Podemos ver o modificar la lista de bibliotecas disponibles mediante la función ".libPaths". Estas bibliotecas se pueden clasificar en tres grupos: las que forman parte del sistema base y estarán en cualquier instalación, los paquetes recomendados y otros paquetes desarrollados por investigadores de todo el mundo para tareas diversas. Destacando áreas nuevas como ciencias de la salud, epidemiología, bioinformática, geoestadística, métodos gráficos, etc.

Una característica del lenguaje R es que permite al usuario combinar en un solo programa diferentes funciones estadísticas para realizar análisis más complejos.

### 2.4 Integración de R con servicios web

Existe una gran variedad de alternativas que facilitan y/o permiten la utilización de R en conjunto con una plataforma web. En el presente proyecto se estudiaron varias de ellas, siendo características deseadas en las mismas su flexibilidad, performance, menor curva de aprendizaje, buena documentación, mantenibilidad, distribución y uso gratuito. Sumado a los aspectos recién mencionados se consideraron solo aquellas herramientas que se acoplaban con herramientas de desarrollo ya conocidas por el equipo, es por ello que se estudiaron alternativas para integrar R con Java o Ruby.

Se nombran a continuación las alternativas existentes y los motivos por los cuales fueron seleccionadas o no como candidatas a ser usadas en el proyecto. A grandes rasgos se pueden dividir en herramientas que permiten acceder a las funcionalidades de R en modalidad de servicio y las que son un medio de interactuar con R desde otros lenguajes de programación, como por ejemplo un intérprete de código.

Se hablará primero de las herramientas que permiten el uso de R como un servicio. Entre ellas encontramos soluciones como *rApache*[6], *Rserve*[8][13] y *RHadoop*[11].

Una solución existente es brindada por *Apache,* la misma es *rApache*, una herramienta que permite el desarrollo de aplicaciones web utilizando el lenguaje R y el servidor web Apache. Básicamente brinda un módulo llamado *mod_R* para embeber un intérprete de R en el servidor web y otro módulo, *libapreq,* para manejar los requests del cliente. La idea principal de esta solución es conectar requests de un servicio web, provenientes del cliente, con funciones de R definidas previamente. De forma que con *rApache* se cuenta desde un inicio con una arquitectura cliente servidor, donde se pueden ejecutar funciones o scripts de R a partir de los pedidos que lleguen al servidor.

Otra interesante alternativa es Rserve, un servidor TCP/IP que permite que otros programas utilicen facilidades del lenguaje R desde varios lenguajes de programación sin la necesidad de inicializar R o realizar una vinculación con la biblioteca de R. Cada conexión tiene un espacio de trabajo y un directorio de trabajo separado. Las implementaciones del lado del cliente están disponibles para lenguajes populares como C/C ++, PHP y Java. Rserve admite conexión remota, autenticación y transferencia de archivos. El uso típico es integrar el backend R para el cálculo de modelos estadísticos, gráficos, etc. en otras aplicaciones.

El objetivo primario de Rserve es brindar una interfaz que permite ejecutar cálculos en R. Como menciona Simon Urbanec en *Rserve A Fast Way to Provide R Functionality to Applications*[18], Rserve fue concebido pensando en tres factores, velocidad de ejecución, la fácil integración y flexibilidad.

Rserve presenta un diseño cliente/servidor, con el que se busca separar el sistema R de la aplicación. Al mantener esta separación se evitan errores o manejos de aspectos internos a R, la interfaz es muy sencilla de utilizar y permite acceder a las funcionalidades de R, sin dejar de lado la velocidad de ejecución, ya que no es necesario levantar una sesión de R desde cero cada vez que se desea ejecutar código R.

Como se explica en *Rserve A Fast Way to Provide R Functionality to Applications,* los cálculos son realizados por el core de Rserve, el cual es un servidor atendiendo pedidos de clientes, como las aplicaciones. La comunicación, como ya se ha mencionado antes, se realiza

mediante websockets, sobre el protocolo TCP/IP. Esto permite la utilización de un servidor central con Rserve y varias computadoras remotas, el uso de varios servidores Rserve para distribuir los cálculos y el caso más simple, la convivencia de Rserve y una aplicación cliente en el mismo nodo. Un servidor Rserve puede atender múltiples clientes de manera simultánea. También es bueno destacar que una aplicación puede abrir varias conexiones al mismo servidor para procesar tareas en paralelo. La transferencia de datos entre las aplicaciones y Rserve se realiza de forma binaria. Los objetos intermedios, necesarios en los cálculos, son guardados en Rserve, pero solo se transfieren objetos de interés al cliente. Rserve soporta dos clases de comandos para comunicarse con R, creación de objetos y evaluación de código R. Los tipos de objetos más básicos pueden ser construidos directamente y su contenido es enviado hacia el servidor.

Todos los objetos se pasan por valor, y no por referencia, para separar los espacios del cliente y el servidor. De esa manera tanto el cliente como el servidor pueden eliminar los datos cuando sea necesario, previniendo errores.

La segunda clase de comandos es la evaluación de código R, el código es enviado en texto plano a Rserve, y es tratado como si hubiera sido escrito desde la consola en R. El objeto resultante de la evaluación del código puede ser enviado al cliente. La mayor parte de los tipos de datos de R son soportados (números escalares, textos, vectores, listas, etc.), esto permite a Rserve pasar modelos enteros al cliente.

En cuanto al manejo de errores, una evaluación de código tiene tres posibles resultados, evaluación exitosa, error en tiempo de ejecución en el código o error de parseo. El estado de la evaluación siempre se devuelve al cliente para que se puedan tomar las acciones correspondientes.

Un uso típico de Rserve es cargar toda la información necesaria en R, realizar cálculos basados en parámetros establecidos por el usuario y enviar los resultados a la aplicación para ser mostrados. Toda la información y los objetos son persistidos hasta que se cierra la conexión.

Rserve cuenta con un cliente Java que permite acceder a todas sus funcionalidades y mapear los objetos existentes en Rserve a objetos o clases de Java.

Finalmente entre las soluciones que permiten acceder a R como servicio, se encontró *RHadoop*[11], el mismo es una colección de cinco paquetes R que permiten a los usuarios administrar y analizar datos con Hadoop. Es una solución en la nube, muy potente si se desean hacer análisis de grandes sets de datos con R, que hace uso justamente de Hadoop.

A continuación se hablará de las otras alternativas existentes, que permiten hacer uso de R desde otro lenguaje de programación como Ruby o Java.

La primera de las alternativas de esta índole estudiada es *JRI*[7], esta una interfaz Java/R, que permite ejecutar R dentro de aplicaciones Java como un único hilo. Básicamente, carga la biblioteca dinámica R en Java y proporciona una API Java para las funcionalidades de R. Admite llamadas simples a funciones R y un REPL de ejecución completa.

En cierto sentido, JRI es el inverso de rJava y ambos se pueden combinar (es decir, puede ejecutar el código R dentro de JRI que devuelve la llamada a la JVM a través de rJava).

JRI usa código nativo, pero admite todas las plataformas donde Java está disponible, incluidos Windows, Mac OS X, Sun y Linux (tanto de 32 bits como de 64 bits).

Otra opción que permite usar R dentro de Java es *Renjin*[9], un intérprete para el lenguaje de programación R escrito en Java, al igual que el proyecto R oficial, denominado GNU R, es la implementación de referencia para el lenguaje R utilizado por esta herramienta.

El objetivo de Renjin es ser compatible con R, de manera que la mayoría de los programas construidos con R se puedan ejecutar en Renjin sin la necesidad de realizar ningún cambio en el código. Renjin actualmente no es 100% compatible con R por lo que algunos paquetes pueden no estar implementados y por consiguiente no disponibles para su uso.

La mayor ventaja de Renjin es que el intérprete R en sí es un módulo de Java que se puede integrar en cualquier aplicación de Java. Esto prescinde de la necesidad de cargar bibliotecas dinámicas o de proporcionar alguna forma de comunicación entre procesos separados.

Renjin también se beneficia del ecosistema de Java que, entre otras cosas, incluye herramientas profesionales para la gestión del ciclo de vida de los componentes (o aplicaciones). Otra ventaja de Renjin es que no se requiere nada adicional para habilitar la interfaz R/Java, Renjin proporciona acceso directo a los métodos de Java utilizando una interfaz que es completamente discreta.

Por último dentro de las alternativas para Java se encuentra *RCaller*[10], que es una biblioteca de software que está desarrollada para simplificar las llamadas a R desde Java. A pesar de que no es la forma más eficiente de invocar código R desde Java, es muy simple de usar y su curva de aprendizaje es baja. Con la funcionalidad de llamado secuencial de comandos, el rendimiento no se pierde a través de un solo proceso externo. Aunque R permite un único hilo, múltiples hilos de R pueden ser creados y manejados por múltiples instancias de RCaller en Java.

Una aplicación basada en servlets puede crear instancias de muchos objetos RCaller, así como también puede usar un solo objeto usando la funcionalidad de invocación secuencial de comandos.

RCaller está escrito exclusivamente en Java y no depende de ninguna biblioteca externa, es decir, está listo para ejecutarse en cualquier máquina con instalaciones de Java y R.

Los estudios de simulación muestran que las otras bibliotecas como Rserve y rJava superan a la RCaller en tiempos de interacción. Como resultado de esto, RCaller no es adecuado para los proyectos que tienen muchos clientes que solicitan cálculos relativamente simples y pequeños.

Ya que parte del equipo de trabajo cuenta con conocimientos en desarrollo de aplicaciones web con Ruby, pareció pertinente estudiar alguna alternativa para hacer la integración de R con Ruby, y ver si existían grandes ventajas que justificaran el uso de un lenguaje no tan conocido por todo el equipo. Surgió así de la búsqueda *RinRuby*[12], el cual es una biblioteca de Ruby que integra el intérprete R en Ruby, haciendo que las rutinas estadísticas y las gráficas de R estén disponibles en Ruby. La biblioteca consta de un solo script de Ruby que es simple de instalar y no requiere ninguna compilación especial o instalación de R. Como la biblioteca es 100% Ruby, funciona en una variedad de sistemas operativos, implementaciones de Ruby y versiones de R. Los métodos de RinRuby son simples, lo que hace que el código sea legible.

Al ser Ruby 100% puro, RinRuby no necesita ser compilado con cada versión incremental de R y Ruby. Permite aprovechar las capacidades estadísticas y gráficas familiares y completas de R en Ruby (RinRuby no proporciona acceso a Ruby desde R, como es el caso de JRI y rJava).

El diseño de RinRuby permite acceder a R desde Ruby en cualquier implementación de Ruby usando un instalación estándar de R en sistemas operativos capaces de ejecutar R y Ruby (incluidos Linux, Mac OS X y Microsoft Windows).

Por último mencionamos a *Shiny*[5], esta alternativa queda aparte ya que brinda una forma de crear aplicaciones web utilizando R. Se trata un producto de *RStudio*[4], es un paquete de R que permite la creación de aplicaciones web directamente desde R. Con *Shiny* se pueden crear interfaces gráficas, usando tags HTML, templates CSS y JavaScript. Las aplicaciones creadas con *Shiny* tienen un componente gráfico, con el que interactúa el usuario, y un conjunto de instrucciones para hacer el build de la aplicación. La estructura de todas las aplicaciones de *Shiny* es la misma, un archivo *app.R* que contiene la definición de la UI y el Servidor o conjunto de instrucciones para hacer el build. Para acceder a esta aplicación desde la web se compila y genera el código HTML, CSS y JavaScript, las aplicaciones Shiny terminan siendo un documento HTML que puede ser alojado en un servidor como cualquier otro.

## 2.5 Conclusión

Si bien no existen antecedentes de aplicaciones web que brinden el mismo servicio que se busca construir en este proyecto, la variedad de alternativas existentes para integrar R con lenguajes de programación conocidos por el equipo de trabajo, y el hecho de que estas alternativas sean usadas por la comunidad, lleva a pensar que el proyecto es viable desde un punto de vista técnico.

En cuanto a R, se puede concluir que es un lenguaje estadístico completo, muy usado, que ha ganado fuerza con los años y cuenta con una comunidad activa, se puede decir que es el lenguaje estadístico de facto. Lo que hace pensar que la plataforma web que se construyó en este proyecto de grado puede ser de gran utilidad y puede abarcar un público variado, ya que R es ampliamente usado.

# 3. Desarrollo del proyecto

## 3.1 Análisis

A continuación se describe el alcance acordado con los directores, el mismo fue elaborado en un principio pensando en un tiempo realizable de 1 año. Se tomó en cuenta cubrir todos los aspectos del producto que resultan esenciales para el valor del mismo. Posteriormente se validó dicho alcance con los directores, y se planteó una priorización sobre un conjunto de funcionalidades esenciales.

### 3.1.1 Requerimientos.

Se realizaron dos reuniones presenciales con todos los directores del proyecto. En la primer instancia se relevaron y validaron la mayoría de los requerimientos, y en la siguiente se validaron los mismos, se despejaron dudas y se acordó el alcance del sistema. Los requerimientos relevados se listan a continuación.

1. Se debe construir una plataforma web mediante la cual se puedan subir y ejecutar algoritmos en código R.
2. La plataforma web debe distinguir dos tipos de usuarios o roles, usuarios normales y usuarios administradores.
3. Los usuarios normales deben poder subir sus datos y ejecutar los algoritmos disponibles en la plataforma web con los mismos.
4. Los usuarios administradores, además de poder acceder a las mismas funcionalidades que un usuario normal, debe poder subir nuevos algoritmos y versionar algoritmos ya existentes en la plataforma.
5. Versionar un algoritmo permite el cambio de codigo R. No se permite el cambio de las entradas y/o salidas ni ninguna propiedad de estas (nombre, descripción, etc. .)
6. Los algoritmos deben contar con una descripción extendida y una resumida, la versión resumida será la que se mostrará por defecto en la ejecución de los algoritmos, ver la versión más detallada es opción de los usuarios.
7. Los usuarios administradores deben autenticarse para acceder a las funcionalidades reservadas para ese rol.
8. Los usuarios administradores deben poder ver todos los datos subidos por los usuarios normales y administradores.

9. Los usuarios normales solo pueden ver los datos de que los otros usuarios, normales o administradores, hayan marcado como públicos.
10. Los usuarios deben poder descargar los resultados de las ejecuciones individuales de los algoritmos y además los resultados de todas la ejecuciones.
11. Los usuarios deben poder subir sus datos en una planilla con formato csv.
12. Los usuarios deben poder ingresar datos de manera individual para ejecutar los algoritmos más rápidamente, con el fin de brindar una manera de que los usuarios ganen confianza en la aplicación sin perder mucho tiempo en las primeras experiencias.
13. Los usuarios deben tener la posibilidad de ingresar datos complementarios al momento de ejecutar los algoritmos. Estos datos son: instituto para el cual se corren los algoritmos, lugar y fecha.
14. La plataforma web debe validar las entradas antes de ejecutar los algoritmos y mostrar mensajes adecuados si se encuentran inconsistencias.
15. La plataforma web debe contar con un mapa interactivo donde se puedan visualizar las ubicaciones donde los algoritmos han sido ejecutados.
16. La plataforma web debe soportar la subida de R datas y packages de R para su uso en conjunto con los algoritmos subidos.
17. Los usuarios deben poder ejecutar algoritmos en secuencia, siendo la salida del algoritmo anterior en la secuencia, la entrada del siguiente.
18. Los usuarios deben poder compartir una foto del agua, acompañada de un valor comprendido en una escala categórica como “clara”, “yerba”, ”verde”, ademas de fecha y ubicación. Esta funcionalidad no implica la ejecución de código R.
19. La plataforma web debe contar con una versión en inglés, además de la versión en español, ya que se espera que a la web accedan usuarios de todo el mundo.

Se identificaron además dos requerimientos no funcionales, la interfaz web de la aplicación debe ser amigable y tener buena usabilidad.

Los anteriores son los requerimientos relevados durante las primeras reuniones, de los mismos se acordó incluir en el alcance los requerimientos del 1 al 16, dejando por fuera los requerimientos 17, 18, 19.

### 3.1.2 Público objetivo.

Se prevén dos grupos, usuarios con roles administrativos y usuarios normales.

Los usuarios con roles administrativos son los que cuentan con la capacidad de subir y modificar los algoritmos que estarán disponibles para que los usuarios normales hagan uso de los mismos. Los usuarios con rol administrativo tienen un perfil científico, con cierto conocimiento en programación, ya que son los que proveen los algoritmos en código R, estos usuarios son principalmente investigadores o científicos.

Mientras que los usuarios normales son aquellos que usarán la plataforma web para ejecutar los algoritmos disponibles en la misma, estos usufructuarios de la aplicación cuentan con un perfil técnico o científico. Concretamente se espera que estos usuarios sean empleados de entes como OSE o miembros de institutos científicos.

Más allá de los grupos de usuarios esperados que se mencionan, no existe ningún tipo de restricción que impida a un individuo ejecutar los algoritmos con sus muestras y compartir la información con la comunidad científica mediante la plataforma.

## 3.2 Diseño

En este sección se describen las decisiones de diseño que se tomaron, incluyendo la arquitectura del sistema y una descripción de la base de datos utilizada.

### 3.2.1 Arquitectura.

La plataforma web está diseñada en 3 capas, las mismas son Capa de Presentación, encargada de presentar el sistema al usuario, Capa Lógica, donde son procesadas las peticiones del usuario para luego enviarle una respuesta y por último la Capa de Datos, donde residen los datos y es encargada de acceder a los mismos.

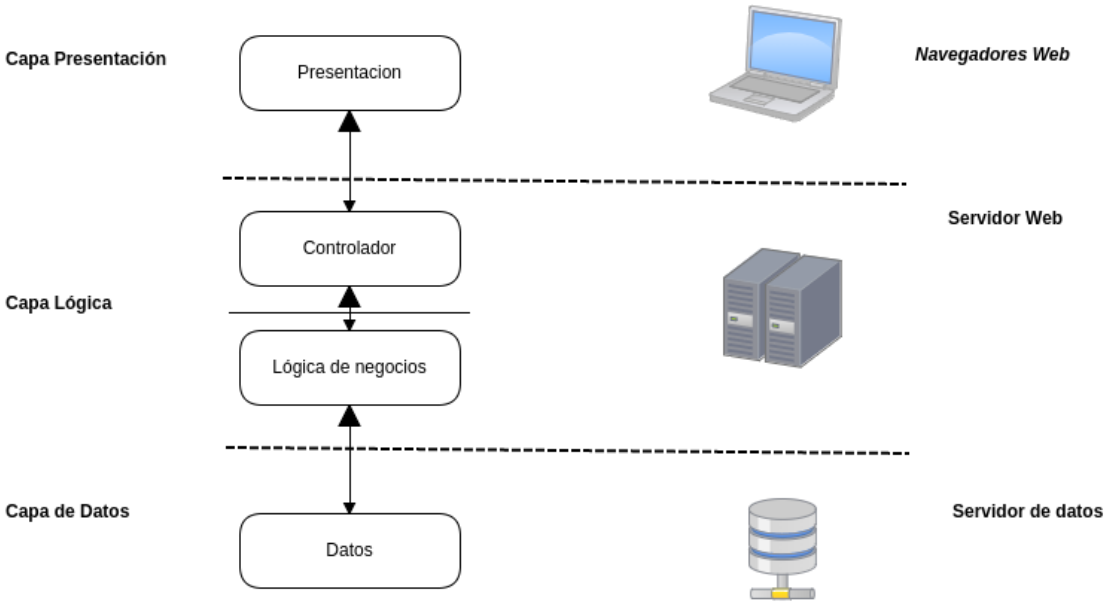



Figura 2

Por otro lado, como ilustra la figura 2, la capa lógica y de datos fueron diseñadas en módulos separados pero dentro de la misma aplicación web, que corre en un único servidor, siendo por fuera la configuración de acceso al servidor de base de datos. Además la arquitectura está basada en el patrón MVC, la cual puede verse en más detalle a continuación.

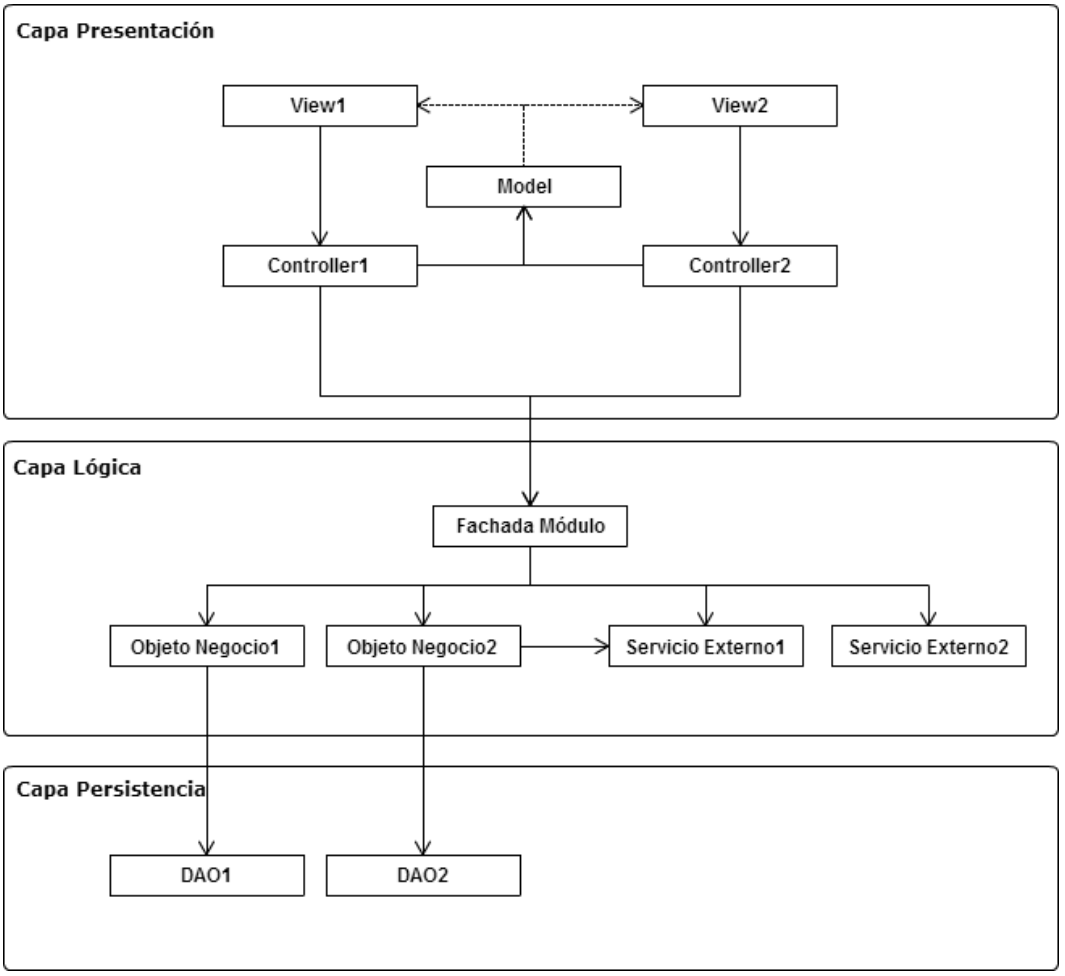



Figura 3

En la figura 3 se ve en más detalle la arquitectura en capas elegida, se explican a continuación algunos de sus actores.

**Model**

Contiene el conjunto de datos necesarios (estado) del contexto de ejecución del sistema.

**View**

Esta clase se encarga de dar formato a la representación de los datos a mostrar al usuario, estos datos los extrae de la clase modelo.

**Controller**

El rol principal de esta clase es recibir las instrucciones por parte del usuario. En base a estas instrucciones, el controlador se comunica con capas inferiores y posteriormente afecta el modelo.

**Fachada Módulo**

Brinda acceso a la capa lógica de un módulo, agrupando todas las funcionalidades provistas por este último. Esto se realiza fundamentalmente para permitir modularidad vertical, es decir poder modificar la capa de presentación de forma independiente de la capa lógica. Además este diseño provee modularidad horizontal ya que cada módulo se puede implementar por separado.

**Objeto Negocio**

Su función principal es ejecutar las reglas de negocio pertinentes a las operación que se está realizando, como por ejemplo: validaciones, formateo de datos, etc.

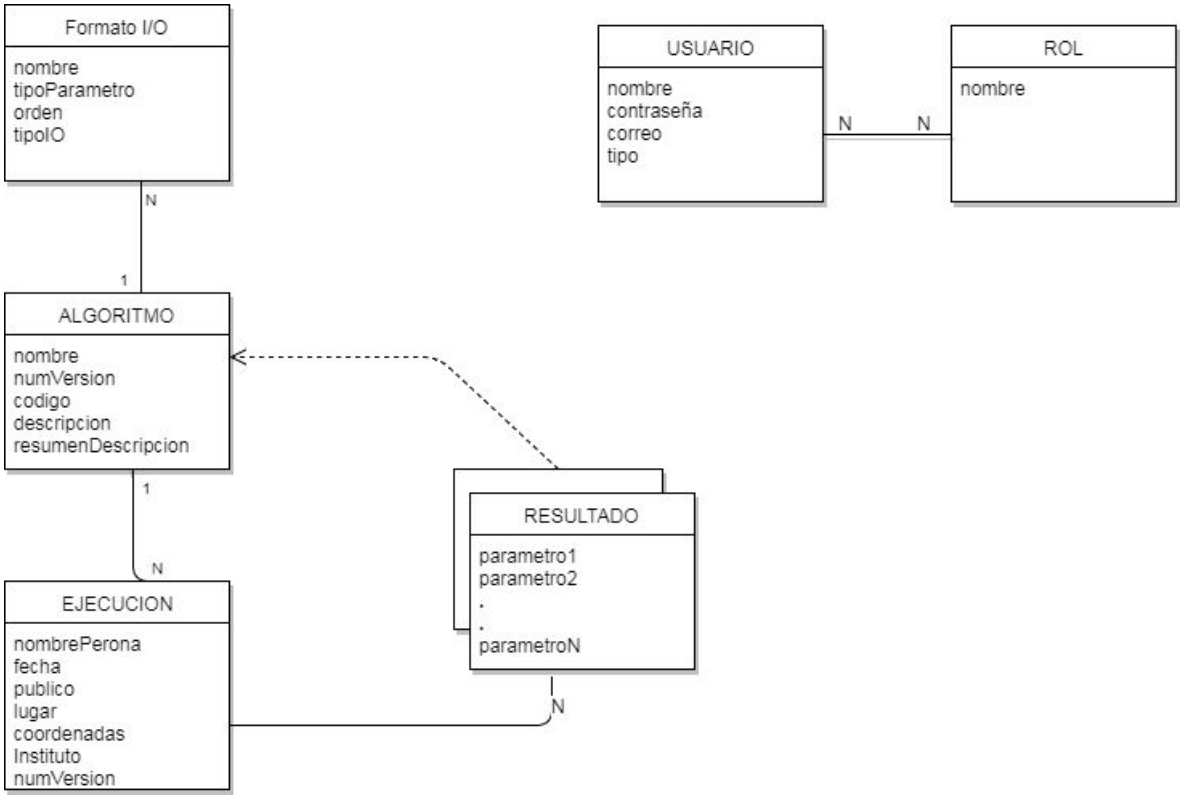
**DAO (Objeto de acceso a datos)**

Como su nombre lo sugiere, el rol de esta clase es el de acceder al servidor de base de datos, así como consultar, ingresar, modificar y borrar registros de ella.

### 3.2.2 Modelado del problema.

La necesidad de ejecutar algoritmos dados de alta por los administradores del sistema trae consigo el problema de llegar a una representación genérica de los mismos, que contemple la posible inclusión de otros algoritmos en el futuro. Además el modelo debe soportar el guardado de los resultados de cada ejecución de los algoritmos, junto con la información complementaria de quién los ejecuta y la ubicación geográfica de donde provienen los datos. A continuación se presenta la solución elegida y las diferentes decisiones tomadas en el proceso de desarrollo.

En el problema planteado, se requiere modelar algoritmos en lenguaje R, cargados desde un archivo y/o ingresados desde la web, dichos algoritmos son almacenados en el sistema y los mismos son visibles en la plataforma web para su utilización, actualmente se cuenta con 3 códigos, 2 independientes y otro que depende de uno de ellos para un cálculo previo. Se asume que todos los algoritmos reciben una línea de entrada con sus parámetros correspondientes o un CSV con múltiples líneas de entrada, la salida de forma similar, devuelve una única línea de resultado o múltiples líneas.




Modelo de dominio.

Figura 4

Se define la entidad Algoritmo, esta es responsable de contener el código R del algoritmo, que es ingresado al sistema junto con su información adicional, un algoritmo tiene una o varias entradas/salidas, llamadas Formato I/O en el modelo de dominio, estas entradas/salidas, tienen un nombre, un tipo, que indica el tipo de dato para su posterior validación y un orden para ser interpretado en la vista. A los algoritmos también se les asocia una ejecución realizada por un usuario, esta contiene los datos que los directores especificaron en el documento de relevamiento. Cabe resaltar el campo público que indica si los resultados son visibles a otros usuarios.

Por cada algoritmo existe una entidad Resultado, que es adaptada a las especificaciones de los resultados de cada algoritmo, por lo cual existirían varias entidades Resultado, en las cuales las instancias almacenadas en la misma corresponden a cada línea de un resultado, adelantando a etapas posteriores de diseño, esto quiere decir que un resultado se identifica con un id propio indicando la línea del resultado y un id de ejecución. Estas dos claves indican en la tabla

correspondiente a cada algoritmo las líneas y el fin de un resultado concreto. Esta forma de diseño está pensada para hacer que la inserción y la recuperación de información tenga la mejor performance posible.

Luego se cuenta con una sub agrupación de entidades correspondientes a la parte de seguridad de la aplicación, esta trata sobre los usuarios y sus posibles privilegios.

### 3.2.3 Modelo de base de datos.

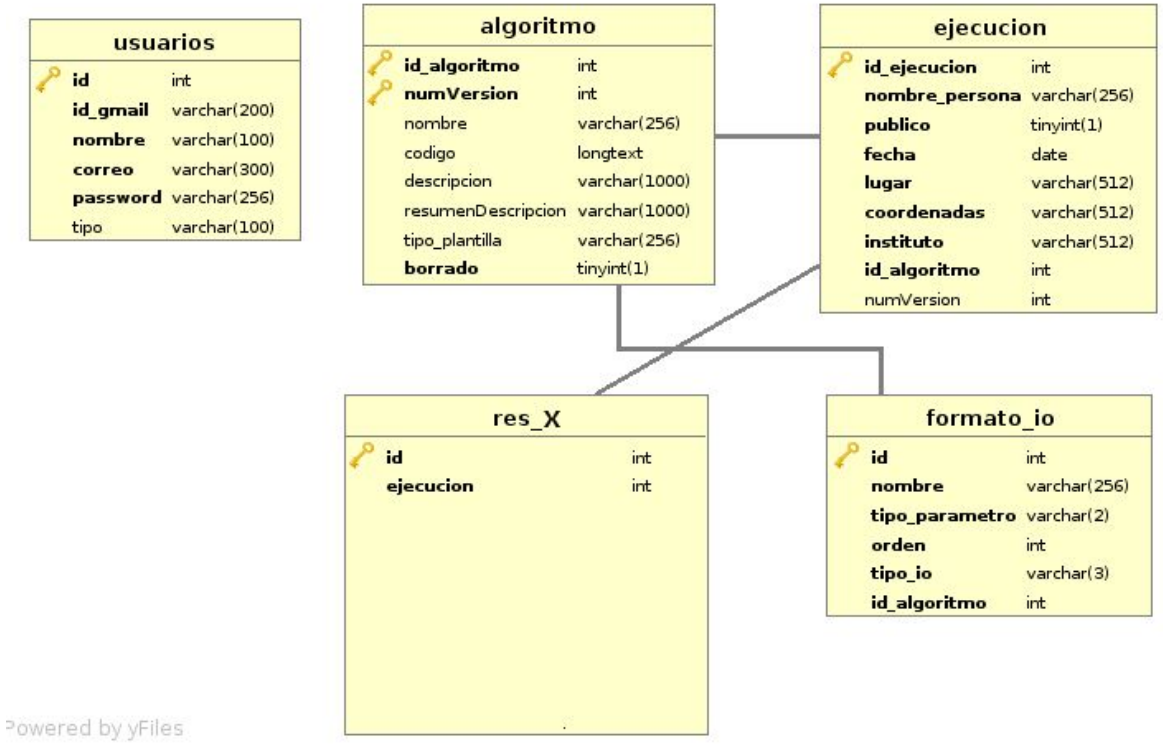



Figura 5

**algoritmo:**

Tabla que es usada para guardar información propia del algoritmo.

-id : Identificador de la tabla.

-numVersion: Número de versión.

-nombre: Nombre del algoritmo.

-codigo: Código en lenguaje R del algoritmo.

-descripcion: Descripción larga del algoritmo.

-resumenDescripcion: Descripción corta del algoritmo.

-tipo_plantilla: Utilizado para en un futuro "enganchar algoritmos".

-borrado: Indica si un algoritmo debe ser mostrado en la interfaz.

**formato_io**

Tabla que es usada para guardar información sobre las entradas y salidas de cada algoritmo.

-id: Identificador de la entrada o salida

-nombre: Nombre de la entrada o salida

-tipo_parametro: Indica si el tipo de parámetro de entrada o salida. Estos valores pueden ser D (Double), I (Número entero), S (Texto plano) o SH (figura geométrica).

-orden: Orden que tienen las entradas y las salidas en relación a las columnas representadas por la matriz.

-tipo_io: Indica si es una entrada o una salida. Los valores posibles son IN o OUT.

-id_algoritmo: Identificador del algoritmo al que pertenece la entrada o salida.

**ejecucion**

Tabla que es usada para guardar información de las ejecuciones de los algoritmos.

-id: Identificador de la ejecución.

-nombre_persona: Indica el nombre del usuario que ejecutó el algoritmo.

-publico: Indica si el resultados del algoritmo puede ser visto por cualquier usuario o solo por administradores.

-fecha: Fecha que se ejecutó el algoritmo

-lugar: Lugar donde se obtuvieron las muestras a ser analizadas

-coordenadas: Coordenadas de donde se obtuvieron las muestras a ser analizadas.

-instituto: Instituto responsable de las muestras a ser analizadas.

-id_algoritmo: Foreign key que hace referencia al campo id de la tabla algoritmos.

-numVersion: Número de versión del algoritmo

**res_X**

Tabla que almacena los valores de las entradas y salidas del algoritmo de id X.

Esta tabla tiene tantos campos como la suma de las entradas y salidas del algoritmo de id X

Además tiene los siguientes campos:

-id: Identificador del resultado.

-ejecucion: Foreign key que hace referencia al campo id de la tabla ejecuciones.

**usuarios:**

Tabla que almacena la información de los usuarios registrados en el sistema.

-id_gmail: Identificador de cuenta en gmail

-nombre: Nombre del usuario

-correo: Dirección de correo electrónico del usuario.

-password: Password encriptada del usuario.

-tipo: Indica si el usuario es Administrador o un usuario común. Los valores posibles son ADMIN y COMUN

## 3.3 Implementación

A continuación se presenta cuáles fueron las opciones que se tuvieron en cuenta al momento de integrar R al sistema, los prototipos construidos y cómo se implementó la ejecución de algoritmos.

### 3.3.1 Integración de R

En la sección 2.4 *Integración de R en aplicaciones web* se mencionaron las diferentes alternativas existentes para manejar la integración de R en plataformas web. En esta sección del informe se presenta el análisis hecho sobre cada una de estas alternativas y los motivos por los cuales se descartan o eligen las mismas para realizar análisis más profundos.

Cabe destacar que los frameworks webs considerados fueron Spring y Ruby on Rails, para los lenguajes de programación Java y Ruby respectivamente.

**Alternativas descartadas**

Se descartó *Shiny* porque el equipo de trabajo ya tiene expertise realizando interfaces gráficas para aplicaciones web. La utilización de *Shiny* tendría una curva de aprendizaje importante, además de la pérdida de flexibilidad a la hora de diseñar la UI de la aplicación. Su

uso parece más adecuado para un equipo de desarrolladores con experiencia en R y bajos conocimientos en desarrollo de aplicaciones web. Además, la "comunidad" de desarrolladores usando esta tecnología es muy pequeña comparada con otras como la de Java.

Por otro lado no se optó por *RCaller* debido a que existen otras alternativas más eficientes como JRI, Renjin o Rserve, además la documentación de RCaller no brinda seguridad ya que es escasa y no tiene mucho renombre en la comunidad. Por último la propia documentación de la herramienta desalienta su utilización para casos de uso muy similares al proyecto que debe llevarse a cabo.

Una solución en la nube como *RHadoop* no pareció adecuada, ya que no es necesario contar con esta infraestructura para este proyecto, y la misma agrega complejidad en aspectos que con otras alternativas son relativamente sencillos. El uso de *RHadoop* se puede evaluar, y tiene más sentido, en un contexto donde se requiera hacer una análisis pesado y extenso de información. En el contexto de este proyecto una solución como esta no justifica la complejidad que agrega.

*rApache* no cuenta con clientes en Java o Ruby, que permitan integrar fácilmente con el resto del proyecto web. Esta herramienta carece de buena documentación, implica manejar los pedidos del cliente a bajo nivel utilizando *Apache* y en función de esto ejecutar funciones o scripts de R. *rApache* no es una alternativa que puede sumarse al stack tecnológico que desea manejar el equipo de trabajo, o no lo es de manera sencilla. Esta alternativa no es conveniente teniendo en cuenta otras estudiadas.

**Alternativas elegidas para analizar en profundidad**

Se consideró muy importante al momento de evaluar las alternativas para integrar R, el hecho de que la mismas fueran adaptables con frameworks y lenguajes de desarrollo familiares para los integrantes del equipo encargado de la implementación del proyecto, como lo son Java o Ruby. Fue por ello que pareció más adecuado optar por alternativas que brindan suficiente flexibilidad al mismo tiempo que abarcan completamente las necesidades del proyecto y reducen considerablemente la curva de aprendizaje y el riesgo.

Una de las opciones tenidas en cuenta es Rserve, varios de los aspectos de esta herramienta la hacen una buena alternativa, empezando por que cuenta con un cliente Java. Además brinda una interfaz sencilla y flexible que permite acceder a gran parte de las funcionalidades de R, entre ellas la posibilidad de evaluar código junto con parámetros de entrada, como pueden ser datos brindados por los usuarios de la aplicación. Sumado a esto Rserve permite comenzar con un caso de uso base muy sencillo, donde el servidor y el cliente conviven en el mismo nodo, sin perder la posibilidad de escalar a varios nodos ejecutando servidores Rserve y eventualmente distribuir los cálculos. Rserve fue pensado para ser utilizado en situaciones muy similares al contexto de este proyecto, por lo que pareció sensato abordar esta alternativa en profundidad.

Tanto JRI/RJava como Renjin permiten la integración de R con Java, ambas soluciones son muy usadas y recomendadas por la comunidad, son fácilmente integrables en el proyecto web y sencillas de usar. JRI/RJava es una opción muy recomendada por la comunidad para integrar Java con R ya que brinda una API de fácil utilización, se integra con un lenguaje de programación muy potente y conocido por el equipo y ejecuta código nativo lo cual es positivo ya que no se excluyen funcionalidades de R.

Al contar también con conocimientos en Ruby se decide profundizar en el estudio de RinRuby y su integración con R. Con la premisa de que brinda una forma sencilla de evaluar scripts R desde Ruby, accediendo a las variables definidas en R, lo cual es justamente una de las necesidades del proyecto.

### 3.3.2 Prototipos

Una vez recabados todos los requerimientos y habiendo hecho sobre los mismos verificaciones de validez, totalidad y consistencia se realizaron prototipos para verificar el realismo de ciertos requerimientos, para garantizar que podrían implementarse. Mediante los prototipos se buscó validar todos los requerimientos relacionados con la ejecución de códigos en lenguaje R desde una plataforma web. Los prototipos se realizaron con las herramientas que se consideran como mejores alternativas en la sección 3.3.1 *Integración de R,* estas son JRI/rJava, Renjin, Rserve y RinRuby. En los cuatro casos se logró crear un prototipo que

ejecutara código en R, y se ejecutaron satisfactoriamente los tres algoritmos iniciales, brindados por los directores.

Los prototipos realizados son simples, evalúan código R proveniente de un archivo, el código R lee los datos que precisa de un archivo csv u ods y escribe los resultados en un archivo cvs u ods. Esto es justamente lo que se precisa de las herramientas evaluadas para llevar a cabo la aplicación web deseada. Es importante hacer énfasis en que se puede dar por hecho que todos los algoritmos que se usarán respetarán esas condiciones o formato, es decir: el código en R lee sus datos de un cierto archivo, se ejecuta todo el código y por último se escriben los resultados en otro archivo.

**Observaciones**

Para la instalación de JRI/rJava Instalación rJava:

- Ejecutar el siguiente comando en un entorno de R: `install.packages('rJava')`
- Setear variables del sistema (ubuntu): `R_HOME=/usr/lib/R/`
- Agregar las siguientes bibliotecas al proyecto Java: JRI.jar, JRIEngine.jar, REngine.jar

**Problemas encontrados**

El comando write.table se usa en R para crear un archivo excel. Uno de los ejemplos enviados por los directores contiene la siguiente línea de código R:

```
write.table(CLASIF,"CLASIF.csv",col.names=T,row.names=FALSE,sep=",",dec=".")
```

En la misma se crea un archivo csv que usa el caracter "," para separar columnas y el carácter "." para separar decimales. rJAVA separa los decimales usando el caracter ",", por lo que al abrir el archivo csv no se distingue una nueva columna de un decimal y las columnas quedan desfasadas y con valores incorrectos.

**Conclusiones**

La alternativa existente para utilizar Ruby, RinRuby, no presentó ninguna ventaja en comparación con Renjin, y como el equipo de trabajo es más afín a Java se descartó. Inicialmente se decidió utilizar Renjin porque es la alternativa más fácil de usar con Java, la que mejor se acopla, recomendada por la comunidad y la documentación es muy buena. Se tomó en cuenta también que para rJava se deben implementar controles de concurrencia, mientras que usando Renjin eso ya está resuelto.

En una de las reuniones con los directores de proyecto en el CURE se relevó la intención de utilizar ciertos algoritmos que tenían como dependencias algunos packages no implementados para Renjin, por lo cual se decidió añadir Rserve a la plataforma web, ya que éste ejecuta código nativo, y cualquier package de R puede ser instalado de necesitarlo. En la sección *3.3.3 Descripción de la implementació*n se habla más detalladamente de esta decisión.

### 3.3.3 Descripción de la implementación

Cada código R se puede ver como una función. Es decir, existen entradas, se aplican cálculos a esta entradas que generan una salida. El sistema tiene que soportar la mayor variedad de códigos R posible, esto implica aceptar la mayor variedad de entradas posibles. Por ejemplo, algunos algoritmos de R tienen como entrada 1 único valor, otros tienen como entrada un vector de valores.

Entrada - > Salida

(x,y,z) --- algoritmo R - > (s,r)

Se decidió que la mejor manera de generalizar las entradas es que estas sean matrices NxM con valores de tipo strings (texto), y generar una matriz salida NxP, también con valores de tipo strings.

**Alta de algoritmos**

Al dar de alta un algoritmo R en el sistema, se debe indicar la cantidad de entradas que este utiliza y la cantidad de salidas que se generan. Para cada entrada y salida, se debe especificar el tipo de dato (string, integer, boolean o shape) para luego poder aplicar su correcta manipulación, y una descripción para especificar unidades (por ej, $m^2$). También se ingresa un nombre, una descripción detallada y otra descripción breve. Esta última es la que se muestra por defecto al momento de ejecutar el algoritmo.

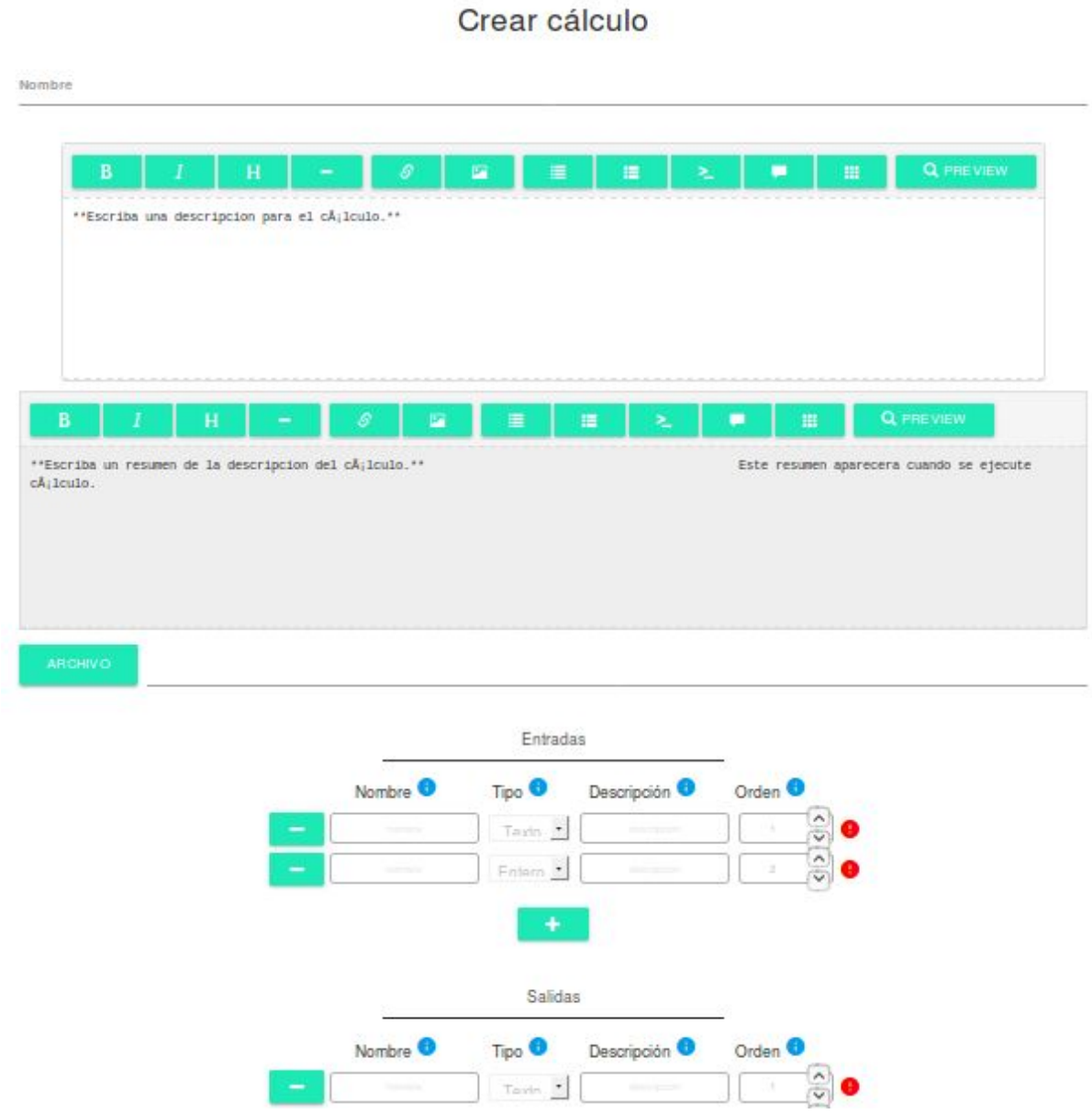



Figura 6

**Versión de algoritmos**

Al editar un algoritmo, solo se puede modificar el algoritmo R, nombre y descripción. No así las salidas y las entradas. Esto es porque ya se generó en la base de datos una estructura que las representa. A esta edición del algoritmo, la llamamos una nueva versión del algoritmo, que es la, que a partir del momento de edición, va a ser ejecutada, dejando obsoleta la versión anterior del algoritmo.

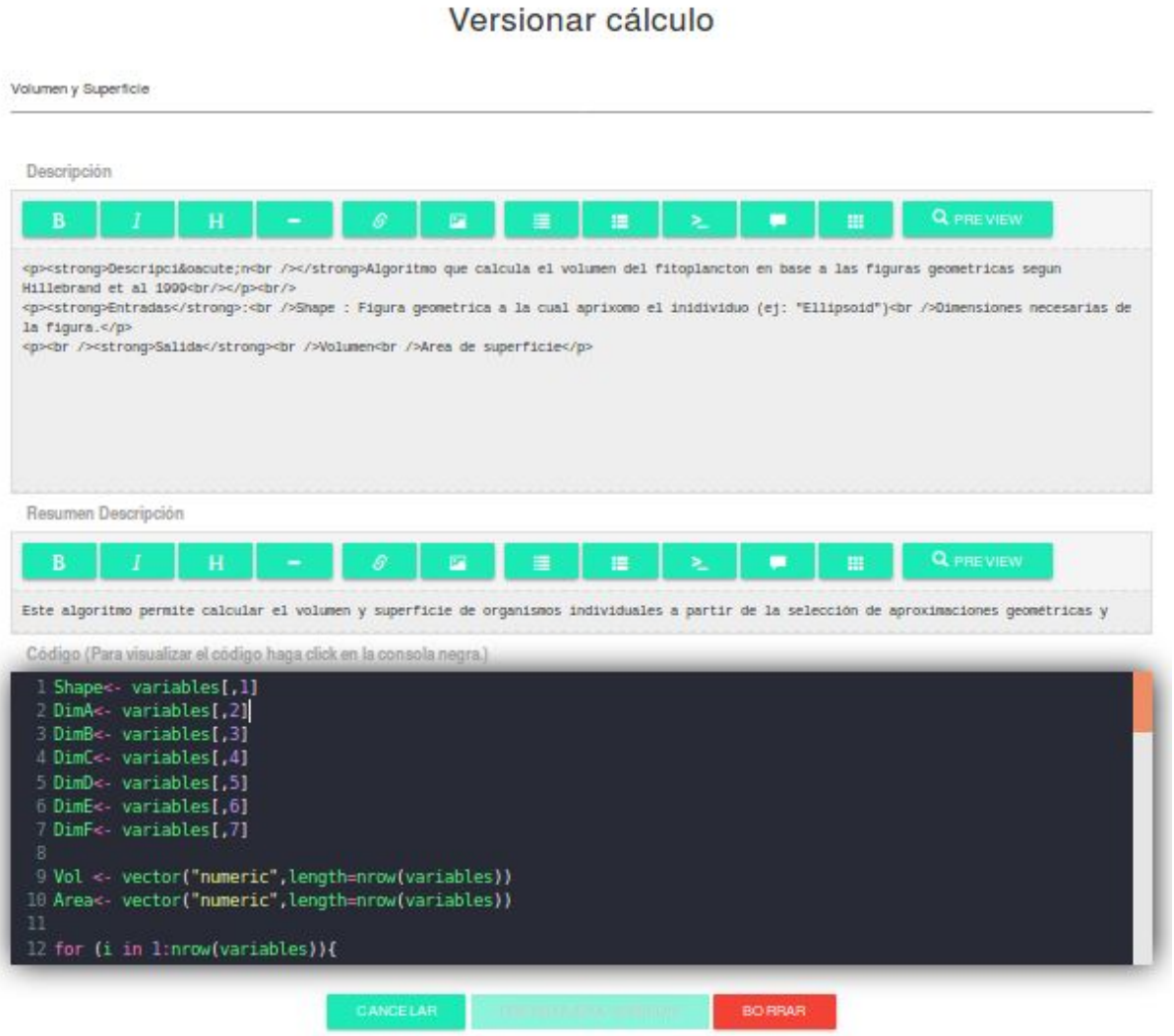



Figura 7

**Ejecución de algoritmos**

Al momento de ejecutar un algoritmo, el usuario sube un archivo (csv o ods) con la matriz de entrada para el algoritmo. Esta matriz entrada es separada por columnas. Siendo cada columna una entrada distinta del algoritmo.

Al subir el archivo el sistema chequea que la entrada tenga todas las entradas (columnas) que el algoritmo necesita, en caso de faltar alguna entrada, el algoritmo no es ejecutado y se avisa al usuario cual es la entrada faltante. También se chequea que cada entrada sea del tipo de parámetro definido en el alta del algoritmo, en caso de no serlo, se le avisa al usuario cual es el tipo de parámetro esperado.

Una vez obtenida la entrada separada por columnas, se ingresan los valores en la "sesión R", es decir, mediante código java, se llama al motor de R, se abre una nueva sesión, y se ingresan los valores de la entrada ejecutando un código en R. Esto genera, por cada entrada, en R una variable de tipo vector con nombre igual al nombre de la entrada, y valores del vector igual a los valores de la entrada ingresada por el usuario.

Se ejecuta el algoritmo R correspondiente en la sesión R existente. Esto genera las salidas en forma de vectores. Y luego, teniendo los nombres y tipo de parámetro de la salida, se va a buscar desde java a la sesión R los vectores de salida.

Una vez obtenidas las salidas, se concatenan las columnas de salidas con la matriz de entrada (agregando columnas) y se genera la matriz de salida final que es la que se le presenta al usuario. Si en tiempo de ejecución existe algún error inesperado, el sistema despliega el error tanto en la pantalla como en el log del sistema.

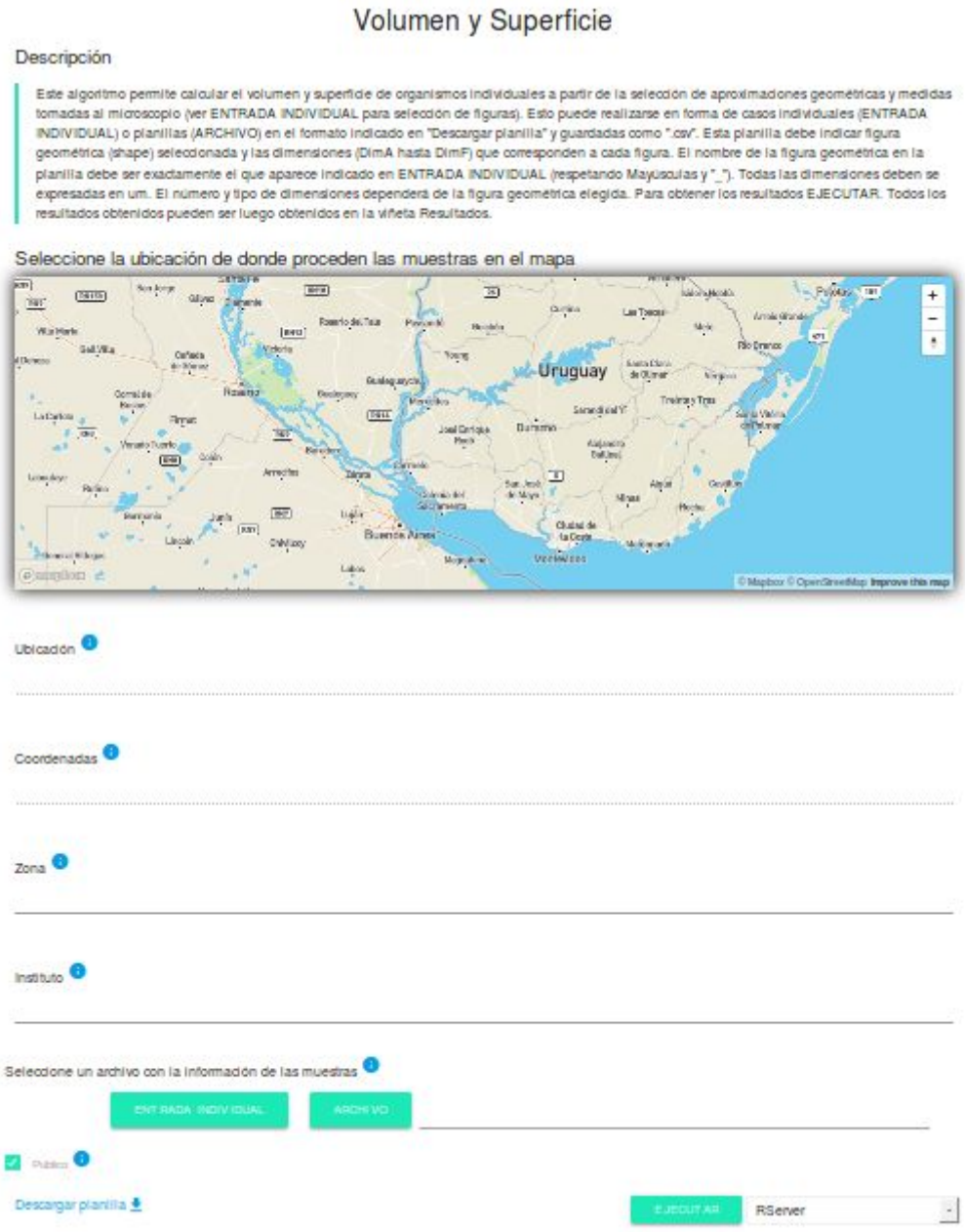

Volumen y Superficie
Descripción
Este algoritmo permite calcular el volumen y superficie de organismos individuales a partir de la selección de aproximaciones geométricas y medidas tomadas al microscopio (ver ENTRADA INDIVIDUAL para selección de figuras). Esto puede realizarse en forma de casos individuales (ENTRADA INDIVIDUAL) o planillas (ARCHIVO) en el formato indicado en "Descargar planilla" y guardadas como ".csv". Esta planilla debe indicar figura geométrica (shape) seleccionada y las dimensiones (DimA hasta DimF) que corresponden a cada figura. El nombre de la figura geométrica en la planilla debe ser exactamente el que aparece indicado en ENTRADA INDIVIDUAL (respetando Mayúsculas y "_"). Todas las dimensiones deben se expresadas en um. El número y tipo de dimensiones dependerá de la figura geométrica elegida. Para obtener los resultados EJECUTAR. Todos los resultados obtenidos pueden ser luego obtenidos en la viñeta Resultados.
Seleccione la ubicación de donde proceden las muestras en el mapa
Uruguay
Ubicación
Coordenadas
Zona
Instituto
Seleccione un archivo con la información de las muestras
ENTRADA INDIVIDUAL
ARCHIVO
Publico
Descargar planilla
EJECUTAR
RServer

Figura 8

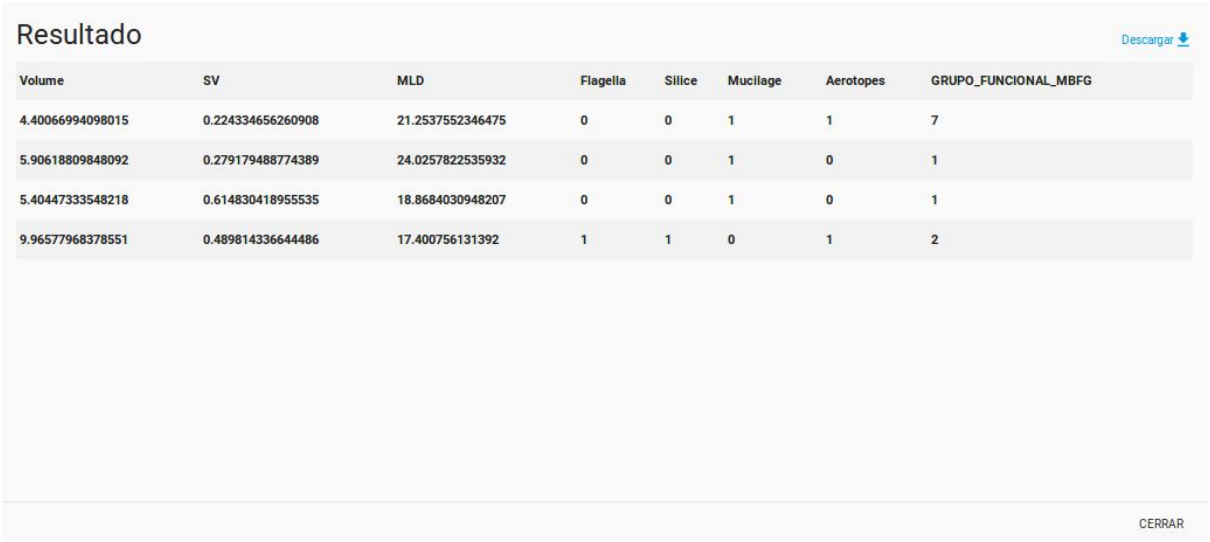



Figura 9

**Rdata**

Los archivos Rdata son archivos con extensión .rdata que contiene objetos R inicializados (variables, funciones, bibliotecas, etc.). Estos sirven para cargar en un ambiente de trabajo, mediante un comando en R, los objetos R definidos. Varios códigos de R presentados por los directores utilizaban Rdata para cargar objetos de R. Para esto se creo una carpeta en el servidor donde se alojan los Rdata disponibles y se implementó una funcionalidad para subir los archivos .rdata a la carpeta. Antes de que cada código R sea ejecutado, el sistema ejecuta un comando en R para definir esa carpeta y así el motor de R sabe la ruta donde debe ir a buscar los Radata para ser cargados.

**R packages**

Los R packages son colecciones de funciones R, código compilado y datos. Son guardados en el directorio "library" del ambiente R. Por defecto, R instala una lista de paquetes durante la instalación de R. Otros paquetes pueden ser añadidos luego. Cuando se inicializa la consola R, solo algunos paquetes están disponibles por default. Otros paquetes que ya están instalados tienen que ser cargados explícitamente para ser usados.

Para instalar paquetes en R se necesita usar el siguiente comando:

```
install.packages("nombrePaquete")
```

Una vez instalado el paquete necesita ser cargado para su utilización. Esto se logra con el comando:

```
library() ó require()
```

La diferencia entre estos comandos es que library tira un error si no encuentra el paquete. Por otro lado, require permite seguir con la ejecución del programa si el paquete no se encuentra.

**R packages y Renjin**

Cuando se usa Renjin, los paquetes R son tratados como cualquier otra dependencia Java, y deben ser ubicados en el classpath de la aplicación usando Maven o herramienta similar. Se prevé de un repositorio donde se encuentran todos los paquetes CRAN (the Comprehensive R Archive Network) en *http://packages.renjin.org*. Los paquetes ahí ubicados, fueron construidos para ser usados con Renjin. No todos los paquetes (los que no pertenecen a CRAN) están construidos para ser usados con Renjin. Y algunos de los que sí están, pueden presentar problemas de compatibilidad. Por eso es recomendado chequear en el repositorio si el paquete deseado está disponible para Renjin.

**Ejemplo con Random Forest**

En una reunión de trabajo con los directores, se mencionó que uno de los paquetes que solicitaban era el "Random Forest". Este paquete si está disponible en el repositorio.

Para su instalación se tuvo que agregar la siguiente dependencia utilizando las siguientes líneas de código en el archivo *pom.xml* del proyecto:

```
<dependency>
            <groupId>org.renjin.cran</groupId>
            <artifactId>randomForest</artifactId>
            <version>4.6-12-b50</version>
</dependency>
```

Luego, en el código R se invocó el paquete utilizando el siguiente comando:

```
require(randomForest);
```

El resultado fue exitoso debido a que dentro del código se utilizan objetos R cargados por el paquete RandomForest. Si no se hubiese cargado correctamente, estos objetos R no hubiesen podido ser reconocidos y el programa hubiese dado el error: *no applicable method for 'predict' applied to an object of class "randomForest"*.

**R packages y Rserve**

Rserve, al interactuar directamente con el R instalado en el servidor, no tiene problemas de compatibilidad de paquetes. Para que un paquete esté disponible para ser utilizado en el servidor web, este debe estar instalado en el ambiente de R del servidor.

**Rserve y Renjin**

Si bien ambos motores de R (Renjin y Rserve) son compatibles con la mayoría de los códigos de R presentados por los directores, cada uno tiene su ventajas y desventajas.

Por parte de Renjin tenemos un mejor acoplamiento con Java. Pero se pierde compatibilidad con R al no tener todos los paquetes disponibles.

Por parte de Rserve se gana compatibilidad, asegurándose poder ejecutar cualquier código de R que se presente. La desventaja es que se ingresan otros aspectos ajenos a Java que pueden generar nuevos problemas (por ejemplo, nuevas conexiones TCP/IP, transformación de objetos de R a Java) tanto de seguridad como de performance.

Teniendo en cuenta lo mencionado anteriormente, que el sistema se ha probado con un grupo acotado de algoritmos y que eventualmente pueden usarse otros algoritmos nuevos, se decidió contar con los dos motores disponibles en la plataforma web, ya que los nuevos algoritmos pueden tener diferencias de performance dependiendo del motor de R que se use. Solo los usuarios administradores pueden elegir con qué motor correr un algoritmo, esto se hace desde la interfaz de ejecución de un algoritmo. Los usuarios normales utilizan por defecto Rserve, no siendo visible para ellos la alternativa de motores. Se eligió Rserve como motor por defecto porque como ya se explicó cuenta con todos packages de R.

### 3.4 Decisiones de diseño

A medida que se conocía la realidad del problema, el modelo de dominio y por consiguiente el de datos fue cambiando, en un principio desde el punto de vista de la accesibilidad, se manejó utilizar permisos por funcionalidades que se asocien a roles como comúnmente se estila. En el relevamiento se manifestó que únicamente van a existir usuarios normales y administradores, ambos con funcionalidades fijas y que a priori no había intenciones de que exista otro grupo de usuarios.

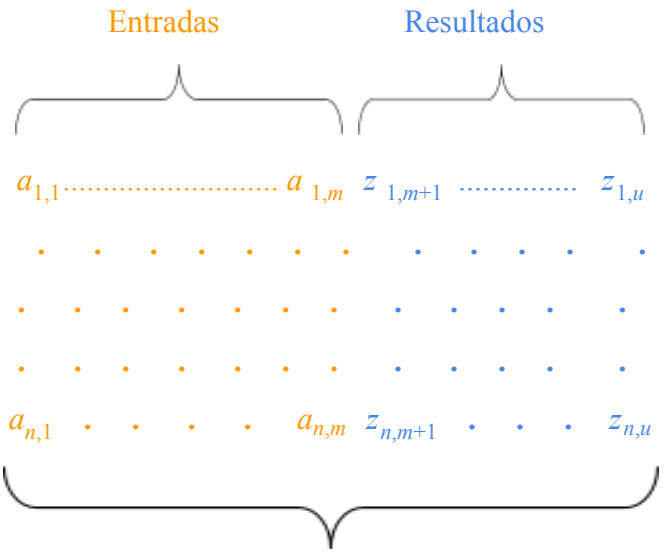
En base a esto, se decidió en vez de utilizar varias entidades para modelar este aspecto simplificar y asumir que este requerimiento no va a cambiar, de esta forma se agregó un único campo tipo de usuario, el cual es el atributo dado para decidir luego en los controladores qué vistas y request puede solicitar el usuario.

El ingreso/registro al sistema se puede realizar por medio de dos vías, desde el autenticador de google o ingresando al sistema por medio de usuario y contraseña, siendo esta última opción solo para administradores.

El punto fuerte de este proyecto, es poder dar de alta y ejecutar algoritmos que cumplan con una cierta clase de equivalencia entre sí, estos algoritmos están desarrollados en el lenguaje R, corriendo sobre una plataforma Java. Como se indicó en la secciones *3.2.2* y *3.3.3*, se estableció una clase de equivalencia para los algoritmos que pueden vivir en el sistema, ella indica que su entrada es una matriz $A_{nxm}$ y su salida otra matriz $B_{nxu}$, esta última incluyendo la entrada. Por motivos de performance previendo un análisis de datos futuro, se decidió agregar más carga sobre la estructura, intentando simplificar las consultas de la base de datos y poder distribuir los datos en tablas más específicas, y se estableció que por cada algoritmo dado de alta, se construya una tabla de forma dinámica la cual almacena como contenido todas las matrices $B_{nxu}$ obtenidas como la salida de las ejecuciones de un algoritmo dado. Además, esto haría mucho más sencilla la tarea de procesar datos, si es que en un futuro se quisieran analizar.

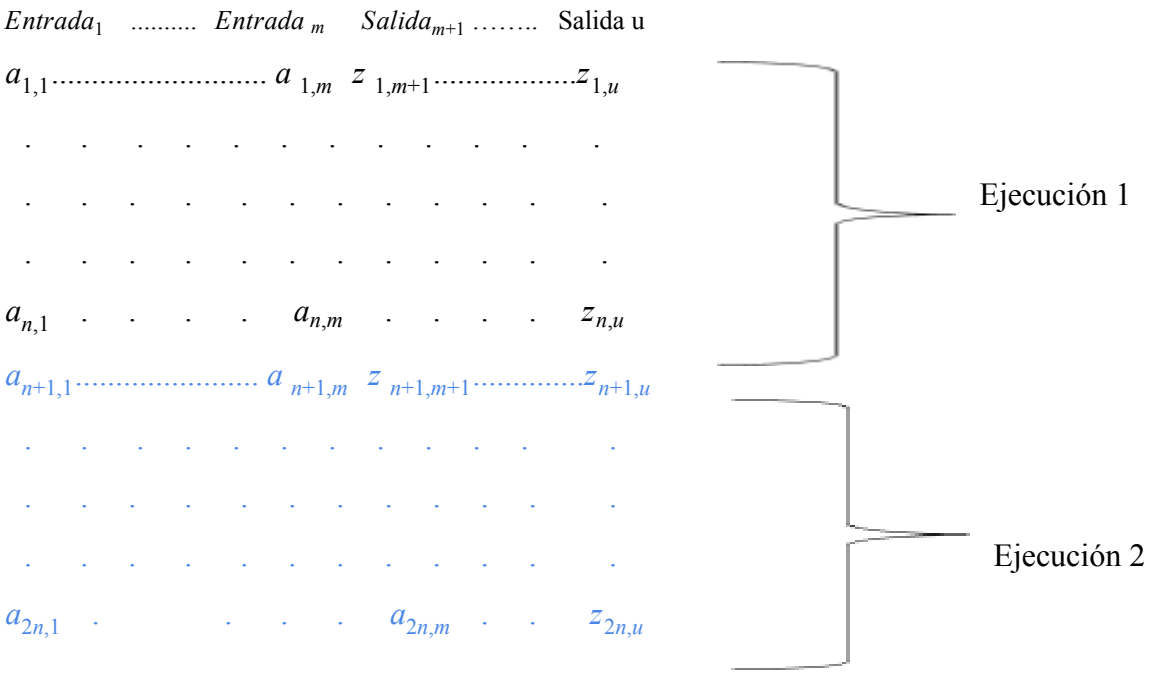
Por lo cual una ejecución con una matriz de entrada $A_{nxm}$ genera como salida una matriz

$B_{nxu}$ de la siguiente forma:




Matriz B (Matriz de salida)

Por consiguiente la tabla resultado asociada a dicho algoritmo es construida con las siguientes columnas, "$Entrada_1$ ..... $Entrada_m$ $Resultado_{m+1}$ …….. $Resultado_u$" almacenando cada matriz de salida generada desde distintas ejecuciones de la siguiente forma ordenada:



Otra decisión importante, es referente al formato_IO, este refiere a las entradas y salidas como bien se define en la sección **3.3.3**. La necesidad de estandarizar tanto las entradas como las salidas de los algoritmos que residen en el sistema, surge en un principio por un requerimiento de interés de los clientes que por temas de tiempo quedó fuera del alcance por ende se preparó para su futura implementación. Este requerimiento habla sobre poder enganchar algoritmos entre sí, en otra palabras que la salida de uno sea la entradas del siguiente, y este luego de ejecutar el primero algoritmo dejará precargadas las entradas del segundo algoritmo.

**Ejemplo: Alta y ejecución de algoritmo**

Supongamos que se tiene un algoritmo llamado SUMA que tiene los siguientes parámetros de entrada: ***entrada1*** de tipo entero, ***entrada2*** de tipo entero, y ***entrada3*** de tipo entero. Como salida tiene el valor ***salida*** de tipo entero, y es la suma de los tres valores de entrada.

Al dar de alta el algoritmo SUMA en el sistema, este le asocia un id. Digamos que se le asocia el id número 15 al algoritmo.. El sistema automáticamente crea la tabla en base de datos llamada *res_15* con las siguientes columnas:

- *id Integer*
- *ejecucion Integer* (clave foránea a la tabla donde se guarda información de la ejecución, como usuario que la ejecutó, fecha, lugar, etc)
- *entrada1 Integer*
- *entrada2 Integer*
- *entrada3 Integer*
- *salida Integer*

Supongamos que se realiza una ejecución del algoritmo de múltiples líneas de entradas con los siguientes valores:

- 1, 2 y 3
- 2, 3 y 4

Al terminar la ejecución se insertarán las siguientes dos filas en la tabla *res_15:*

- *(id1, id_ejecucion, 1, 2, 3, 6).*
- *(id2,id_ejecucion, 2 ,3 ,4 ,9).*

Siendo *id1* y *id2* identificadores distintos asignados por el sistema, y *id_ejecucion* un identificador asignado a la ejecución.

# 4. Seguridad

Al momento de desarrollar la aplicación, se intentaron cubrir los temas de seguridad básicos, dado que es una aplicación que se puede acceder desde Internet Pública, está expuesta a diferentes ataques que pueden afectar tanto el servidor como los datos de la aplicación.

Se tomaron en cuenta dos aspectos de seguridad relevantes:

**Control de acceso**

Este aspecto se usa para controlar quién accede al sistema. El sistema solo permite que usuarios registrados puedan acceder y solicitar distintas peticiones. Además, existen 2 tipos de usuarios diferentes (Admin, Común), que según su tipo, pueden acceder a partes del sistema distintas. Este control se desarrolló utilizando el **framework Spring Security** que está integrado con Spring y permite un manejo sencillo del contexto del usuario. También nos apoyamos en OAuth2 para registrar y poder cargar el contexto de un usuario en el sistema.

**SQL Injection**

Esto ocurre generalmente cuando se le pide al usuario que ingrese un input y en vez de ingresar un valor válido, ingresa una sentencia SQL que se ejecutará en la base de datos, pudiendo ocasionar daños importantes, como la pérdida total de datos. Para prevenir esto, se usó PreparedStatement de Java. Esto hace que los parámetros de entrada sean tratados como parámetros y no como sentencias sql, evitando así la inyección sql. En el caso de la creación dinámica de tablas para los resultados, se tomaron las siguientes precauciones:

1) El uso del servicio web, para dar de alta un nuevo algoritmo solo puede ser utilizado si en el contexto de la sesión del usuario están cargadas las credenciales de un administrador del sistema.

2) El nombre de las tablas creadas dinámicamente son res_(número de secuencia)

3) En el caso de las entradas, éstas son validadas en el frontend, no permitiendo la entrada de caracteres no alfanuméricos ni espacios y desde el back-end se implementa el mismo control.

# 5. Pruebas

Con el fin de evaluar la calidad y rendimiento del sistema, se plantean distintos escenarios de prueba teniendo en cuenta distintos algoritmos en R y variando sus respectivas entradas. También se plantea la prueba de un conjunto de paquetes de Renjin dentro de los cuales Renjin asegura su funcionamiento.

## 5.1 Objetivos

Se plantea como objetivo principal de las pruebas la evaluación del correcto funcionamiento del sistema teniendo en cuenta distintos escenarios. También se plantean distintas pruebas sobre Rserve y Renjin, considerando que este último no provee soporte a todas las posibilidades de R, para ello se solicitó a los directores que seleccionen un conjunto de algoritmos que sean de su interés para realizar dichas pruebas.

## 5.2 Prueba Funcional Sobre un Algoritmo Simple

Como una primera instancia de prueba, se selecciona un algoritmo, brindado por los clientes, el mismo es simple, es una función que permite obtener las dimensiones necesarias para calcular el volumen y la superficie de las figuras geométricas (shapes) que representan el fitoplancton.

```
# Input - un tipo de forma geometrica (character). Ej. "Ellipsoid"
# Output- Vector with a number of dimensions

dimensiones_formas<- function(shape=NULL){
    Common_shapes<-c("Sphere","Prolate_spheroid","Ellipsoid","Cylinder","Cone","Double_cone","Cone_half_sphere",
                     "Rectangular_box","Prism_on_elliptic_base","Prism_on_parallelogram_base", "cone_3cylinder")

    if(!shape%in%Common_shapes) stop(paste("La forma geometrica", shape, "no coincide con ninguna de la base de datos"))
    Necessary_Dimensions<-c(1,2,3,2,3,3,3,3,3,3,6)# Dimensiones necesarias para calcular el volumen de esas formas geometricas
    #Este vector es correlativo con el vector de "Common_shapes" e indica el numero de medidas requeridas para calcular vol. y area
    selShape<-which(Common_shapes==shape)

    return(Necessary_Dimensions[selShape])
}#End function
```

Para realizar las pruebas funcionales de este algoritmo, se propusieron 3 pruebas simples, seleccionando 3 Shapes, y comparando el resultado de sus dimensiones con la dimensión esperada, el resultado fue el siguiente:

**1. Test Cases**

| ID | Title | Assigned To | Status |
|---|---|---|---|
| T1 | Ingreso de Shape tipo Sphere, salida esperada 1 | Santiago N. | Passed |
| T2 | Ingreso de Shape tipo Cone, salida esperada 3 | Santiago N. | Passed |
| T3 | Ingreso de Shape tipo Rectangular_Box, salida esperada 3 | Santiago N. | Passed |

Figura 10

### 5.3 Prueba Funcional Sobre un Algoritmo con Paquete Externo y un R-Data

El siguiente algoritmo calcula la probabilidad de que haya floración, devolviendo dicha probabilidad en la salida True y su complemento en la salida False (la probabilidad de que no haya).

```
require(randomForest);
load('RandomForest_COLONY_HAB_Segura_etal2017.RData');

VientoInt_ms1 <- variables[,'VientoInt_ms1']
T_oC    <- variables[,'T_oC']
Turbidez_NTU    <- variables[,'Turbidez_NTU']
Salinidad <- variables[,'Salinidad']
data_prueba <- data.frame(VientoInt_ms1,T_oC,Turbidez_NTU,Salinidad)

CLASIF = predict(rf_FINAL, newdata=data_prueba[1:4], type="prob");

falso = CLASIF[,1];
verdadero = CLASIF[,2];
```

Para esta prueba se toma como objetivo el correcto funcionamiento tanto del paquete incluido como el R-Data, por lo cual se toma como un conjunto de entradas las cuales son corridas en R nativo, y registramos su salida. Luego se procede a hacer la prueba en el proyecto tanto con Renjin y Rserve y se comparan los resultados con R nativo.

1. Test Cases

| ID | Title | Assigned To | Status |
|---|---|---|---|
| T6 | RandomForest Entrada: (VientoInt_ms1: 20, T_oC: 12, Turbidez_NTU: 3, Salinidad 0.5) Salida esperada: ( Falso 0.161, Verdadero: 0.839 ) | Santiago N. | Passed |
| T7 | RandomForest Entrada: (VientoInt_ms1: 20, T_oC: 26, Turbidez_NTU: 1, Salinidad 0.6) Salida esperada: ( Falso 0,176, Verdadero: 0,824 ) | Santiago N. | Passed |

Figura 11

### 5.4 Pruebas de Renjin

Renjin tiene como punto fuerte que su utilización en un proyecto es relativamente más fácil que otras opciones evaluadas y mantiene una comunidad activa a la fecha de junio de 2018, pero como gran punto en contra es el hecho de que no todos los paquetes están disponibles y algunos son solo confiables en ciertas condiciones. Anteriormente se utilizó Random Forest para realizar una prueba en la aplicación, este paquete resulta un caso interesante para estudiar dado que no todas las pruebas que realizó la comunidad de Renjin pasaron.

A continuación hablaremos brevemente sobre las pruebas realizadas por parte de Renjin sobre dicho paquete.

El paquete Random Forest puede ser cargado por Renjin, pero no existe garantía de su funcionamiento, como sí la hay en otros paquetes, dado que fallaron 7 de 20 pruebas. De todos modos, gran parte de las fallas se debieron a que se usan funciones como MDSplot y tuneRF. Estas funciones se relacionan con la manipulación de gráficas en R, factor que la aplicación desarrollada no soporta, y al utilizarse el motor de R da una excepción en la evaluación del código.

# 6. Cierre de Proyecto

A modo de cierre, se participó en conjunto con Carla Kruk y Ángel Segura exponiendo el proyecto realizado en Ingeniería de Muestra 2018 [16] en el CURE, Rocha. Con el agregado de un conjunto de sensores para simular la obtención de muestras, los mismos permiten medir temperatura, salinidad y turbidez de distintas muestras obtenidas en el Lago del Parque Rodó, la Laguna de Rocha y Laguna del Diario. Estos datos son ingresados en la aplicación, dentro del algoritmo de predicciones MAC y se obtiene una probabilidad de existencia de floración indicando si existe riesgo, o simplemente se debe controlar.

En el caso de la Laguna del Diario la probabilidad obtenida luego de la ejecución del algoritmo fue del entorno de 0.96 - 0.97 y en el caso de la laguna de Rocha una probabilidad en el entorno de 0.136 y 0.30.

Haciendo mención del valor obtenido para la Laguna del Diario puede decirse que es un valor alarmante, pero ya conocido dado el deterioro de la misma, ya que si la vegetación impide el pasaje de luz del sol hacia la profundidad, no se cumple la fotosíntesis y por lo tanto no hay oxígeno acabando con la vida de algunas especies.

# 7. Resultados

En este punto del Informe, se resume la planificación del trabajo realizado y los tiempos reales que llevó en las distintas etapas del proyecto, se presentan lecciones aprendidas y finalmente las conclusiones del trabajo realizado.

## 7.1 Planificación y ejecución

Se planificaron 5 etapas con la intención de realizarlas en 1 año, las mismas son:

1) Adquirir conocimientos teóricos, introducirnos en el lenguaje R.
2) Relevar información
3) Prototipos
4) Implementación
5) Liberar en el ambiente del los directores de MAREN

A continuación la duración planificada en verde contra la duración real en rojo puede observarse en las figuras 12 y 13.

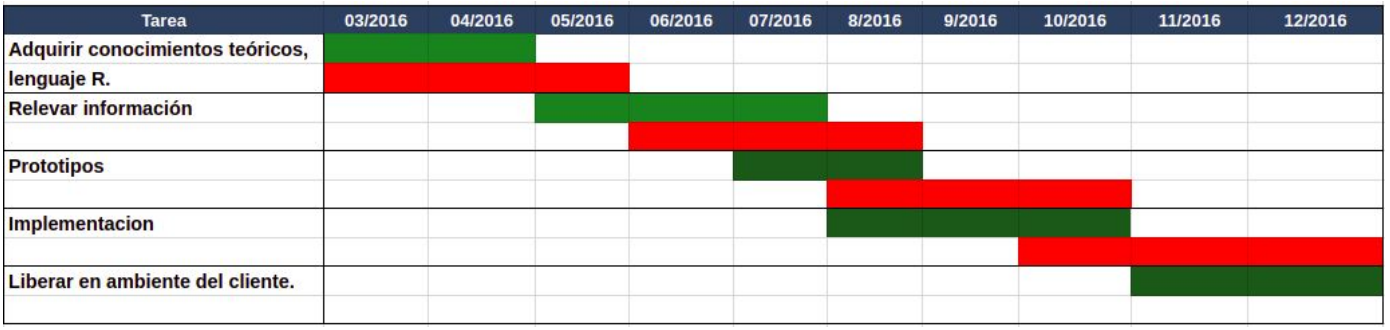



Figura 12


Figura 13

### Tareas realizadas en 2018


Figura 14

**Adquirir conocimientos teóricos, introducirnos en el lenguaje R.**

La primera fase involucra todo aquello que se entiende como marco básico para poder comenzar a entender la realidad planteada por los directores y contar con herramientas para poder relevar los requerimientos del proyecto. Teniendo en cuenta que los directores estaban ubicados en la ciudad de Rocha, se tomó la decisión de extender el período para ir mejor preparados a la primer reunión, para ello se estudió el uso del lenguaje R y además se le solicitó a los directores documentación sobre el negocio.

**Relevar información**

Durante esta etapa, se pactó una primera reunión presencial con los directores, para posteriormente continuar a distancia y eventualmente pactar nuevas reuniones. Se puede acotar que el relevamiento de requerimientos no fue una tarea relativamente complicada dada la contraparte científica y con conocimientos informáticos de los directores.

**Prototipos**

Para poder validar los requerimientos con los directores y evaluar la viabilidad de uso de ciertas tecnologías como rJava o Renjin, o mismo antes de abordar la solución con Java, como la posibilidad que se evaluó de usar Ruby.

Se fabricaron cuatro prototipos, uno con Ruby y otros tres con Java, usando RinRuby, rJava, Renjin y Rserve respectivamente.

**Implementación**

La implementación duró más de lo previsto, si bien la aplicación estaba casi terminada en su totalidad para los primeros 3 meses de esta etapa, se plantearon varios cambios en la usabilidad sumados a idas y venidas para ajustar detalles con los directores.

**Liberar en el ambiente de los directores de MAREN**

La liberación se realizó en un servidor proporcionado por el tutor con un ambiente Linux, sumado a una petición de dominio para su mejor acceso. Durantes esta etapa, surgieron idas y venidas con los directores, dado que se empezaron a correr más algoritmos en R sobre la plataforma, en este periodo se encontraron errores los cuales fueron reparados.

En esta etapa pudimos ver que algunos algoritmos de interés de los directores no eran compatibles con Renjin.

En este sentido se decidió dedicar un mes más fuera de lo estimado para intentar integrar una segunda opción de motor para R, optamos por integrar la aplicación con Rserve, obteniendo muy buenos resultados, por lo cual dejamos la misma como opción por defecto, y se habilitó la opción para que los administradores de la aplicación puedan elegir con qué motor de R correr una prueba, sea con Renjin o con Rserve.

### 7.2 Conclusiones

Se desarrolló una plataforma web que cumple con los objetivos solicitados dentro del alcance acordado y cubriendo las funcionalidades esperadas, permitiendo ejecutar algoritmos codificados en R (con una clase de equivalencia pre establecida) en la web, procesando los datos ingresados como entrada y devolviendo datos de salida. Se realizaron pruebas sobre los algoritmos de mayor interés para los directores, las cuales fueron exitosas. Se puede concluir que el proyecto logró el cumplimiento de los objetivos planteados.

Cabe destacar que existe un pequeño grupo de códigos R que no podían ser ejecutados en el sistema, dado que usan paquetes R que todavia no estan disponibles, o no son confiables, en la biblioteca Renjin, esto fue solucionado agregando el motor de R, Rserve.

En cuanto a la planificación del proyecto tuvo una duración mayor a la prevista. El estudio previo que requirió la etapa 1 causó que desde el comienzo el proyecto se desfasara.
Posteriormente la etapa de desarrollo y liberación contribuyeron en mayor medida a este desfasaje. Si bien en septiembre de 2017 culminó la mayor parte del proyecto, luego se dedicó a documentar, cuando se alcanzó un nivel de madurez suficiente se realizó un seminario con los usuarios finales y los directores (administradores del sistema), en el CURE, exponiendo la aplicación y realizando una jornada con su uso.

Como punto final respecto al equipo y forma de trabajo, se puede destacar que los tres integrantes que conformaron el grupo ya habían trabajado juntos en proyectos anteriores, por lo cual ya se contaba con una forma de trabajo establecida y eficiente. Con respecto a los directores, el trato fluyó sin problemas, se destaca la buena disposición e interés de los mismos.

# 8. Trabajo futuro

## 8.1 Enganche de Algoritmos

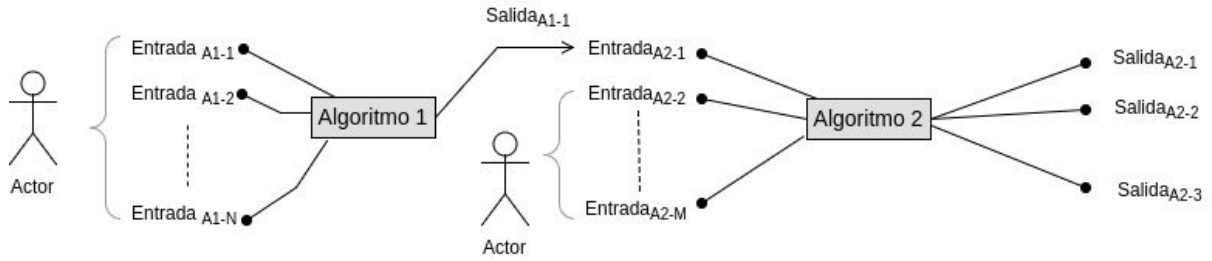



Figura 15

Como se habló en la sección de requerimientos, la idea es que se puedan ejecutar algoritmos en secuencia, esto quiere decir que desde un algoritmo con N entradas, ingresadas por el usuario, termine con la invocación a otro algoritmo donde se pre cargan en sus entradas los resultados del primer algoritmo y dando la posibilidad al usuario de completar las entradas restantes.

Para facilitar la futura implementación de este requerimiento, se estandarizó la forma en que se modela las entrada y salidas, para poder completar este requerimiento se debería hacer foco en modificar el ABM de Algoritmo y su impacto en la presentación al usuario final.

## 8.2 Compartir fotos

Esta funcionalidad implica la posibilidad de compartir fotos desde un dispositivo móvil con su información geográfica asociada. El objetivo es que los usuarios puedan compartir imágenes de los recursos hídricos junto con una clasificación del estado de los mismos. De esta manera sencilla se permite que los usuarios brinden información temprana sobre concentraciones anormales de microalgas. Yendo un paso más adelante se podría realizar un análisis automático de las imágenes, lo que podría remplazar o complementar la clasificación manual hecha por el usuario.

Esta funcionalidad, involucra todo un módulo nuevo a desarrollar, y es independiente a la implementación anterior, dada la forma en que fue diseñado no existirían mayores complicaciones para su integración.

### 8.3 Multilenguaje

Más allá de que los usuarios iniciales de la aplicación son de habla hispana, se pretende que la aplicación tenga mayor alcance en el futuro y se use en otras partes del mundo, por lo que el soporte de otros lenguajes como inglés es necesario.

Una posible solución a este requerimiento sería identificar todos los textos con una etiqueta y obtener su valor desde un archivo de configuración, pudiendo existir varios de estos indicando cada uno un lenguaje específico.

### 8.4 Mejoras de Usabilidad

Como todo sistema, este debe ir evolucionando a lo que el usuario final del mismo considere más práctico y cómodo de usar. Por medio de un feedback continuo la herramienta podría ir mejorando sus interfaces y interacción con el usuario.

### 8.5 Rango de Datos

Es de interés para los directores poder acotar el rango de datos en las entradas, siendo el fin darle al usuario final un control más sobre posibles datos incorrectos o inconsistentes ante el algoritmo definido en R a utilizar. Su implementación demandará modificar el módulo de Alta de Algoritmos y el esquema de datos, para que según el tipo de dato de entrada, este se pueda acotar por valores que el usuario administrador defina.

### 8.6 Scheduler de ejecuciones

Dado que los recursos son limitados y la ejecución de un algoritmo puede ser costosa según el mismo, la implementación de un scheduler de ejecuciones, podría evitar problemas a futuro. Se definiría una cola de espera, donde se ubican todas las tareas (ejecuciones de algoritmos) a ser corridas por el scheduler cada cierto tiempo, una vez procesada la tarea, se le informará al usuario el resultado.

Hay que destacar que si bien esta opción es buena para evitar que colapse el sistema por falta de recursos, se estaría obteniendo una penalización grande para el usuario final en términos de usabilidad, dada la espera que este tendría hasta obtener los resultados de su ejecución.

### 8.7 Gráficas para los resultados

Los usuarios indicaron que obtener una gráfica a partir de un conjunto de resultados obtenidos en una ejecución, como por ejemplo cruzar entradas con salidas, seria de gran utilidad para el usuario final, dado que permitiría una comprensión más rápida de los resultados obtenidos. Otro ejemplo es la clasificación de información según algunos de los datos de salida o entrada ingresados para realizar la ejecución del algoritmo.

Para poder implementar este requerimiento, se tendría que revelar qué tipo de gráfica usar y en qué casos son de interés, para poder dar la opción de graficar la matriz de resultados obtenida.

### 8.8 Medios de autenticación

Actualmente la autenticación se puede realizar con gmail o brindando email y contraseña. Si fuera pertinente o se identificara que nuevos usuarios son más afines a otro medio de autenticación como una red social, sería sencillo implementar una integración para permitirlo.

### 8.9 Uso de rJava

Como se ha explicado antes se optó por Renjin, posteriormente se sumó Rserve para realizar la integración de R en Java. La ventaja de Renjin es que al ser una biblioteca de Java no es necesario resolver problemas de concurrencia al utilizar el módulo. La desventaja es que no todos los packages de R están soportados por la biblioteca, por lo que en ciertas ocasiones el usuario puede verse limitado a lo estrictamente contenido en Renjin. Si bien esto fue cubierto al integrar la opción de Rserve se podría plantear en un futuro la utilización de más opciones de integración, como lo es rJava. Se debe considerar que al utilizar la misma, se está usando el entorno de R nativo instalado en el servidor, por lo que es responsabilidad de la aplicación resolver los problemas de concurrencia, ya que solo un algoritmo podría correr en paralelo. Este punto de trabajo a futuro se puede relacionar con el scheduler de ejecuciones mencionado anteriormente, ya que para usar rJava contemplando los posibles problemas de concurrencia se puede implementar un scheduler que maneje el acceso a R de las diferentes ejecuciones.

### 8.10 Seguridad

Cabe destacar que la seguridad en una aplicación web como ésta, expuesta a internet, debe ser un proceso continuo. Durante su desarrollo se tomaron las precauciones correspondientes, utilizando Spring Segurity, Auth2, y control de inyecciones, estas consideraciones y más deberían seguir siendo evaluadas en futuros desarrollos.

# 9. Anexos

Documento de Instalación - Ir al documento

Manual para Administrador - Ir al documento

# 10. Referencias

1- Santana, J. (2014) - El arte de programar en R: un lenguaje para la estadística

2 -Contreras Gracia, J (2007) - Introducción a la programación estadística con R para profesores

3 - Spring Framework; Acceso Agosto 7, 2016; https://spring.io

4 - RStudio;Acceso Noviembre 11, 2018; https://www.rstudio.com

5 - Shiniy; Acceso Noviembre 11, 2018; http://shiny.rstudio.com/

6 - rApache; Acceso Noviembre 13, 2017; http://rapache.net/manual.html

7 - RJava; JRI/RJava; Acceso Julio 15, 2017; https://www.rforge.net/JRI/

8 - RServe; Acceso Noviembre 11, 2018; http://www.rforge.net/Rserve/

9 - Renjin; Acceso Noviembre 11, 2018; http://www.renjin.org/

10 - RCaller; Acceso Noviembre 11, 2018; https://github.com/jbytecode/rcaller

11 - RHadoop; Acceso Noviembre 11, 2018; https://github.com/RevolutionAnalytics/RHadoop/wiki

12 - RinRuby; Acceso Junio 8, 2016 https://sites.google.com/a/ddahl.org/rinruby-users/

13 - Rserve; Acceso Septiembre 11, 2018; https://www.r-project.org/conferences/DSC-2003/Proceedings/Urbanek.pdf

14 - R; Acceso Septiembre 11, 2018; http://www.r−project.org

15 - OAuth 2; Acceso Noviembre 11, 2018; https://digitalocean.com/community/tutorials/una-introduccion-a-oauth-2-es

16 - Ingeniería de Muestra 2018; Acceso Noviembre 11, 2018; http://idm.fing.edu.uy/

17 - Jun Sun and Dongyan Liu (2003) - Geometric models for calculating cell biovolume and surface area for phytoplankton

18 - Simon Urbanek (2003) - Rserve, A Fast Way to Provide R Functionality to Applications