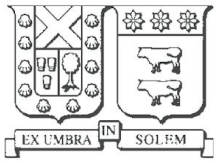
Departamento de Informática
Universidad Técnica Federico Santa María



Informe de Proyecto – INF-225-2018-1-CC
Proyecto "Valoración de Opciones Sobre Acciones"
3 de Agosto, 2018

Integrantes:

Nombres y Apellidos	Email	ROL USM
Jose Luis Gallardo P.	jose.gallardo.14@sansano.usm.cl	201430003-4
Marcelo González H.	marcelo.gonzalez.14@sansano.usm.cl	201430028-K
Esteban Jara C.	esteban.jara.13@sansano.usm.cl	201330002-2

Índice

1. Requisitos clave (Actualizado)

Los requisitos tanto funcionales como no funcionales actualizados se presentan en las tablas 1 y 2, respectivamente.

Req. funcional	**Descripción y medición**
Generación de trayectorias	El sistema debe permitir generar simulaciones que obtienen posibles valores para las opciones de acción en forma de trayectorias.
Selección de parámetros para análisis de acciones	El sistema debe permitir la selección de la acción, periodo de tiempo, tipo, trayectorias y número de trayectorias para obtener posibles valores de las acciones.
Recopilación de valores de acciones	El sistema debe guardar un historial con el valor de las acciones consultadas.

Tabla 1: Requisitos Funcionales (FR).

Req. extra-funcional	**Descripción y medición (máximo 2 líneas)**
Serialización	Serialización para separar información de cada usuario
Importación de datos (online como offline)	El sistema debe permitir la importación de datos financieros desde un archivo .csv o la API de Yahoo Finance.
Gráficos de comportamiento de acciones	El sistema debe mostrar un gráfico con el valor de la acción en función del tiempo.
Gráficos de valoración y trayectorias	El sistema debe mostrar un gráfico de valoración de acción v/s número de trayectorias utilizadas.
Plataforma “Desktop”	El sistema debe ejecutarse en un ambiente “Desktop” y sin la necesidad de conectarse a internet para funcionar.
Tiempo de respuesta	Tiempo de respuesta de consultas < 20s
Desempeño	El sistema debe realizar los cálculos numéricos utilizando el lenguaje de programación R

Tabla 2: Requisitos No Funcionales (NFR).

2. Árbol de Utilidad (Actualizado)

El árbol de utilidad actualizado se presenta en la figura 1. Este contiene la modificación de los requisitos: ahora se presentan los NFRs mientras que en la entrega anterior se presentaron los FR.

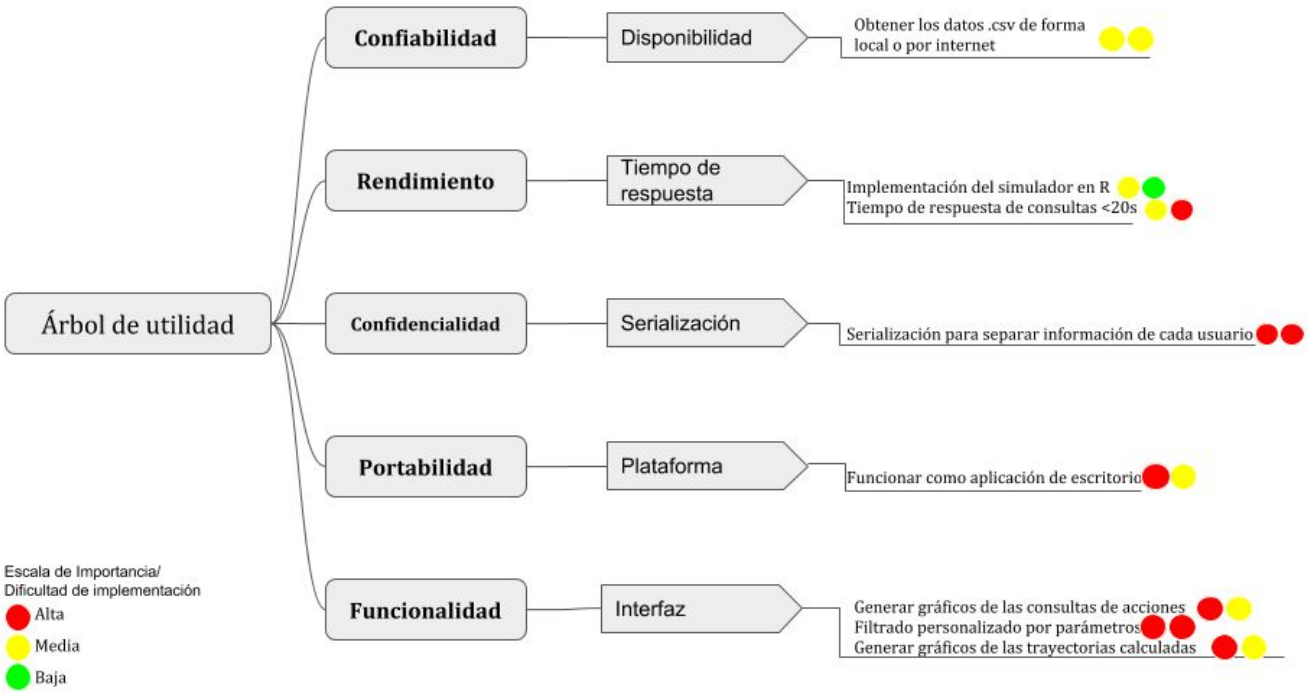


Figura 1: Árbol de utilidad del proyecto.

3. Modelo de Software

A continuación (Figura 2) se muestra el diagrama de clases del proyecto con las clases relevantes. Note que las clases con el tag <<component>>, a la hora de implementación, son clases que heredan de React.Component (por ello todas tienen el método render).

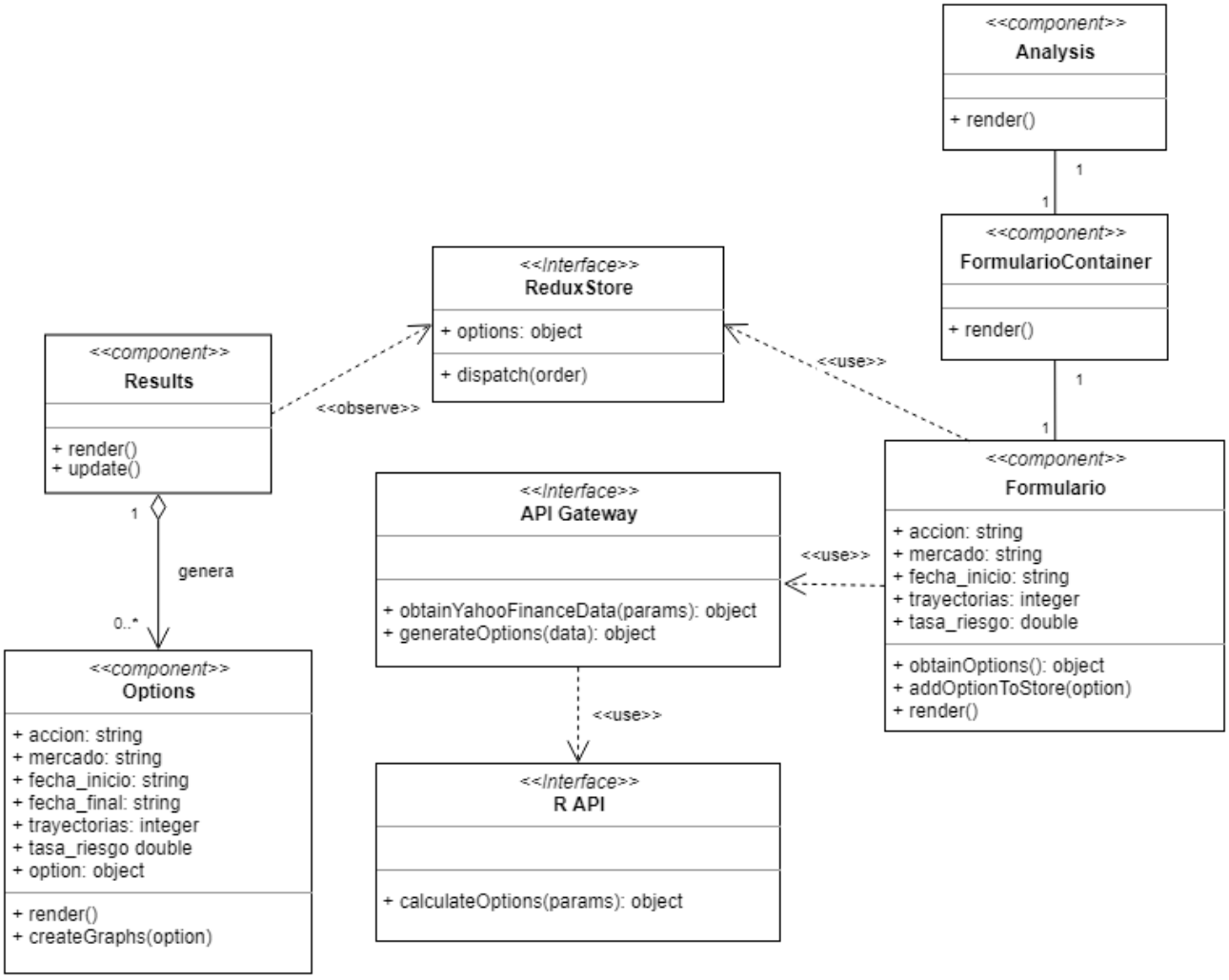


Figura 2: Diagrama de Clases del proyecto.

Intención	**Patrón de Diseño**	**Razonamiento**
Generar historial de las consultas hechas por el consumidor	Observer	Se necesita actualizar el estado del componente que muestra los gráficos de las consultas hechas por el usuario (Results) a medida que se hagan solicitudes en el componente Formulario. La implementación de hacer aquello mediante comunicación directa entre los componentes involucrados en React implica una tediosa implementación y difícil debugeo, por lo que se dispondrá de una store que guarde los datos analizados, así Results, mediante la observación de la store, puede detectar cambios en los datos y luego actualizar automáticamente la vista.
Encapsular implementación de algoritmo de extracción de datos obtenidos por Yahoo! Finance, transformación y tratamiento de datos.	Gateway	Se necesita encapsular la interacción tanto con la API de Yahoo Finance como con la API de R dado que no se desea mezclar la lógica de muestra de datos de la vista con la lógica de extracción, transformación y tratamiento de datos. Para ello, se creó una API (que es el backend de la aplicación) que se encarga de la última lógica mencionada, así sólo basta que se haga consulta a esa API con los parámetros necesarios para entregar el resultado final.

Tabla 2: Selección de Patrones

4. Trade-offs entre tecnologías

El Softgoal Interdependency Graph (SIG) se presenta en la figura 2.

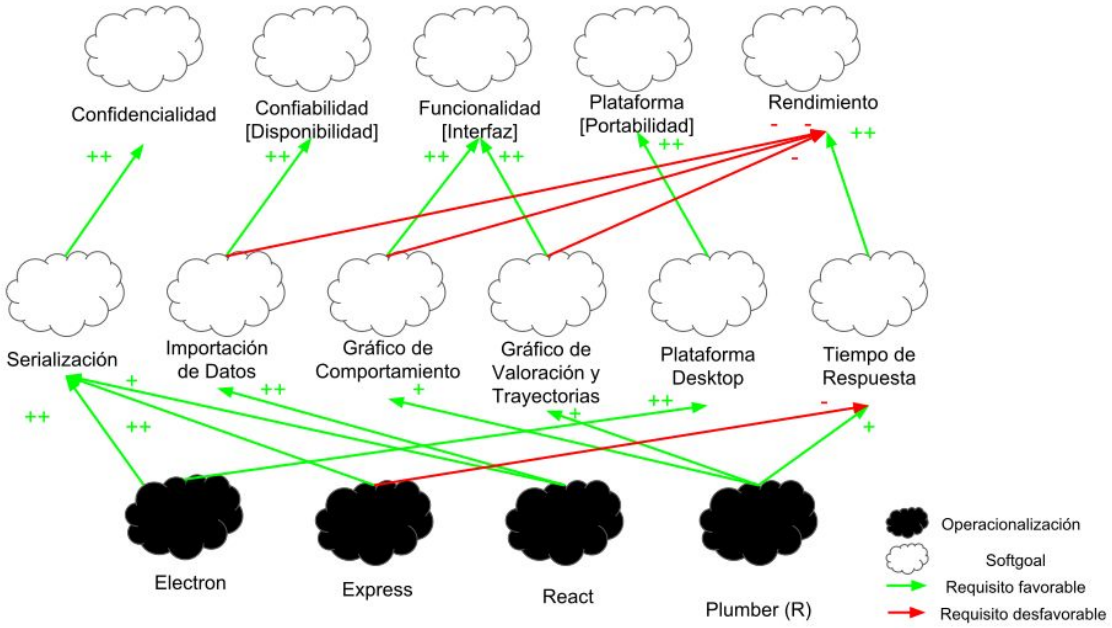


Figura 3: Softgoal Interdependency Graph (SIG).

Los Trade-offs entre las opciones tecnológicas utilizadas se muestra en la siguiente tabla.

Decisión	**Softgoal**	**Evaluación**	**Razonamiento**
Serializar para separar la información de cada usuario	Confidencialidad	++	Al establecer un login de usuario mediante una “serial key”, asociada a cada usuario, se está resguardando la privacidad de los datos del mismo.
Serialización mediante Electron	Confidencialidad	++	Al establecer un “serial key” mediante Electron, y si no se tiene el correcto, la aplicación no seguirá el curso normal (no se genera la ventana de carga de datos).

Serialización mediante Express	Confidencialidad	++	Si no se tiene la llave al iniciar la aplicación, esta no seguirá el curso normal (no transiciona a la vista de carga de datos).
Serialización mediante React	Confidencialidad	+	Si bien contribuye a la seguridad, poco ayudaría (y no tendría sentido) que el Front End de la aplicación maneje dicha acción.
Importación de datos mediante archivo (tanto online como offline)	Confiabilidad (Disponibilidad)	++	Si no se tiene conexión a internet, es posible realizar las simulaciones consultando la API de R (offline).
Importación de datos vía React	Confiabilidad (Disponibilidad)	++	Permite el uso de una UI sencilla para el usuario que quiera cargar su archivo .csv para analizar.
Importación de datos	Rendimiento	-	Poco podría ayudar al rendimiento considerando lo extenso, largo o "pesado" que pueden llegar a ser los archivos que cargue el usuario.
Gráfico de Comportami-ento ayuda a tener una UI más amigable	Funcionalidad (Interfaz)	++	El tener el gráfico en la UI permite al usuario ver que el programa, y su simulación, funcionan correctamente. Además, es útil tener dicha información de inmediato.
Gráfico de Valoración y Trayectorias ayuda a tener una UI más amigable	Funcionalidad (Interfaz)	++	Mismo razonamiento anterior.
Gráfico de comportami-ento	Rendimiento	-	Poco ayudaría el tener un gráfico con los resultados en cuanto a tiempos de carga del mismo en la

			aplicación y/o en la obtención el mismo vía la API de R o generación vía JavaScript.
Gráfico de Valoración y Trayectorias	Rendimiento	-	Mismo razonamiento anterior.
Uso de Plumber (R) para generar gráficas	Funcionalidad (Interfaz)	++	Si las gráficas se generan al llamar a la API de R (después de simular), sería bastante rápido ocupar los mismos y solamente mostrarlos por pantalla.
Portar la aplicación a formato “Desktop” mediante Electron	Plataforma (Portabilidad)	++	Al ocupar un framework que permite portar la aplicación a Windows, Mac o Linux, simplifica bastante la programación al no tener que desarrollar código para cada sistema operativo.
Despliegue de resultados vía Electron	Rendimiento	-	Si los servicios utilizados demoran en responder de vuelta, la aplicación demorará en responder también.
Uso de Plumber (R) para analizar datos	Rendimiento	++	El cliente mismos decidió que todos los cálculos se realicen utilizando R debido a su precisión numérica y rapidez.

Tabla 3: Trade-offs entre opciones tecnológicas.

JGP - MGH - EJC