# ESCUELA POLITÉCNICA NACIONAL

## FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

## DESARROLLO DE UN PROTOTIPO DE APLICACIÓN WEB INTERACTIVA PARA LA SIMULACIÓN DE LAS PRINCIPALES DISCIPLINAS DE PLANIFICACIÓN DE PAQUETES PARA LA GESTIÓN DE QOS

**TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÓNICA Y REDES DE INFORMACIÓN**

**BRYAN FERNANDO SIMBAÑA PINEIDA**

**DIRECTOR: ING. LUIS FELIPE URQUIZA AGUIAR, PhD.**

**Quito, febrero 2022**

# AVAL

Certifico que el presente trabajo fue desarrollado por Bryan Fernando Simbaña Pineida, bajo mi supervisión.

**LUIS FELIPE URQUIZA AGUIAR, PhD**
**DIRECTOR DEL TRABAJO DE TITULACIÓN**

# DECLARACIÓN DE AUTORÍA

Yo, Bryan Fernando Simbaña Pineida, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración dejo constancia de que la Escuela Politécnica Nacional podrá hacer uso del presente trabajo según los términos estipulados en la Ley, Reglamentos y Normas vigentes.

BRYAN FERNANDO SIMBAÑA PINEIDA

# DEDICATORIA

Dedico este Proyecto de Titulación a mi familia, quienes han sido una fuente de constante apoyo y aliento durante los desafíos que se me han presentado. Especialmente a mis padres, Carlos y Cristina, quienes con sus buenos ejemplos me han enseñado a trabajar duro para cumplir mis objetivos.

Bryan

# AGRADECIMIENTO

Estoy profundamente agradecido con mis padres, quienes siempre han estado a mi lado para apoyarme y guiarme en el transcurro de mi vida para poder alcanzar con éxito los objetivos que me he planteado.

De igual manera, un agradecimiento especial a mi director de tesis Luis Urquiza, PhD. por todo el apoyo y la paciencia que me ha brindado para el desarrollo exitoso de este Proyecto de Titulación.

# ÍNDICE DE CONTENIDO

# ÍNDICE DE FIGURAS

# ÍNDICE DE TABLAS

# ÍNDICE DE CÓDIGOS

# RESUMEN

El presente Proyecto de Titulación tiene como objetivo principal desarrollar un prototipo de aplicación web interactiva para la simulación de las principales disciplinas de planificación de paquetes para la gestión de la calidad de servicio (QoS) de las redes de telecomunicaciones.

El prototipo de aplicación web fue desarrollado utilizando el lenguaje de programación R y utiliza el *framework Shiny* para la creación de los componentes interactivos. Para la exploración y visualización de los datos de forma interactiva se utilizó los paquetes de gráficos Plotly y Esquisse. El prototipo está alojado en el servicio de almacenamiento en la nube de shinyapps.io.

En el Capítulo uno se describen los fundamentos teóricos de los temas esenciales para el desarrollo del Proyecto de Titulación. Además, se expone de forma breve un resumen sobre las tecnologías utilizadas y la metodología ágil de desarrollo de software Kanban. En el Capítulo dos se realiza el análisis de los requerimientos funcionales y no funcionales obtenidos a partir de las entrevistas. En base a los requerimientos se elabora el diseño del prototipo, y se realiza su implementación. En el Capítulo tres se realiza la validación los resultados generados por el prototipo. Además, se muestra los resultados de las pruebas realizadas. Finalmente, en el Capítulo cuatro se presentan las conclusiones y recomendaciones obtenidas tras la culminación del desarrollo del prototipo.

**PALABRAS CLAVE:** aplicación web interactiva, disciplinas de planificación, Shiny, Plotly, Esquisse.

# ABSTRACT

The main objective of this Degree Project is to develop a prototype of an interactive web application for the simulation of the main packet scheduling disciplines for the management of the quality of service (QoS) of telecommunication networks.

The prototype web application was developed using the R programming language and uses the Shiny framework for the creation of the interactive components. Plotly and Esquisse graphics packages were used for interactive data exploration and visualization. The prototype is hosted in the cloud storage service shinyapps.io.

Chapter one describes the theoretical foundations of the essential topics for the development of the Degree Project. In addition, a summary of the technologies used, and the agile Kanban software development methodology is presented. In Chapter two the analysis of the functional and non-functional requirements obtained from the interviews is made. Based on the requirements, the prototype design is elaborated, and its implementation is carried out. Chapter three validates the results generated by the prototype. In addition, the results of the tests performed are shown. Finally, Chapter Four presents the conclusions and recommendations after the completion of the prototype development.

**KEY WORDS**: interactive web application, scheduling disciplines, Shiny, Plotly, Esquisse.

# 1. INTRODUCCIÓN

La planificación de paquetes es una herramienta clave para brindar garantías de rendimiento a las aplicaciones que lo requieran [1]. Obtener una compresión profunda de los conceptos fundamentales de las disciplinas de planificación de paquetes es necesario para construir QoS en Internet. Aunque en la red hay disponible una gran variedad de software especializado para simular el comportamiento de las disciplinas de planificación, estos requieren un conjunto de habilidades adicionales para familiarizarse con su uso, además los resultados que se obtienen muchas veces no son fáciles de interpretar por el estudiante [2]. Esto disminuye considerablemente la motivación de los estudiantes y plantea un desafío para comprender y apreciar los procesos involucrados en el funcionamiento de las disciplinas de planificación.

Los avances en las tecnologías de la información y la comunicación (TIC) han permitido el desarrollo de herramientas y sistemas con el fin de mejorar las experiencias de enseñanza y aprendizaje. Una de estas herramientas son las aplicaciones interactivas, que han permitido aumentar la motivación y participación de los estudiantes en la adquisición de nuevos conocimientos y habilidades [2].

Teniendo esto en cuenta, en el presente Trabajo de Titulación se plantea el desarrollo de un prototipo de aplicación web interactiva para la simulación de las principales disciplinas de planificación de paquetes. Este prototipo servirá como un complemento para las actividades tradicionales de enseñanza, así como material de apoyo para facilitar el aprendizaje y la participación de los estudiantes de las carreras de telecomunicaciones o carreras afines.

## 1.1. OBJETIVOS

El objetivo general de este Proyecto Técnico es:

- Desarrollar un prototipo de aplicación web interactiva para la simulación de las principales disciplinas de planificación de paquetes para la gestión de QoS.

Los objetivos específicos del Proyecto Técnico son:

- Analizar los conceptos fundamentales de calidad de servicio, disciplinas y algoritmos de planificación de paquetes, y su implementación en R.

- Diseñar los módulos del prototipo de aplicación web de acuerdo con las características planteadas.
- Implementar los módulos de la aplicación web y su funcionalidad de acuerdo con el diseño realizado.
- Analizar los resultados de la simulación del prototipo mediante la aplicación de las pruebas correspondientes.

## 1.2. ALCANCE

En el presente proyecto de titulación se detalla el desarrollo de un prototipo de aplicación web interactiva para la simulación de las principales disciplinas de planificación de paquetes para la gestión de QoS en redes de telecomunicaciones. La aplicación estará compuesta de varias interfaces gráficas de usuario, cada una con diferentes funcionalidades para que el usuario interactúe con ellas. Por medio de las interfaces gráficas, los usuarios podrán configurar los parámetros de simulación, para posteriormente seleccionar la disciplina deseada, simular, y observar los resultados generados mediante reportes, gráficos interactivos y tablas dinámicas.

La aplicación se desarrollará completamente en R, utilizando *Shiny*, un paquete R de código abierto que proporciona un potente *framework* web para la creación de aplicaciones web interactivas directamente desde R. *Shiny* es altamente personalizable, por esta razón se utilizará HTML, CSS y JavaScript para realizar pequeños cambios en las interfaces gráficas y su comportamiento.

El prototipo de aplicación web se alojará en la plataforma shinyapps.io, un servicio en línea para alojar aplicaciones *Shiny* en la nube. Los usuarios podrán acceder a la aplicación desde cualquier navegador web, como Mozilla Firefox, Google Chrome o Microsoft Edge.

De forma general, los módulos que integrarán el prototipo de aplicación web son los siguientes: configuración de parámetros, disciplinas de planificación y gráficos avanzados.

- **Módulo de configuración de parámetros**: Este módulo contará con una interfaz de usuario con varios campos numéricos y de selección en los que el usuario podrá configurar los diferentes parámetros necesarios para la simulación de cada disciplina implementada. Además, contará con una sección en donde se mostrará un resumen de la información ingresada por el usuario, de esta forma el usuario podrá observar todos los valores configurados y modificarlos de ser necesario.

- **Módulo de disciplinas de planificación**: Este módulo contará con varios submódulos, uno por cada disciplina de planificación implementada. Cada submódulo implementará una interfaz de usuario, en la cual estarán disponibles varios componentes interactivos para la exploración y visualización de los resultados de la simulación. Además, el usuario tendrá la opción de generar reportes de los resultados de la simulación.

  En este módulo, el usuario tendrá a su disposición las siguientes disciplinas de planificación: *First-In First-Out* (FIFO), *Priority Queuing* (PQ), *Round Robin* (RR) y *Weighted Fair Queueing* (WFQ). Los gráficos y datos generados en la simulación podrán ser descargados por el usuario.

- **Módulo de gráficos avanzados**: Este módulo contará con una amplia gama de funcionalidades provistas por el paquete Esquisse, que permitirán al usuario crear una gran variedad de gráficos y personalizarlos de acuerdo con sus necesidades.

Kanban será la metodología de desarrollo que se utilizará para el desarrollo de este proyecto por ser simple, adaptable y rápido.

## 1.3. MARCO TEÓRICO

En esta sección se describen los conceptos teóricos importantes sobre QoS, mecanismos de QoS y planificación de paquetes. Además, se analiza brevemente los aspectos importantes relacionados con la metodología ágil de desarrollo de software Kanban, sus principios, beneficios e implementación.

Por otra parte, se muestra una breve descripción sobre las herramientas, tecnologías y paquetes utilizados en el desarrollo del prototipo de aplicación web interactiva.

### 1.3.1. FUNDAMENTOS DE CALIDAD DE SERVICIO (QoS)

#### 1.3.1.1. Introducción

El aumento en la potencia de procesamiento de los dispositivos computacionales y el rápido desarrollo de las redes de telecomunicaciones han permitido la integración de nuevos servicios y aplicaciones para los usuarios de internet, como transmisiones de voz y video en tiempo real. Estos nuevos servicios y aplicaciones imponen nuevos requerimientos a la red, y exigen un monitoreo del comportamiento de la red, generalmente con parámetros como latencia, ancho de banda, tasa de perdida de paquetes, entre otros parámetros más [1].

Es importante tener en cuenta que no todos los servicios y aplicaciones son iguales, por lo que los paquetes que fluyen a través de la red necesitan una forma de ser diferenciados. Además, a medida que el contenido de datos, voz y video va convergiendo en la misma red, estas aplicaciones necesitan compartir recursos de una forma justa, esto debido principalmente a que los recursos de la red no son ilimitados [1], [3].

La tecnología o mecanismo principal que permite el cumplimiento de los puntos mencionados anteriormente se conoce como calidad de servicio (QoS). La QoS se encarga de la administración de los recursos, la diferenciación del flujo de tráfico, y garantiza el envío confiable del contenido a todos los usuarios.

#### 1.3.1.2. Definición de QoS

De acuerdo con la recomendación E.800 de la *International Telecommunication Union - Telecommunication Standardization Sector* (ITU-T), el termino calidad de servicio en el ámbito de las redes de telecomunicaciones se define como: "*Totalidad de características de un servicio de telecomunicaciones que inciden en su capacidad para satisfacer necesidades declaradas e implícitas del usuario del servicio*" [4]. Esta definición aborda la QoS desde la perspectiva del usuario final (persona o máquina).

La QoS también se puede definir como una medida de la capacidad de los dispositivos informáticos y de red para proveer diferentes niveles de servicios a determinadas aplicaciones, esto con el fin de gestionar los recursos para maximizar el rendimiento de la red. El rendimiento de la red puede ser descrito por varios parámetros o características, como el retardo, el ancho de banda, *jitter*, tasa de pérdida de paquetes, entre otros más [5].

A continuación, se proporciona una breve descripción de los principales parámetros de rendimiento de la red [5]:

- **Retardo:** El retardo, también denominada latencia, es la cantidad de tiempo que se tarda en enviar los datos desde una fuente a un destino.
- **Ancho de banda:** El ancho de banda es una medida de la cantidad de datos que una interfaz puede enviar cada segundo. Se mide en bits por segundo.
- ***Jitter*:** *Jitter* es la variación en el retraso de los paquetes.
- **Tasa de pérdida de paquetes:** La tasa de pérdida de paquetes indica el número de paquetes que no llegan al destino en relación con todos los paquetes enviados.

Se puede afirmar, en síntesis, que la QoS se refiere a la aplicación de diferentes niveles de tratamiento de red para diferentes paquetes con el fin de mejorar la experiencia del usuario final.

### 1.3.1.3. Descripción del tráfico

**Tipos de fuentes de tráfico**

La tasa de bits de una aplicación es un factor importante para la red en la asignación de recursos. La dinámica de la tasa de bits a lo largo del tiempo describe el comportamiento de una fuente de tráfico. Según la dinámica de la tasa de bits, todas las aplicaciones se pueden clasificar en dos categorías principales: Tasa de bits constante y Tasa de bits variable [1].

- **Tasa de bits constante (CBR):** Estas aplicaciones envían tráfico a una velocidad constante. Muchas aplicaciones multimedia se incluyen en esta categoría.
- **Tasa de bits variable (VBR):** Las tasas de tráfico de estas aplicaciones no son constantes.

**Parámetros de tráfico**

Aunque es bastante trivial describir el comportamiento del tráfico de una fuente CBR, no es fácil describir completamente el patrón de tráfico de una fuente VBR. No obstante, es posible limitar el tráfico de una fuente VBR utilizando algunos parámetros de tráfico elegidos inteligentemente. Los siguientes tres parámetros se utilizan comúnmente para vincular el tráfico de origen:

- La **tasa pico** (*peak rate*) es la velocidad máxima de datos en cualquier intervalo de tiempo. El tráfico CBR se puede describir completamente utilizando la tasa máxima del tráfico.
- La **tasa promedio** (*average rate*) es la media "a largo plazo" de la tasa de tráfico para una fuente VBR.
- El **tamaño de ráfaga** (*burst size*) se refiere a la cantidad de paquetes que se pueden entregar en la velocidad máxima.

### 1.3.1.4. Modelos para implementar QoS

Internet es incapaz de proporcionar calidad de servicio de extremo a extremo garantizado, por lo que se necesita de un sistema que sea capaz de resolver este problema. A continuación, se presentan los modelos de Servicios Integrados y Servicios Diferenciados.

**Servicios Integrados (IntServ)**

El modelo de Servicios Integrados (RFC 1633), también conocido como IntServ fue desarrollada por la IETF (*Internet Engineering Task Force*) para garantizar la QoS de extremo a extremo en aplicaciones en tiempo real. IntServ se basa en la reserva de recursos de la red según los requisitos de cada aplicación [5].

IntServ utiliza el Protocolo de señalización RSVP (*Resource Reservation Protocol*) para señalar explícitamente las necesidades de QoS del tráfico que atraviesa la red. En IntServ, la aplicación solicita un tipo específico de servicio de acuerdo con su perfil de tráfico. La aplicación enviará los datos a través de la red solo después de que reciba una confirmación de la red [6]. Cabe mencionar que el protocolo RSVP solo se utiliza para la señalización de las necesidades de QoS y no para la transmisión de datos.

IntServ reserva los recursos a lo largo de toda la ruta, además requiere que todos los nodos intermedios mantengan el estado del flujo. Esto ocasiona un alto consumo de recursos en los nodos de la red y, por lo tanto, no es escalable [7].

**Servicios Diferenciados (DiffServ)**

El modelo de Servicios Diferenciados (RFC 2475), más comúnmente conocido como DiffServ, surge como una alternativa para abordar los problemas de escalabilidad del modelo IntServ. A diferencia de IntServ, que utiliza un enfoque de extremo a extremo, DiffServ utiliza un enfoque de salto a salto, además utiliza flujos agregados en lugar de flujos individuales [6].

La escalabilidad se consigue por medio funciones complejas de clasificación y acondicionamiento de tráfico implementadas en los nodos límite de la red. Estos nodos serán los encargados de mapear los paquetes entrantes en alguna de las clases de tráfico soportadas por la red. Para la identificación de los agregados de tráfico se utiliza un código DSCP (*DiffServ Code Point*). Cada paquete que contiene el mismo código DSCP recibe el mismo trato en cada nodo. PHB (*Per Hop Behaviour*) define un comportamiento específico para cada clase de tráfico en cada nodo de la red [6].

Si bien IntServ ofrece la máxima garantía de QoS, consume muchos recursos y, por lo tanto, no se puede escalar fácilmente. Por el contrario, DiffServ consume menos recursos y es más escalable. A veces, los dos se implementan conjuntamente en implementaciones de QoS de red. En la Tabla 1.1 resume las características más importantes de los dos modelos.

**Tabla 1.1.** Modelos para implementar QoS

| Modelo | Descripción |
|---|---|
| Servicios integrados (IntServ) | • IntServ proporciona QoS muy alta a paquetes IP con entrega garantizada.<br>• Define un proceso de señalización para que las aplicaciones le indiquen a la red que requieren QoS especial durante un período y que se debe reservar ancho de banda.<br>• IntServ puede limitar severamente la escalabilidad de una red. |
| Servicios diferenciados (DiffServ) | • DiffServ ofrece alta escalabilidad y flexibilidad en la implementación de QoS.<br>• Los dispositivos de red reconocen las clases de tráfico y proporcionan diferentes niveles de QoS a diferentes clases de tráfico. |

#### 1.3.1.5. Herramientas para la implementación de QoS

Hay tres categorías de herramientas de QoS, las cuales se definen a continuación [8]:

- **Herramientas de clasificación y marcado:** Las herramientas de QoS se encargan del monitoreo de los flujos de tráfico y de la clasificación de los paquetes. Cuando se determina la clase de tráfico, se marcan los paquetes.
- **Herramientas para evitar la congestión:** Las herramientas para evitar la congestión monitorean el tráfico de la red. Cuando el tráfico excede los recursos de red disponibles, parte del tráfico puede descartarse, retrasarse o volverse a marcar de forma selectiva para evitar la congestión.
- **Herramientas de administración de congestión:** Las herramientas de QoS administran la planificación y la configuración del tráfico mientras los paquetes esperan su turno en una cola para salir de la interfaz.

La Figura 1.1 muestra las herramientas de QoS para la implementación de QoS en redes de telecomunicaciones.

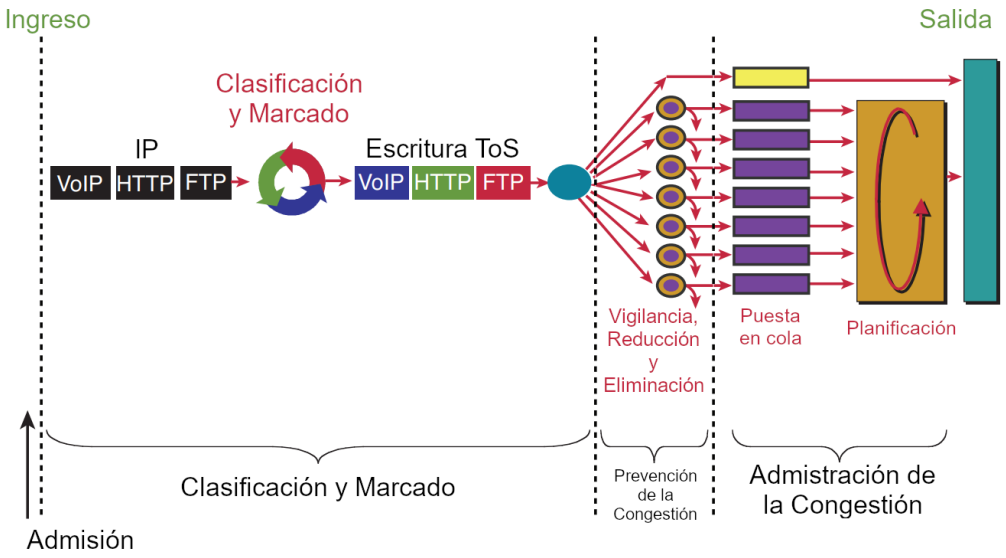


**Figura 1.1.** Herramientas de QoS [8]

### 1.3.1.6. Puesta en cola y planificación

En las redes de conmutación de paquetes, la puesta en cola y la planificación son dos mecanismos básicos que permiten que el ancho de banda de un enlace sea compartido por múltiples fuentes de tráfico [1]. El planificador aplica un comportamiento diferenciado a diferentes clases de servicio, como se ilustra en la Figura 1.2.

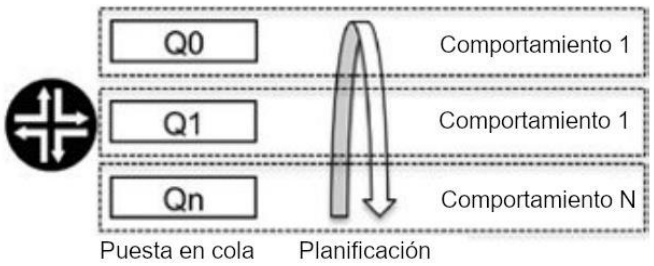


**Figura 1.2.** Puesta en cola y planificación [9]

La puesta en cola se refiere al proceso de almacenar en el búfer los paquetes entrantes a medida que ingresan al enlace. Si hay varios paquetes en espera en el búfer, un algoritmo de planificación define como se asignan los recursos a cada cola [1].

Las redes de QoS de hoy utilizan técnicas avanzadas de planificación y puesta en cola para ayudar a garantizar la QoS requerida por los servicios y aplicaciones.

## 1.3.2. PLANIFICACIÓN PARA LA GESTION DE QoS

La planificación de recursos (ancho de banda del enlace, búferes disponibles, y más) es esencial para garantizar un rendimiento óptimo de las aplicaciones que requieren QoS [1]. De forma general, los *routers* y *switches* identifican los paquetes que requieren un trato

diferenciado, los clasifican, y los envían al enlace de acuerdo con el algoritmo de planificación utilizado.

### 1.3.2.1. Disciplinas de planificación

Hay varias técnicas de planificación diferentes. A continuación, se presenta una discusión detallada sobre algunas de las disciplinas de planificación más destacadas que se utilizan en Internet.

#### *1.3.2.1.1. First-In, First-Out*

*First-In First-Out* (FIFO), también conocido como algoritmo *First Come First Served* (FCFS), es la disciplina o esquema de planificación más simple y rápida disponible en los *routers*. Los paquetes de todos los flujos se colocan en una cola y el *router* procesa los paquetes en el orden de llegada, como se ilustra en la Figura 1.3 [1]. FIFO no discrimina entre tipos o clases de tráficos y, por lo tanto, no toma decisiones sobre la prioridad de los paquetes, es decir, FIFO no brinda un trato diferenciado, todos los paquetes se tratan por igual independientemente de su tamaño, tipo, contenido o cualquier otra característica del paquete [9], [10].

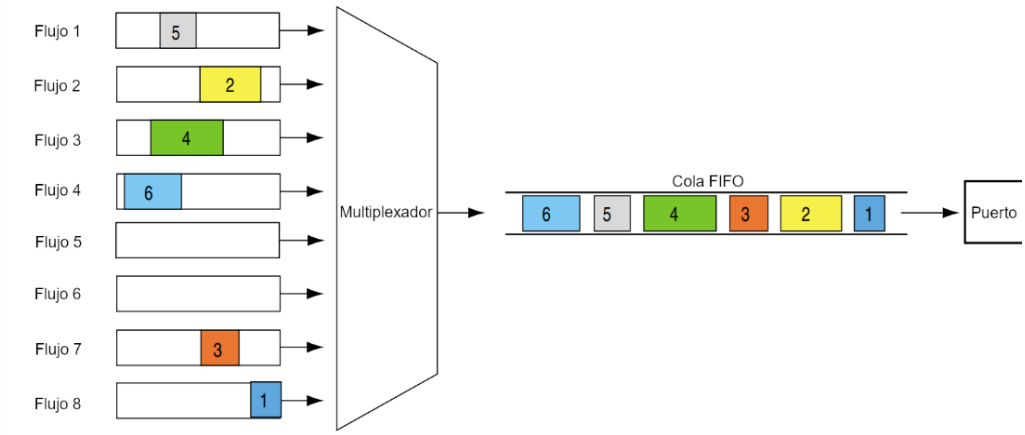


**Figura 1.3.** Operación FIFO básica [11]

FIFO no es adecuado para redes congestionadas porque las fuentes codiciosas pueden ocupar la mayor parte de la cola, lo que podría producir retrasos en otros flujos que utilizan la misma cola. FIFO es eficaz para enlaces con baja latencia y congestión mínima. Si se tiene un enlace con una congestión alta, FIFO puede descartar los paquetes independientemente de su importancia [1].

La Tabla 1.2 resumen los principales beneficios y limitaciones de la disciplina de planificación FIFO [9].

**Tabla 1.2.** Beneficios y limitaciones de FIFO

| Beneficios | Limitaciones |
|---|---|
| • El algoritmo FIFO es sencillo de implementar.<br>• Proporciona un rendimiento aceptable si el enlace no está congestionado.<br>• Consume pocos recursos de procesamiento.<br>• Se puede utilizar en interfaces de alta velocidad. | • FIFO no ofrece un trato diferenciado a los paquetes. En casos de congestión extrema, elimina los paquetes entrantes.<br>• El retardo y el *jitter* no se pueden controlar. Por lo tanto, FIFO no es una solución adecuada para aplicaciones en tiempo real.<br>• Los flujos de una misma fuente pueden consumir todo el espacio del búfer disponible. |

#### *1.3.2.1.2. Priority Queuing*

*Priority Queuing* (PQ) utiliza múltiples colas con prioridades asignadas (de $0$ a $n$) para proporcionar un tratamiento diferenciado a los diferentes flujos de tráfico. Las prioridades asignadas a cada cola establecen el orden (de mayor a menor) para dar servicio a las colas [10]. La Figura 1.4 ilustra la operación básica de la disciplina de planificación PQ.

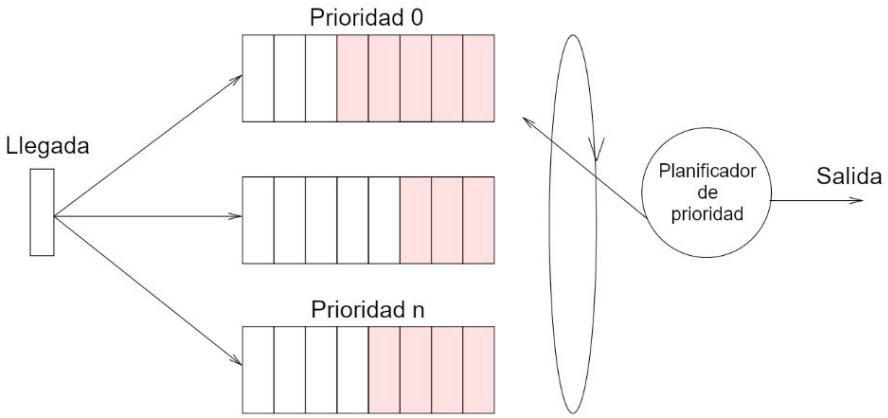


**Figura 1.4.** Operación PQ básica [1]

En PQ, los paquetes de la cola de mayor prioridad se atienden antes que los paquetes de las colas de menor prioridad. Además, las colas de menor prioridad se sirven solo si todas las colas de mayor prioridad están vacías. Debido a este comportamiento, si existe un excesivo flujo de tráfico de alta prioridad, los paquetes pertenecientes a las colas de menor prioridad pueden no obtener recursos (problema de inanición) [10].

PQ es ideal para aplicaciones que son sensibles al retraso y que no manejan la pérdida de paquetes. La Tabla 1.3 resumen los principales beneficios y limitaciones de la disciplina de planificación PQ [9].

**Tabla 1.3.** Beneficios y limitaciones de PQ

| Beneficios | Limitaciones |
|---|---|
| • PQ proporciona un método simple para admitir clases de servicio diferenciadas.<br>• La implementación del algoritmo PQ es simple.<br>• Los requisitos computacionales de almacenamiento en búfer y planificación son bajos y no muy complejos.<br>• Alto rendimiento, menor retraso y mayor ancho de banda para los paquetes en colas de mayor prioridad. | • Si el volumen de tráfico de alta prioridad se vuelve excesivo, las colas de menor prioridad pueden enfrentar una falta total de recursos. |

#### *1.3.2.1.3. Round Robin*

RR utiliza la lógica de operación por turnos, de ahí su nombre, *round-robin*. RR resuelve la limitación de una sola cola de FIFO, ya que mantiene una cola para cada flujo, además resuelve el problema de inanición de PQ, ya que todas las colas son atendidas periódicamente. Cada paquete entrante se coloca en la cola correspondiente y son atendidos por turnos, tomando un paquete de cada cola no vacía (las colas vacías se omiten) [1].

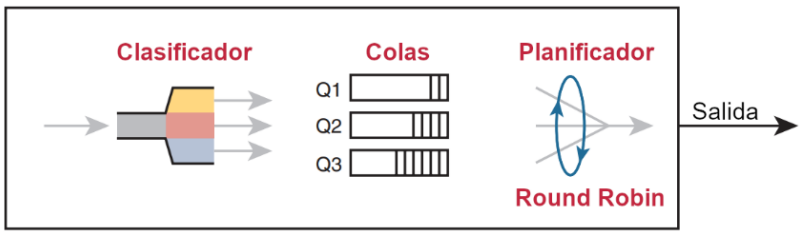


**Figura 1.5.** Operación RR básica [12]

La Figura 1.5 ilustra el funcionamiento de la disciplina de planificación. RR toma algunos paquetes de la cola 1, continua y toma algunos de la cola 2, luego toma algunos de la cola 3, y así sucesivamente, comenzando de nuevo en la cola 1 después de terminar una pasada completa a través de todas las colas [12].

RR intenta asignar el mismo ancho de banda a todas las colas y, por lo tanto, no se logra un trato diferencial. Además, si en una cola hay paquetes de mayor tamaño que las demás colas, en promedio, la cola de paquetes más grandes obtendrá la mayor parte del ancho de banda del enlace [2].

La Tabla 1.4 resumen los principales beneficios y limitaciones de la disciplina de planificación RR [9].

**Tabla 1.4.** Beneficios y limitaciones de RR

| Beneficios | Limitaciones |
|---|---|
| • Todas las colas son atendidas periódicamente.<br>• La implementación del algoritmo PR es simple. | • Pequeños paquetes críticos pueden esperar largos períodos en las colas mientras se atienden los grandes paquetes no críticos.<br>• No ofrece garantías de ancho de banda o retrasos. |

#### *1.3.2.1.4. Weighted Fair Queuing*

*Weighted Fair Queuing* (WFQ) proporciona una asignación justa de ancho de banda a todo el tráfico de la red. WFQ utiliza prioridades o pesos en cada cola para clasificar el tráfico de la red [13]. En WFQ, los paquetes se sirven de acuerdo con un valor de etiqueta, conocido como tiempo de finalización virtual. Cuando el enlace está disponible para enviar un paquete, se selecciona el paquete con el tiempo de finalización virtual más bajo.

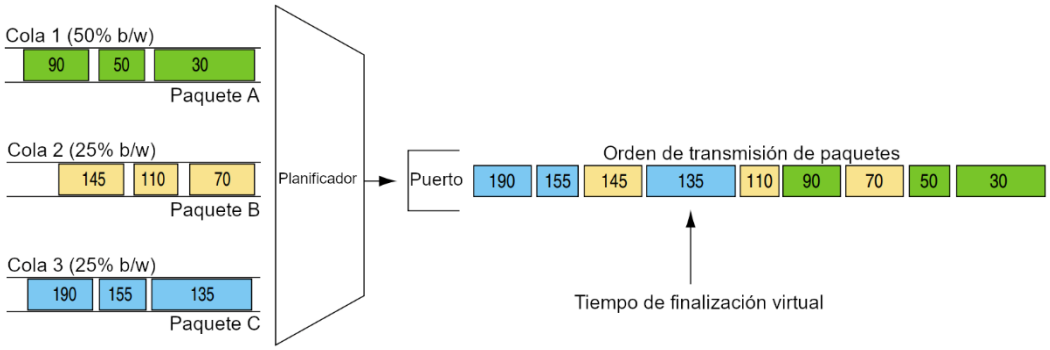


**Figura 1.6.** Operación básica WFQ [11]

**Algoritmo**

La Figura 1.6 ilustra la operación de la disciplina de planificación WFQ. Cada cola tiene un peso $w_k$ $(b/w)$ asociado, donde $k$ representa el índice de la cola. De esta forma, cada cola alcanza una tasa de datos promedio de acuerdo con la Ecuación 1.1, dónde $R$ hace referencia a la tasa de transmisión del enlace. Para la suma de los pesos se debe considerar las colas activas (no vacías).

$$r_k = R \cdot \frac{w_k}{\sum w_k} \tag{1.1}$$

A continuación, para cada paquete se calcula un tiempo de finalización virtual $F_k(i)$ (tiempo en el que un *router* tardaría en enviar un paquete $i$ de la cola $k$) [14]. La Ecuación 1.2 se utiliza para calcular el tiempo de finalización virtual de un paquete.

$$F_k(i) = \max(F_k(i-1), a_k(i)) + \frac{L_k(i)}{r_k} \quad (1.2)$$

Donde $a_k(i)$ y $L_k(i)$ representan el tiempo de llegada y la longitud del paquete $i$ de la cola $k$ respectivamente. Para el primer paquete en la cola, el tiempo de finalización virtual del paquete previo es cero, es decir $F_k(0) = 0$ [14].

Finalmente, cada vez que el enlace esté disponible, se selecciona para su emisión el paquete con el tiempo de finalización virtual más bajo, el cual se determina con la Ecuación 1.3.

$$salida = \min\{F_k(i)\} \quad (1.3)$$

La Tabla 1.5 resumen los principales beneficios y limitaciones de la disciplina de planificación WFQ [9].

**Tabla 1.5.** Beneficios y limitaciones de PQ

| Beneficios | Limitaciones |
|---|---|
| • WFQ proporciona diferenciación de servicios entre clases.<br>• Proporciona una asignación justa de ancho de banda a los flujos. | • WFQ es extremadamente complicado de implementar.<br>• El cálculo de los tiempos de finalización virtual requiere muchos recursos. |

### 1.3.3. METODOLOGÍA ÁGIL KANBAN

Kanban es un método sencillo para la gestión del flujo de trabajo. Está diseñado para ayudarlo a visualizar su trabajo, limitar el trabajo en progreso (WIP) y maximizar la eficiencia [15]. La metodología Kanban se inspira en el sistema de producción de Toyota y en *Lean Manufacturing*[1] (Manufactura esbelta).

La palabra Kanban tiene origen japonés, y se puede traducir al español como tablero o indicador visual. Esta metodología equilibra la demanda de trabajo a realizarse con la capacidad para comenzar un nuevo trabajo. Kanban utiliza un tablero y tarjetas para presentar a todos los participantes una vista de los elementos de trabajo y su progreso [16].

[1] *Lean Manufacturing*: Es una filosofía de gestión derivada principalmente del *Toyota Production System* (TPS). Está enfocada minimizar las pérdidas de los procesos de manufactura y, a su vez, maximizar la creación de valor.

En su forma más simple, un tablero Kanban tiene tres columnas. Cada columna representa una tarea específica. En la primera columna, "Por hacer", se agregan todas las tareas que necesitan resolución. En la segunda columna, "En progreso", se ubican todas las tareas que están en proceso de ser resueltas. Finalmente, está la tercera columna, "Terminado", donde se colocan las tareas finalizadas. En la Figura 1.7 se observa un ejemplo de tablero Kanban.

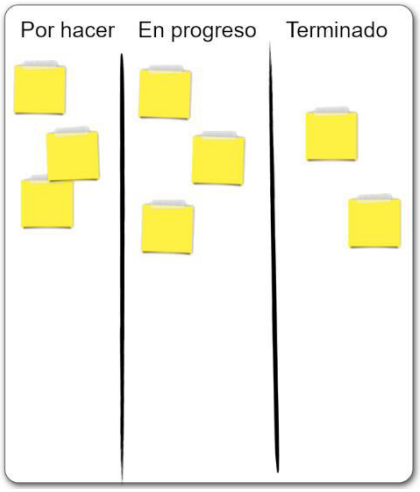


**Figura 1.7.** Tablero Kanban [17]

Cada paso distintivo de un proceso se lo escribe en una tarjeta visual, y se las ubica en un tablero o pizarra. Las tarjetas van moviéndose de acuerdo con el flujo de trabajo hasta que este finaliza. La información presentada en el tablero se debe mantener actualizada en todo momento.

Los principios fundamentales de Kanban son: visualización del flujo de trabajo, limitar el trabajo en proceso, medición y gestión del flujo de trabajo, hacer explícitas las políticas de proceso, y mejora colaborativa [18].

La implementación de Kanban para gestionar el flujo de trabajo trae consigo múltiples beneficios, a continuación, se listan los más relevantes [19]:

- Monitorear y controlar el proceso de producción
- Planificación visual
- Mejorar el flujo
- Alta capacidad de respuesta a los cambios
- Facilitar una alta producción

- Prevenir la sobreproducción
- Mejorar la utilización de la capacidad
- Reducir el tiempo de producción

La implementación de Kanban en el desarrollo de software es muy sencilla. De manera general, solo se necesita seguir las siguientes reglas [20]:

- **Visualizar el flujo de trabajo:** La visualización del flujo de trabajo es importante, por lo cual es necesario dividir el proceso en pasos distintivos y presentarlos como tarjetas en un tablero Kanban. De esta forma, el tablero Kanban muestra el estado del trabajo actual a todo el equipo, fomentando una buena comunicación y colaboración en equipo.
- **Limitar el trabajo en progreso (WIP):** En cada etapa del proceso, el trabajo debe ser lo más eficiente posible. No es beneficioso tener mucho trabajo en proceso, por lo que se debe limitar el número de tarjetas activas en cada columna.
- **Supervisar y mejorar:** Este paso consiste en monitorear y mejorar constantemente el desempeño y rendimiento del desarrollo. Se debe prestar atención a los cuellos de botella y a los elementos apilados, reconsiderar los límites de WIP y realizar los cambios necesarios para aumentar la productividad.

### 1.3.4. HERRAMIENTAS DE DESARROLLO

Para el desarrollo del prototipo de aplicación web se necesita de varias herramientas de desarrollo, las cuales se resumen a continuación.

#### 1.3.4.1. R base

R es un lenguaje de programación orientado a la computación estadística. Es un proyecto GNU similar al lenguaje S, desarrollado por John Chambers y su equipo en *Bell Laboratories*. Se puede pensar en R como una implementación de S, debido a que gran parte del código escrito para S se ejecuta sin cambios en R [21].

R puede ampliarse fácilmente mediante la instalación de paquetes. R cuenta con varios paquetes preinstalados, sin embargo, el usuario puede instalar una amplia gama de paquetes adicionales que están disponibles a través del repositorio de software centralizado CRAN [21].

R es un entorno de software de código abierto. Es gratuito y se puede ajustar y adaptar según los requisitos del usuario y del proyecto. R está diseñado para la manipulación de

datos, cálculo y visualización gráfica, y ofrece una colección amplia, coherente e integrada de herramientas intermedias para el análisis de datos [21]. En definitiva, R es un lenguaje de programación bien desarrollado, simple y efectivo que incluye múltiples características útiles.

#### 1.3.4.2. Paquetes R

Los paquetes R son conjuntos de funciones creadas por la comunidad de desarrollo de R con el único fin de aumentar las funcionalidades de R u optimizar las ya existentes. A continuación, se define de forma breve los dos paquetes de gráficos principales utilizados en el desarrollo del prototipo:

- **Plotly:** Plotly es un paquete R gratuito y de código abierto diseñado para la creación de gráficos interactivos con calidad de publicación. Plotly admite más de 40 tipos de gráficos únicos que cubren una amplia gama de casos de uso [22].
- **Esquisse:** Esquisse es un paquete R que permite la exploración y visualización de los datos de una forma interactiva. Los gráficos son generados mediante el paquete ggplot2. Esquisse posee varias características entre las que destacan la creación de gráficos, filtrado de datos, y exportación de gráficos [23].

#### 1.3.4.3. RStudio Desktop

RStudio Desktop es un entorno de desarrollo integrado (IDE) gratuito y de código abierto para el lenguaje de programación R. RStudio Desktop tiene algunas características interesantes específicamente para la creación, depuración e implementación de aplicaciones web Shiny [24]:

- Editor de resaltado de sintaxis
- Consola
- Entorno de variables
- Utilidades (Historial, Visualizador de gráficos, y más)

RStudio contiene un conjunto de herramientas diseñadas para facilitar el desarrollo de aplicaciones R. RStudio integra funcionalidades de resaltado de sintaxis, autocompletado y sangría inteligente. Además, RStudio permite la administración de varios directorios de trabajo mediante proyectos [25].

RStudio Desktop tiene dos ediciones disponibles para los usuarios. Una edición comercial y otra de código abierto. Ambas ediciones están disponibles para las plataformas Windows,

MacOS y Linux. Cabe destacar que RStudio Desktop también está disponible en su versión web, denominada RStudio Cloud [24].

#### 1.3.4.4. Shiny

*Shiny* es un *framework* R que permite el desarrollo de aplicaciones web utilizando código R. Está diseñado principalmente para científicos de datos, lo que les permite crear aplicaciones *Shiny* complejas sin conocimientos de HTML, CSS o JavaScript. Las funcionalidades y componentes de *Shiny* (servidor e interfaz de usuario) se pueden ampliar y personalizar fácilmente con temas CSS, *widgets* HTML, y acciones JavaScript [26].

Las aplicaciones *Shiny* son fáciles de escribir. No se requieren habilidades de desarrollo web. Está diseñado para un comienzo rápido y, sin embargo, puede evolucionar rápidamente en tareas más complejas. *Shiny* está construido con sólidos principios de ingeniería de software [26].

En su forma más simple, una aplicación *Shiny* requiere de una interfaz de usuario y un servidor. Estos componentes forman la arquitectura básica detrás de todas las aplicaciones Shiny. La función interfaz de usuario (`shinyUI`) controla el diseño y la apariencia de la aplicación, y la función del servidor (`shinyServer`) maneja la lógica detrás de la aplicación [26].

*Shiny* nació como una herramienta que permite a los desarrolladores de R mostrar los resultados de sus investigaciones de forma sencilla e interactiva, sin disponer de un amplio conocimiento en el área del desarrollo web [26].

**Widgets *Shiny***

Las aplicaciones *Shiny* están constituidas casi en su totalidad por elementos HTML de entrada y salida, también conocidos como *widgets*. El paquete *Shiny* proporciona una gran variedad de *widgets* integrados, además, existe una amplia comunidad de desarrollo que se dedica a la creación de paquetes de extensión [26].

**Tabla 1.6.** *Widgets* de entrada y salida

| Tipo | Descripción | Widgets |
|---|---|---|
| Entrada | Los *widgets* de entrada representan un campo de datos que permite a los usuarios ingresar un cualquier tipo de datos. | Los *widgets* de entrada más utilizados son: textInput, numericInput, y selectInput. |
| Salida | Los *widgets* de salida proporcionan un contenedor en el que se insertan los resultados de un cálculo (permite observar los resultados de un cálculo). | Los *widgets* de salida más utilizados son: textOutput, tableOutput, y plotOutput. |

Todos los *widgets* poseen una estructura similar. Los argumentos comunes entre los *widgets* de entrada y salida son los siguientes [26]:

- `inputId`: identificador que se utiliza para conectar el *front-end* con el *back-end.*
- `label`: etiqueta legible para identificar el nombre o función del control.
- `value`: permite establecer un valor predeterminado cuando se inicia la aplicación.

En el ANEXO A se proporciona una descripción más detallada sobre los *widgets* de entrada y salida más utilizados.

**Funciones Shiny**

*Shiny* dispone de múltiples funciones para la creación de aplicaciones web dinámicas. La Tabla 1.7 define los principales tipos de funciones de *Shiny*.

**Tabla 1.7.** Funciones Shiny [27]

| Tipo | Descripción | Funciones |
|---|---|---|
| Diseño de interfaz de usuario | Funciones R para definir la estructura de la interfaz de usuario de una aplicación Shiny. | Las funciones más utilizadas son: `fluidPage`, `fillPage`, `fillRow`, y `column`. |
| Funciones del constructor de interfaces | Funciones R para la construcción de documentos HTML. Implementa funciones para las etiquetes HTML5. | Las funciones más utilizadas son: `tags`, `withTags`, y `HTML`. |
| Funciones de renderizado | *Shiny* dispone de funciones de renderizado, las cuales capturan y procesan las expresiones del lenguaje R. | Las funciones más utilizadas son: `renderPlot`, `renderText`, `renderTable`, y `renderUI`. |
| Funciones reactivas | Funciones de programación reactiva para R. | Las funciones más utilizadas son: `reactive`, `observe`, `observeEvent`, `eventReactive`, `reactiveVal`, y `reactiveValues`. |

Para obtener más información sobre las funciones Shiny, visite el sitio web de la documentación de las funciones *Shiny* [27].

### 1.3.4.5. HTML

HTML es la abreviatura de *Hypertext Markup Language*. HTML es un lenguaje de marcado que se utiliza para estructurar una página web y su contenido. HTML consta de una serie de etiquetas que se utilizan para encerrar diferentes partes del contenido una página web para que se vea o actúe de cierta manera [28].

#### 1.3.4.6. CSS

CSS es la abreviatura de *Cascade Style Sheet*. CSS es un lenguaje basado en reglas destinado a simplificar el proceso de hacer que las páginas web sean presentables, es decir, CSS se encarga principalmente de la apariencia de una página web. Con CSS puede cambiar la fuente, el color, el tamaño y el espaciado del contenido, puede cambiar la disposición de las columnas, añadir animaciones y otros elementos decorativos [29].

CSS hace uso de dos longitudes diferentes: relativa y absoluta. Las unidades de longitud absoluta siempre tienen el mismo tamaño, mientras que, las unidades de longitud relativa son relativas a algo más (p. ej. elemento actual, tamaño de pantalla) [29].

#### 1.3.4.7. JavaScript

JavaScript es un lenguaje de programación que se utiliza principalmente para crear y controlar contenido dinámico en páginas web, es decir, realiza actualizaciones de contenido, mapas interactivos, animación de gráficos 2D/3D, y más [30].

JavaScript junto a CSS y HTML forman parte del conjunto de tecnologías web estándar para el desarrollo web.

#### 1.3.4.8. Shinyapps.io

shinyapps.io es una plataforma de alojamiento que le facilita compartir aplicaciones Shiny en la web en solo unos minutos. El servicio se ejecuta en la nube en servidores compartidos operados por RStudio. Cada aplicación es autónoma y funciona con datos que se cargan con la aplicación o datos que el código extrae de almacenes de datos de terceros, como bases de datos o servicios web [31].

RStudio ofrece una variedad de planes para shinyapps.io, tanto gratuitos como de pago. El plan gratuito permite el despliegue de máximo 5 aplicaciones con un tiempo de actividad limitado de 25 horas mensuales. Mientras que los planes de pago ofrecen mayores beneficios como despliegue de un número ilimitado de aplicaciones, número de horas activas ilimitadas, autenticación, dominios personalizados, entre otros [32].

### 1.3.5. REACTIVIDAD EN *SHINY*

En una aplicación *Shiny*, la lógica del servidor se expresa mediante la programación reactiva. La idea clave de la programación reactiva es especificar un gráfico de dependencias para que cuando cambie una entrada, todas las salidas relacionadas se actualicen automáticamente [26].

En *Shiny*, hay tres tipos de objetos reactivos con sus respectivas clases que se representan con los símbolos de la Figura 1.8 [33].

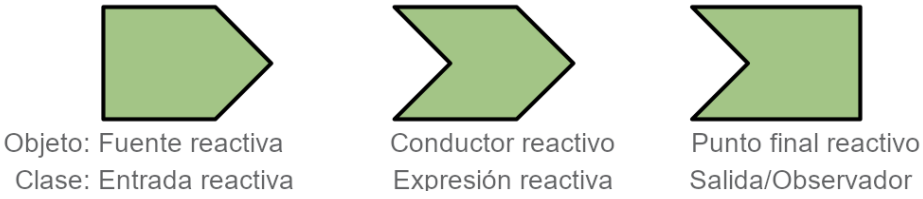


**Figura 1.8.** Objetos y clases reactivas [33]

En el ANEXO B está disponible información más detallada sobre los objetos en programación reactiva y sus respectivas clases.

#### 1.3.5.1. Gráficos reactivos

En aplicaciones de gran tamaño, se vuelve casi imposible tratar de identificar el orden de ejecución observando el código de la aplicación. Para tener una idea del orden de ejecución, se debe observar el gráfico reactivo, el cual describe cómo se conectan las entradas y las salidas [26]. La Figura 1.9 ilustra un ejemplo de un gráfico reactivo.

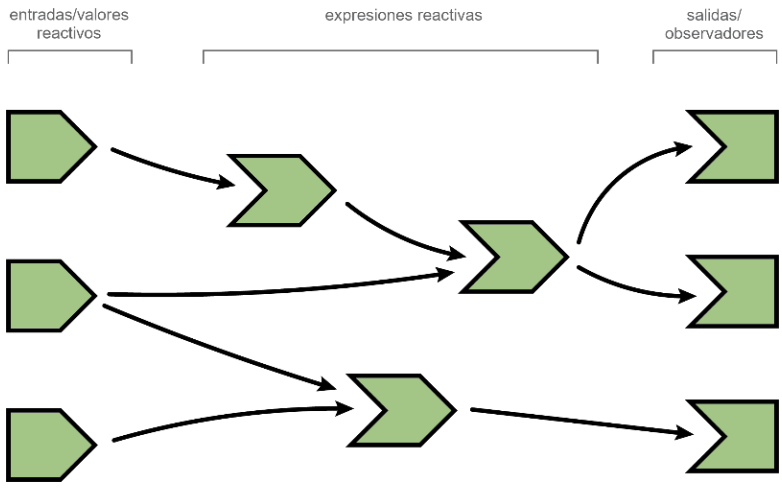


**Figura 1.9.** Gráfico reactivo [26]

Disponer de un gráfico reactivo preliminar facilita en gran medida el proceso de implementación de la lógica del servidor.

**Ejecución de reactividad**

Los objetos reactivos pueden estar en uno de tres estados posibles: listo (valor disponible para retornar), calculando (en proceso de ejecución) e invalidado (ejecución finalizada). Al iniciar la función del servidor de una aplicación *Shiny*, todos los objetos reactivos se encuentran en un estado invalidado, con excepción de las entradas reactivas, que por lo general tienen valores predeterminados asignados.

El proceso de ejecución inicia en las salidas (el orden de selección de una salida es aleatorio). Las salidas necesitan de un valor reactivo para finalizar su ejecución. Si la aplicación es sencilla, las salidas toman los valores de las entradas y terminan su ejecución. En aplicaciones que involucran cálculos más demandantes, se introduce un nodo adicional conocido como expresión reactiva. En este caso, cuando la salida necesita un valor reactivo, la expresión reactiva empieza su proceso de ejecución (salida y expresión reactiva en proceso de ejecución). La expresión reactiva lee una entrada reactiva, realiza los cálculos necesarios, termina su ejecución y retorna un valor reactivo. La salida lee ese valor, actualiza la salida y finaliza su ejecución.

Cabe destacar que las expresiones reactivas solo funcionan cuando es realmente necesario y, si no hay cambios en las entradas, devuelve el valor de su última ejecución guardado en la caché. Una explicación más detallada sobre la ejecución de la reactividad está disponible en el ANEXO C.

**Paquete Reactlog**

El paquete reactlog proporciona una representación visual de la reactividad *Shiny* mediante la construcción de un gráfico de dependencia dirigida del estado reactivo de la aplicación [34]. En la Figura 1.10 se ilustra la interfaz gráfica del paquete reactlog.

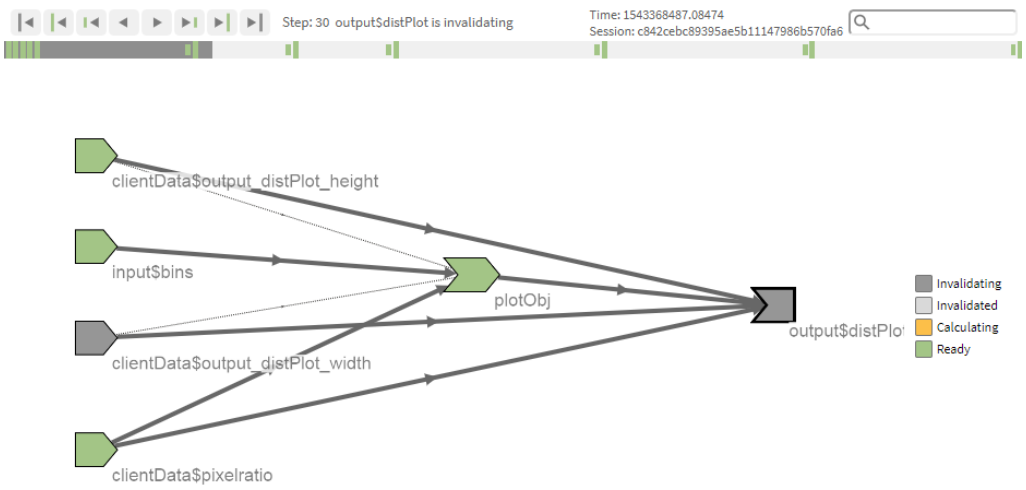


**Figura 1.10.** Interfaz del paquete reactlog [34]

# 2. METODOLOGÍA

Este capítulo está dividido en dos secciones principales, el diseño y la implementación del prototipo de aplicación web interactiva. En la fase de diseño se definen los requerimientos del prototipo y sus módulos. Además, se realiza el bosquejo de los gráficos reactivos y el diseño de las interfaces gráficas. Luego, en la fase de implementación se realiza la preparación del entorno (instalación de herramientas de desarrollo) para la codificación del prototipo y su despliegue en la nube.

Para la implementación del prototipo se realiza un estudio de las aplicaciones Shiny disponibles en la galería de RStudio, estas aplicaciones resaltan características específicas de Shiny, las cuales sirve como punto referencial para el desarrollo del prototipo.

## 2.1. DISEÑO

En esta sección, como primer punto se elabora el tablero Kanban con la lista de actividades a realizarse durante la etapa de desarrollo del prototipo. Posteriormente, para determinar los requerimientos de la aplicación se realiza una entrevista a dos docentes universitarios de la Escuela Politécnica Nacional. Con base a la información obtenida, se establecen los requerimientos funcionales y no funcionales.

En esta sección también se establece la arquitectura del prototipo y se definen los módulos de este. Posterior a la definición de los módulos, se realiza los gráficos reactivos y el diseño de las interfaces de usuario. Para la identificación de los *widgets* de cada módulo se utiliza la información obtenida del estudio de las aplicaciones *Shiny* de la galería de RStudio.

### 2.1.1. TABLERO KANBAN

Para el desarrollo del prototipo de aplicación web se utiliza el tablero Kanban, el cual permite visualizar las diferentes tareas a realizar, además nos permite seguir el flujo de trabajo durante todo el desarrollo del prototipo.

A continuación, en la Tabla 2.1 se presenta el tablero Kanban inicial, el cual está compuesto por tres columnas que corresponden a: “Por hacer”, “En proceso” y “Finalizado”. Adicionalmente, se incluye la columna “ID” para la identificación de cada actividad.

**Tabla 2.1.** Tablero Kanban inicial

| ID | Por hacer | En proceso | Finalizado |
|---|---|---|---|
| 01 | | Entrevistas a dos docentes universitarios. | |
| 02 | | Análisis de aplicaciones Shiny existentes. | |
| 03 | | Obtención de los requerimientos funcionales y no funcionales del prototipo. | |
| 04 | | Definición de los módulos | |
| 05 | Elaboración de los gráficos reactivos del prototipo. | | |
| 06 | Definición de los componentes principales de la interfaz del prototipo. | | |
| 07 | Elaboración de los bosquejos de las interfaces gráficas del prototipo. | | |
| 08 | Instalación y configuración del entorno de desarrollo junto con el framework Shiny, librerías y dependencias necesarias. | | |
| 09 | Estructuración del proyecto. | | |
| 10 | Codificación del módulo configuración de parámetros. | | |
| 11 | Codificación del módulo disciplinas de planificación. | | |
| 12 | Codificación del módulo gráficos avanzados. | | |
| 13 | Despliegue de la aplicación. | | |
| 14 | Validación de los requerimientos funcionales y no funcionales. | | |
| 15 | Validación de los resultados | | |
| 16 | Pruebas automatizadas. | | |
| 17 | Pruebas de compatibilidad. | | |
| 18 | Corrección de errores. | | |

### 2.1.2. RECOLECCIÓN Y ANÁLISIS DE REQUERIMIENTOS

Los requerimientos funcionales y no funcionales se seleccionan en base a los resultados obtenidos de las entrevistas realizadas a los docentes universitarios.

### 2.1.2.1. Estudio de aplicaciones *Shiny*

En la galería de *Shiny* están disponibles una gran cantidad de aplicaciones para inspirarse y aprender de ellas. Las aplicaciones están divididas en dos categorías principales: *Shiny User Showcase* y *Shiny Demos*. *Shiny User Showcase* se compone de contribuciones de la comunidad de desarrolladores de aplicaciones *Shiny*. Por otro lado, *Shiny Demos* son una serie de aplicaciones creadas por los desarrolladores del *framework Shiny* [35]. La Figura 2.1 ilustra las categorías de las aplicaciones *Shiny* disponibles.

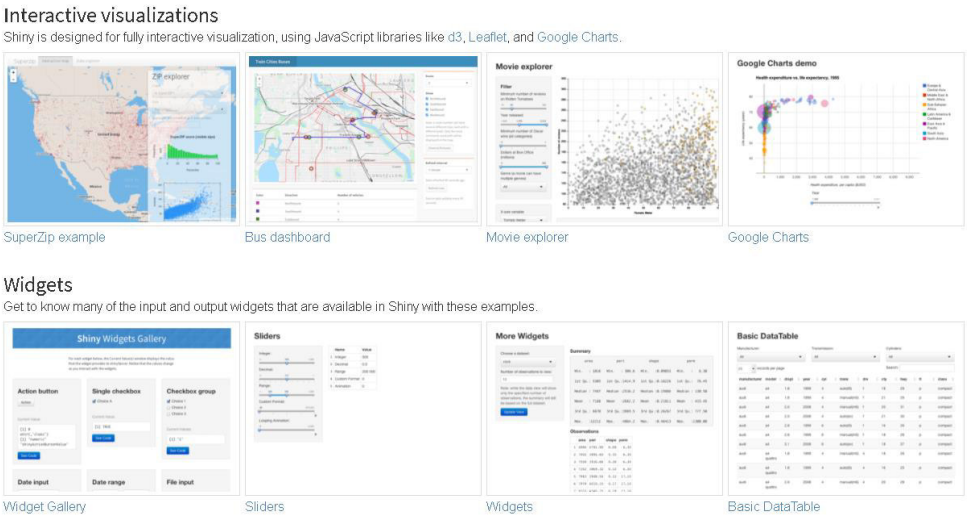


**Figura 2.1.** Galería de aplicaciones *Shiny*

Las aplicaciones disponibles en la galería están diseñadas para resaltar características específicas del paquete *Shiny*. A continuación, se discute brevemente las categorías disponibles y sus principales características [35]:

- **Inicio simple:** Aplicaciones desarrolladas específicamente para desarrolladores nuevos de *Shiny*. Estas aplicaciones implementan *widgets* básicos para el desarrollo de una aplicación funcional.
- **Visualizaciones interactivas:** En esta sección existe una gran variedad de aplicaciones web que implementan visualizaciones interactivas utilizando bibliotecas de JavaScript.
- ***Widgets*:** Las aplicaciones disponibles en esta sección implementan la mayoría de los *widgets* de entrada y salida disponibles en *Shiny*.
- **Diseño de la aplicación:** Cada aplicación de esta categoría demuestra una o más de las funciones que puede utilizar para organizar la interfaz de usuario de la aplicación.

- **Interfaz de usuario dinámica:** Estos ejemplos muestran cómo crear una interfaz de usuario que cambie dinámicamente en base a una entrada proporcionada por el usuario.
- **Programación reactiva:** Estos ejemplos ilustran algunas características y modismos útiles del *framework* de programación reactiva de *Shiny*.
- ***Shiny* avanzado:** Estos ejemplos muestran cómo ampliar *Shiny* y utilizar funciones avanzadas para optimizar el rendimiento de las funciones implementadas.
- **Gráficas interactivas:** Estos ejemplos muestran cómo utilizar las funciones de trazado interactivo de *Shiny*.
- **Funciones profesionales:** Estos ejemplos demuestran algunas de las características únicas de RStudio Connect.

#### 2.1.2.2. Entrevistas

En el presente Trabajo de Titulación se realizaron entrevistas mediante encuestas a dos docentes de Facultad de Ingeniería Eléctrica y Electrónica de la Escuela Politécnica Nacional. Las entrevistas se las realizó con la finalidad de establecer los requerimientos funcionales y no funcionales para el desarrollo del prototipo de aplicación web.

El modelo de la entrevista y los resultados obtenidos se muestran en el ANEXO D.

#### 2.1.2.3. Definición de requerimientos

Los requerimientos funcionales y no funcionales serán descritos mediante el siguiente formato:

- **ID:** permite identificar el requerimiento, el mismo que tiene el siguiente formato: RFXX para los requerimientos funcionales y RNFXX para los requerimientos no funcionales; dónde XX representa el número de requerimiento.
- **Título:** nombre del requerimiento.
- **Descripción:** detalla las funciones de los requerimientos.

##### *2.1.2.3.1. Requerimientos funcionales*

Los requerimientos funcionales (lo que hace el sistema) definen los servicios que debe proporcionar el sistema y como este actúa frente a las entradas [36]. En la Tabla 2.2 se muestra un resumen de los requerimientos funcionales obtenidos.

**Tabla 2.2.** Requerimientos funcionales

| ID | Título | Descripción |
|---|---|---|
| RF01 | Configuración de parámetros de simulación | El prototipo de aplicación web debe permitir la configuración de los siguientes parámetros simulación: Número de flujos / colas, número de paquetes, tipo de distribución de probabilidad (normal, uniforme y exponencial) y velocidad de transmisión del enlace. |
| RF02 | Actualización de los parámetros de simulación | El prototipo de aplicación web debe permitir la visualización y modificación de los parámetros previamente configurados. |
| RF03 | Visualización de resultados | El prototipo de aplicación web debe permitir la visualización de los resultados de la simulación mediante gráficos interactivos (gráficos de barra, gráficos de dispersión, gráficos circulares e histogramas) y tablas. |
| RF04 | Descarga de datos de simulación | El prototipo de aplicación web debe permitir descargar los datos de la simulación en formato de archivos de valores separados por coma (.csv) y texto simple (.txt). |
| RF05 | Reportes de simulación | El prototipo de aplicación web debe generar un informe con un resumen de los parámetros configurados, resultados generados por disciplina y en conjunto. |
| RF06 | Gráficos avanzados | El prototipo de aplicación web debe permitir la creación de gráficos a partir de archivos con extensión .csv o .txt. Se debe permitir la configuración de los siguientes parámetros de los gráficos: titulo, subtitulo, títulos y límites de los ejes, y color. |
| RF07 | Descarga de gráficos de simulación | El prototipo de aplicación web debe permitir la descarga de los gráficos personalizados generados en formato PNG, JPG/JPEG y PDF. |

#### *2.1.2.3.2. Requerimientos no funcionales*

Los requisitos no funcionales definen el comportamiento del sistema frente a atributos o características observables como la presentación, rendimiento, mantenibilidad, escalabilidad, usabilidad, entre otros [36]. En la Tabla 2.3 se muestra un resumen de los requerimientos no funcionales obtenidos.

**Tabla 2.3.** Requerimientos no funcionales

| ID | Título | Descripción |
|---|---|---|
| RNF01 | Presentación y Usabilidad | El prototipo de aplicación web debe tener una interfaz gráfica sencilla e intuitiva de manejar. El prototipo de aplicación web debe proporcionar notificaciones de información, advertencia y error orientados al usuario. |
| RNF02 | Rendimiento | El tiempo para iniciar o reiniciar el prototipo de aplicación web no podrá ser mayor a 15 segundos. |

| | | |
|---|---|---|
| RNF03 | Mantenibilidad | El prototipo de aplicación web debe permitir la adición, modificación y eliminación de funcionalidades. Además, el código debe estar correctamente documentado. |
| RNF04 | Integridad y validez de los datos | Todos los campos de entrada del prototipo de aplicación web deben tener las validaciones de entrada respectivas. |

### 2.1.3. ARQUITECTURA DEL PROTOTIPO

Para el desarrollo e implementación del presente proyecto, que contempla un prototipo de aplicación web interactiva, se ha hecho uso de la arquitectura cliente/servidor. En este tipo de arquitectura el servidor aloja, entrega y gestiona los servicios y recursos que consumirá el cliente a través de un navegador web.

A continuación, en la Figura 2.2 se ilustra de manera gráfica la interacción entre varios clientes y el servidor.

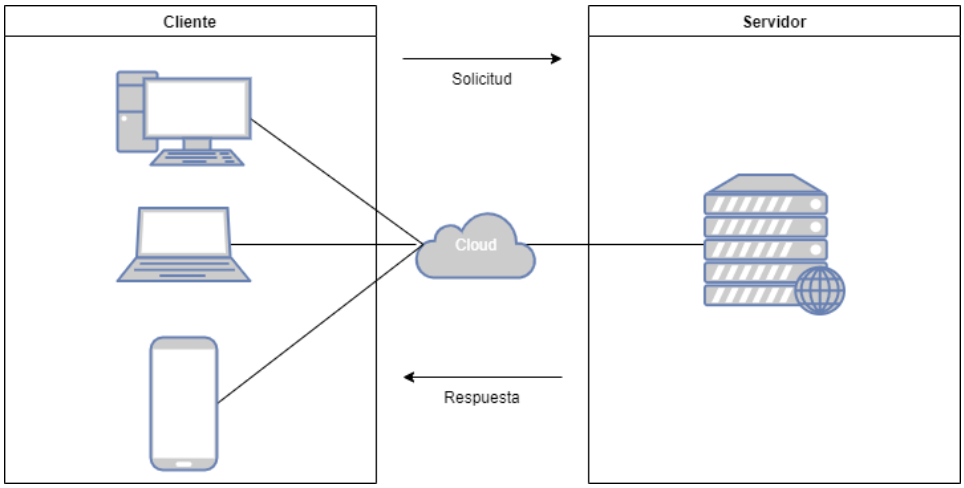


**Figura 2.2.** Arquitectura del prototipo

### 2.1.4. MÓDULOS

#### 2.1.4.1. Módulo configuración de parámetros

El módulo configuración de parámetros estará compuesto por una interfaz gráfica de usuario con varios campos numéricos y de selección. Por medio de estos campos, el usuario podrá configurar los diferentes parámetros (p. ej. número de colas, cantidad de paquetes por cola, tipo de distribución de probabilidad, entre otros) necesarios para la simulación de cada disciplina. Además, cuenta con una sección en donde se muestra un resumen en tiempo real de la información ingresada por el usuario, de esta forma se podrá observar en todo momento los valores ingresados, para así modificarlos de ser necesario.

En la Tabla 2.4 se muestran los requerimientos funcionales para el módulo configuración de parámetros y el número de horas estimadas para su desarrollo.

**Tabla 2.4.** Requerimientos funcionales (configuración de parámetros)

| ID | Horas estimadas |
|---|---|
| RF01 | 20 |
| RF02 | 20 |
| **TOTAL** | **40** |

### 2.1.4.2. Módulo disciplinas de planificación

El módulo de disciplinas de planificación de paquetes consta de un submódulo por cada disciplina de planificación implementada. Cada submódulo implementa una interfaz gráfica de usuario con varios campos de acuerdo con la disciplina seleccionada. En cada interfaz gráfica están disponibles varios componentes interactivos para la exploración y visualización de los datos de simulación.

El usuario podrá elegir entre las siguientes disciplinas de planificación: *First-In First-Out (FIFO)*, *Round Robin (RR)*, *Priority Queuing (PQ)*, y *Weighted Fair Queue (WFQ)*. El usuario podrá generar un informe en dónde se detallarán los parámetros de la disciplina simulada y los resultados obtenidos. Además, los datos y gráficos obtenidos podrán ser descargados.

En la Tabla 2.5 se muestran los requerimientos funcionales para el módulo disciplinas de planificación y el número de horas estimadas para su desarrollo.

**Tabla 2.5.** Requerimientos funcionales (disciplinas de planificación)

| ID | Horas estimadas |
|---|---|
| RF03 | 30 |
| RF04 | 30 |
| RF05 | 10 |
| **TOTAL** | **70** |

### 2.1.4.3. Módulo Gráficos avanzado

El módulo de gráficos avanzados consta de una amplia gama de funcionalidades que permitirán al usuario crear una gran variedad de gráficos y personalizarlos de acuerdo con sus necesidades. Entre las funcionalidades más destacadas se encuentran las siguientes:

- Importar archivos de datos.

- Seleccionar el tipo de gráfico a realizar.
- Modificar el título y subtítulo del gráfico.
- Modificar los títulos y límites de los ejes.
- Seleccionar los colores del gráfico.

En la Tabla 2.6 se muestran los requerimientos funcionales para el módulo gráficos avanzados y el número de horas estimadas para su desarrollo.

**Tabla 2.6.** Requerimientos funcionales (gráficos avanzados)

| ID | Horas estimadas |
|---|---|
| RF06 | 5 |
| RF07 | 5 |
| **TOTAL** | **10** |

### 2.1.5. ACTUALIZACIÓN DEL TABLERO KANBAN

Las actividades del 1 al 4, de la Tabla 2.1, relacionadas con las entrevistas, obtención y análisis de requerimientos, y definición de los módulos del prototipo han finalizado, por lo que pasan de la columna “En Proceso” a la columna “Finalizado”. El siguiente paso consiste en la elaboración de los gráficos reactivos y diseño de las interfaces, por lo que las actividades 5, 6, y 7, de la Tabla 2.1, pasan de la columna “Por Hacer” a la columna “En Proceso” del tablero Kanban. Las demás actividades del tablero Kanban permanecen en la columna “Por Hacer”.

### 2.1.6. GRÁFICOS REACTIVOS

Un gráfico reactivo proporciona a los desarrolladores *Shiny* una vista rápida del funcionamiento de una aplicación reactiva. La Figura 2.3 describe la lógica del módulo de configuración de parámetros, que permite recuperar los valores de los *widgets* de entrada, transformarlos a un formato adecuado para su uso, y mostrarlos en una tabla.

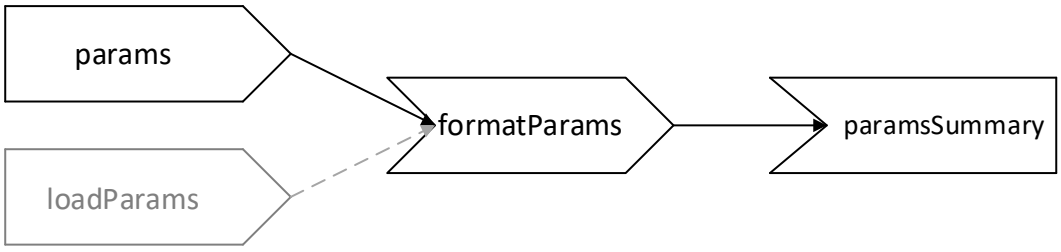


**Figura 2.3.** Gráfico reactivo del módulo de configuración de parámetros

La Figura 2.4 describe la lógica del módulo de disciplinas de planificación, que permite tomar los datos de entrada en un formato adecuado, procesarlos y mostrar su salida por medio de gráficos y tablas. Además, también la lógica para la generación de un reporte dinámico. Es necesario señalar que la lógica para los módulos de las disciplinas de planificación FIFO, PQ, RR y WFQ es la misma, por lo que no es necesario realizar un diagrama individual.

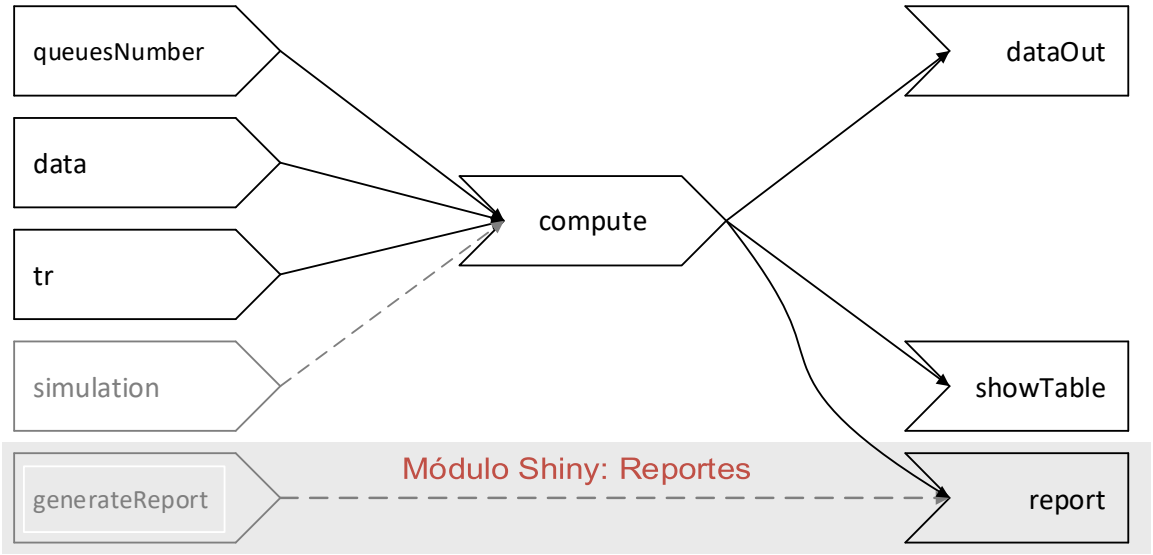


**Figura 2.4.** Gráfico reactivo del módulo disciplinas de planificación

La Figura 2.5 describe la lógica del módulo gráficos avanzados, que permite generar diferentes tipos de gráficos mediante un archivo de datos de simulación. El paquete Esquisse dispone de varios módulos para el análisis y visualización de los gráficos, por lo que a nivel de programación solo es necesario indicarle la ruta del conjunto de datos predeterminado y configurar los parámetros que se visualizarán en el menú de Esquisse.

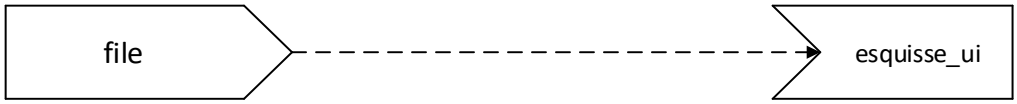


**Figura 2.5.** Gráfico reactivo del módulo gráficos avanzados

### 2.1.7. DISEÑO DE LAS INTERFACES DEL PROTOTIPO

La Tabla 2.7 muestra una lista con los *widgets* de entrada y de salida más adecuados para implementar las funcionalidades del prototipo.

**Tabla 2.7**. Principales componentes del prototipo.

| **Módulo** | ***Widgets*** |
|---|---|
| Configuración de parámetros | - Entradas numéricas |

| | |
|---|---|
| | - Selectores de entrada<br>- Controles deslizantes<br>- Botones de acción<br>- Tablas |
| Disciplinas de planificación | - Entradas numéricas<br>- Botones de acción<br>- Gráficos<br>- Tablas |
| Gráficos avanzados | - esquisse_ui |

El diseño de la composición visual de las interfaces gráficas es una parte fundamental para el desarrollo de aplicaciones web. Los bosquejos de las interfaces gráficas se han diseñado en la herramienta online Framer [37], tomando en consideración los *widgets* de la Tabla 2.7. Los bosquejos realizados contienen texto, selectores, tablas y gráficos, y sirvieron para distribuir el espacio y obtener un primer plano del prototipo a desarrollar.

#### 2.1.7.1. Módulo configuración de parámetros

La Figura 2.6 muestra el diseño preliminar de la interfaz del módulo de configuración de parámetros. En esta interfaz el usuario tendrá a su disposición una variedad de *widgets* de entrada y salida, que le permitirán ingresar y modificar los parámetros de simulación.

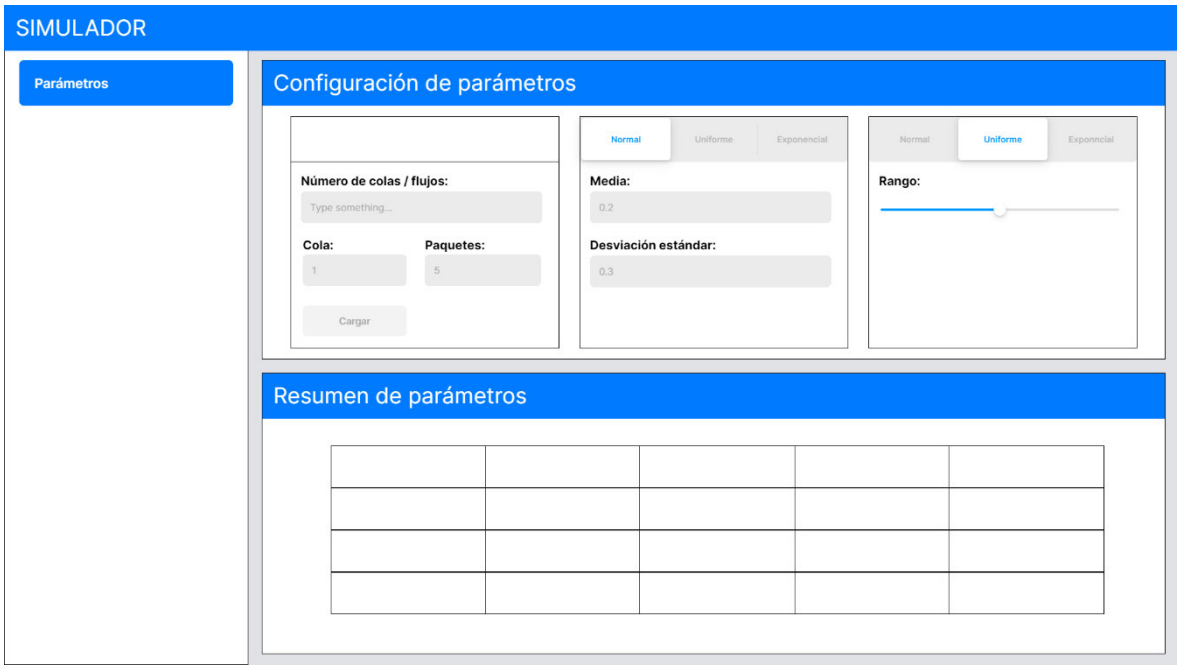


**Figura 2.6.** Interfaz configuración de parámetros

### 2.1.7.2. Módulo configuración de disciplinas

Una vez el usuario ha ingresado correctamente los parámetros de simulación, puede cambiar a la interfaz de la disciplina de planificación deseada. Cabe mencionar que todas las disciplinas de planificación a implementar tienen características similares y, por ende, las interfaces gráficas correspondientes a cada una de estas serán similares.

La Figura 2.7 muestra la vista cuando el usuario selecciona una disciplina de planificación. En esta interfaz están disponible todos los resultados de la simulación de la disciplina seleccionada, la cual consta de varios gráficos interactivos y una tabla de datos.

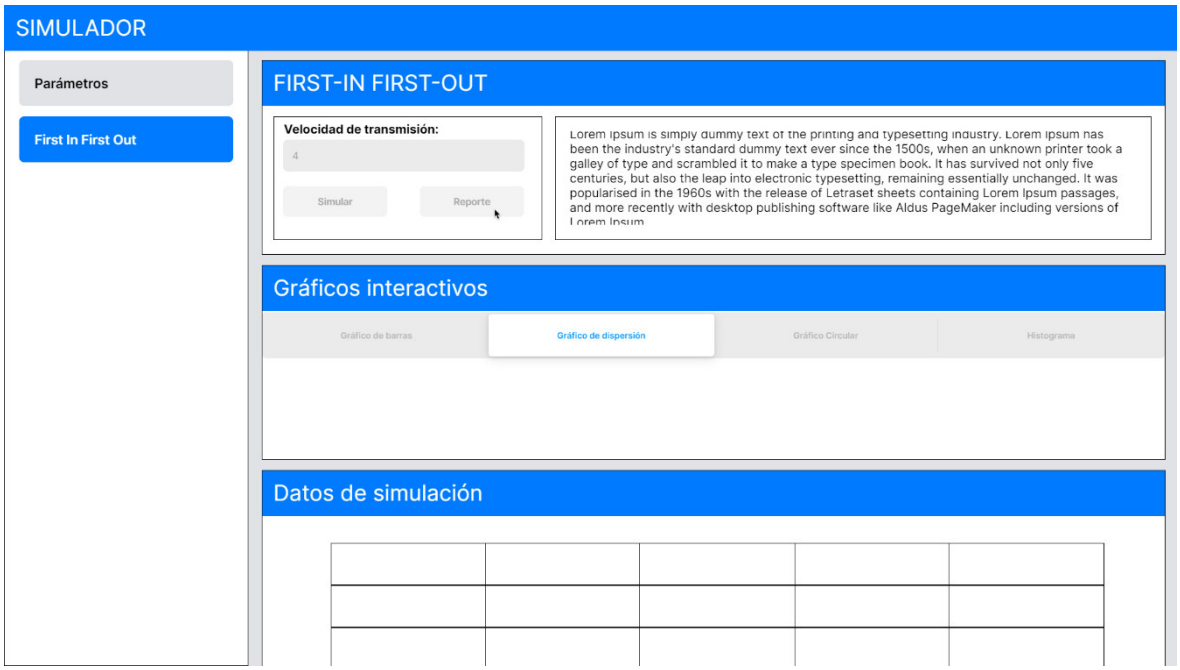


**Figura 2.7.** Interfaz disciplina de planificación

### 2.1.7.3. Módulo de gráficos avanzados

El usuario será capaz de crear gráficos personalizados a través de los archivos generados en la simulación de las disciplinas de planificación. El paquete Esquisse proporciona todos los componentes de la interfaz gráfica. En la Figura 2.8 se puede observar la interfaz del módulo de gráficos avanzados.


**Figura 2.8.** Módulo de gráficos avanzados

## 2.2. IMPLEMENTACIÓN

En esta sección se muestran los detalles correspondientes a la implementación del prototipo de aplicación web de acuerdo con el diseño especificado en la sección 2.1.7. Como primer paso, se actualiza las tareas del tablero Kanban, después se resume brevemente los pasos de la instalación de las herramientas necesarias para el desarrollo del prototipo. Una vez que se han instalado todas las herramientas necesarias, se procede con la codificación del prototipo. Finalmente, se describe el proceso de despliegue del prototipo en la plataforma shinyapps.io.

### 2.2.1. ACTUALIZACIÓN DEL TABLERO KANBAN

Las actividades que se encontraban en la columna “En proceso”, en la Tabla 2.1, han finalizado, por lo que se las mueve a la columna “Finalizado”. Las actividades del 8 al 13, relacionadas con la instalación de las herramientas de desarrollo del prototipo y su implementación inician su proceso de desarrollo, por lo que se mueven de la columna “Por hacer” a la columna “En proceso”. Las demás actividades del tablero Kanban permanecen sin cambios.

### 2.2.2. INSTALACIÓN DE LAS HERRAMIENTAS

El desarrollo competo del prototipo de aplicación web se lo llevó a cabo en un equipo con Windows 10, por lo que la información sobre la instalación y configuración de las diferentes herramientas están basada en este sistema operativo.

#### 2.2.2.1. R base

La instalación de R en un ordenador con el sistema operativo Windows 10 es muy sencillo. El primer paso consiste en descargar el archivo binario de la página oficial de R [38]. Una vez descargado el archivo, lo ejecutamos y seguimos las instrucciones del asistente de instalación de R. Una vez que el proceso de instalación de R ha finalizado, se procede a verificar que el entorno de R funcione correctamente, para lo cual se ejecuta el entorno R. La Figura 2.9 muestra el inicio correcto de la consola de R, lo que demuestra que su instalación fue exitosa.

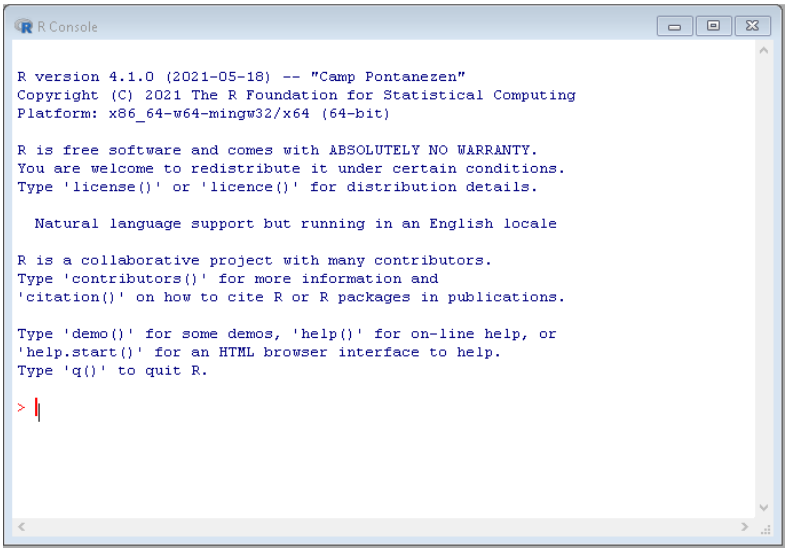


**Figura 2.9.** Versión de R base

#### 2.2.2.2. RStudio Desktop

Una vez que R está instalado en el sistema, puede continuar e instalar el IDE RStudio [39]. Para eso, nos dirigimos a la página de descargas de RStudio y descargamos el archivo binario para Windows. Una vez que se ha descargado el instalador de RStudio, todo lo que queda es navegar a la carpeta de descargas, ejecutar el instalador y seguir las instrucciones de instalación. Durante la instalación, RStudio comprueba la existencia de R en el sistema, si no lo encuentra, la instalación se cancela. Por esta razón es indispensable que la instalación de R sea exitosa.

### 2.2.2.3. Paquetes R

Los principales paquetes de gráficos que se utilizaron en el desarrollo del prototipo son Plotly y Esquisse. Para la instalación de los paquetes es necesario ingresar a RStudio, y en la sección de consola (*Console*) introducir el comando pertinente para cada paquete. El comando para la instalación de cualquier paquete R tiene la siguiente estructura: `install.packages("package")`; dónde `package` representa el nombre del paquete. De esta forma, para la instalación de los paquetes Plotly y Esquisse se utiliza el siguiente comando: `install.packages(c("plotly", "esquisse"))`.

El proceso de instalación de los paquetes de gráficos antes mencionados se observa en la Figura 2.10.

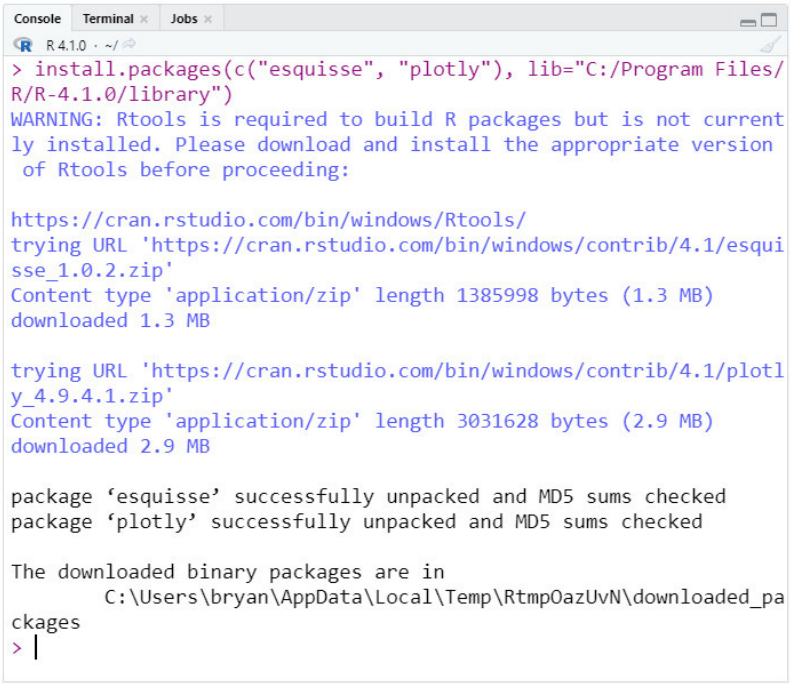


**Figura 2.10.** Instalación de los paquetes Plotly y Esquisse

### 2.2.2.4. shinyapps.io

Shinyapps.io es una plataforma como servicio (PaaS) para alojar aplicaciones web *Shiny* [40]. Para el despliegue de la aplicación es necesario instalar el paquete rsconnect y disponer de una cuenta en shinyapps.io.

#### Instalación de rsconnect

El paquete rsconnect permite el despliegue de aplicaciones *Shiny* en la plataforma de alojamiento shinyapps.io. El paquete rsconnect está disponible en el repositorio CRAN, por lo que se puede instalar ejecutando el comando `install.packages ('rsconnect')`. Una

vez el paquete rsconnect esté instalado, se lo carga en la sesión R actual mediante el comando `library(rsconnect)` [41].

### Cuenta de shinyapps.io

Para la creación de una cuenta en shinyapps.io, es necesario dirigirse a su página oficial y hacer clic en *Sign Up* (Registrarse). El sitio le pedirá que se registre con un correo electrónico y una contraseña, una cuenta de Google o una cuenta de GitHub. Una vez que se ha creado la cuenta e iniciado sesión satisfactoriamente, se observa un *dashboard* similar al de la Figura 2.11.

Con la creación de la cuenta, shinyapps.io genera automáticamente un token y una llave secreta, los cuales son utilizados por el paquete rsconnect para vincular RStudio con la cuenta de shyniapps.io.

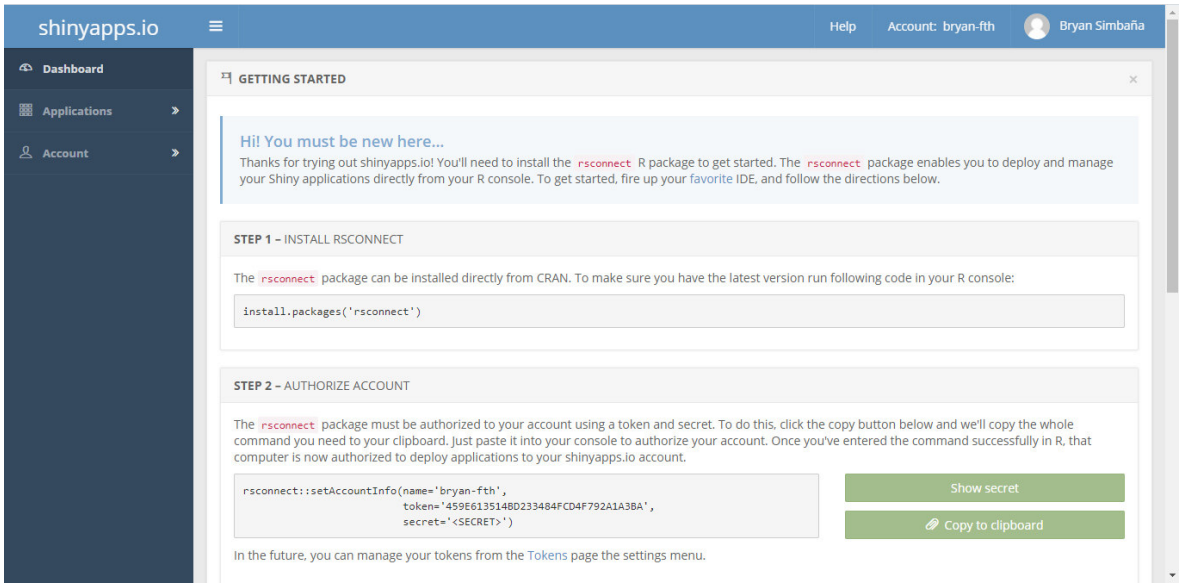


**Figura 2.11.** Dashboard de shinyapps.io

Además de los paquetes ya mencionados, en el desarrollo del prototipo se hizo uso de múltiples paquetes adicionales, los cuales se listan en la Tabla 2.8.

**Tabla 2.8.** Paquetes adicionales

| Paquete | Descripción | Instalación |
|---|---|---|
| bs4Dash | Plantillas HTML (Bootstrap 4) de código abierto para R. | `install.packages("bs4Dash")` |
| shinyvalidate | shinyvalidate agrega capacidades de validación de entrada a los *widgets* Shiny. | `install.packages("shinyvalidate")` |

| | | |
|---|---|---|
| shinyWidgets | Colección de controles de entrada personalizados y componentes de interfaz de usuario. | `install.packages("shinyWidgets")` |
| vroom | El paquete vroom lee y escribe datos (como 'csv', 'tsv' y 'fwf') rápidamente. | `install.packages("vroom")` |
| dplyr | dplyr es un paquete de R para la manipulación de datos. | `install.packages("dplyr")` |
| knitr | El paquete knitr es un motor para la generación de informes dinámicos. | `install.packages("knitr")` |
| kableExtra | El paquete kableExtra está diseñado para extender la funcionalidad básica de las tablas en LaTeX. | `install.packages("kableExtra")` |
| shinyjs | Realiza operaciones comunes de JavaScript. | `install.packages("shinyjs")` |
| shinyloadtest | El paquete shinyloadtest permite realizar pruebas automatizadas de carga. El paquete simula la interacción de n usuarios simultáneos con la aplicación para recopilar métricas de latencia [42]. | `install.packages("shinyloadtest")` |

### 2.2.3. ESTRUCTURA DEL PROYECTO

El prototipo se desarrolló en su totalidad en el IDE de RStudio Desktop, utilizando el lenguaje de programación R, mediante la creación de un proyecto de aplicación web *Shiny* de nombre scheduler-bs4Dash.

Hay dos formas convenientes de crear una aplicación web *Shiny* en RStudio:

- La primera opción consiste en crear un proyecto que contenga la aplicación básica, haciendo clic en *File* | *New Project*, luego en *New Directory* y *Shiny Web Application*. En la ventana emergente se debe introducir el nombre del proyecto y seleccionar una de las dos opciones disponibles: un solo archivo (app.R) o múltiples archivos (ui.R y server.R).
- Por otro lado, la segunda opción consiste en crear de forma manual un directorio para la aplicación y crear los archivos R necesarios (app.R o ui.R y server.R) para desarrollo de la aplicación.

Para una mayor facilidad en la creación y desarrollo del prototipo se decidió crear un proyecto *Shiny* con múltiples archivos (ui.R/server.R). Separar la aplicación en ui.R y server.R hace que sea mucho más fácil depurar y mantener los componentes del prototipo. La Figura 2.12 muestra la creación del proyecto.

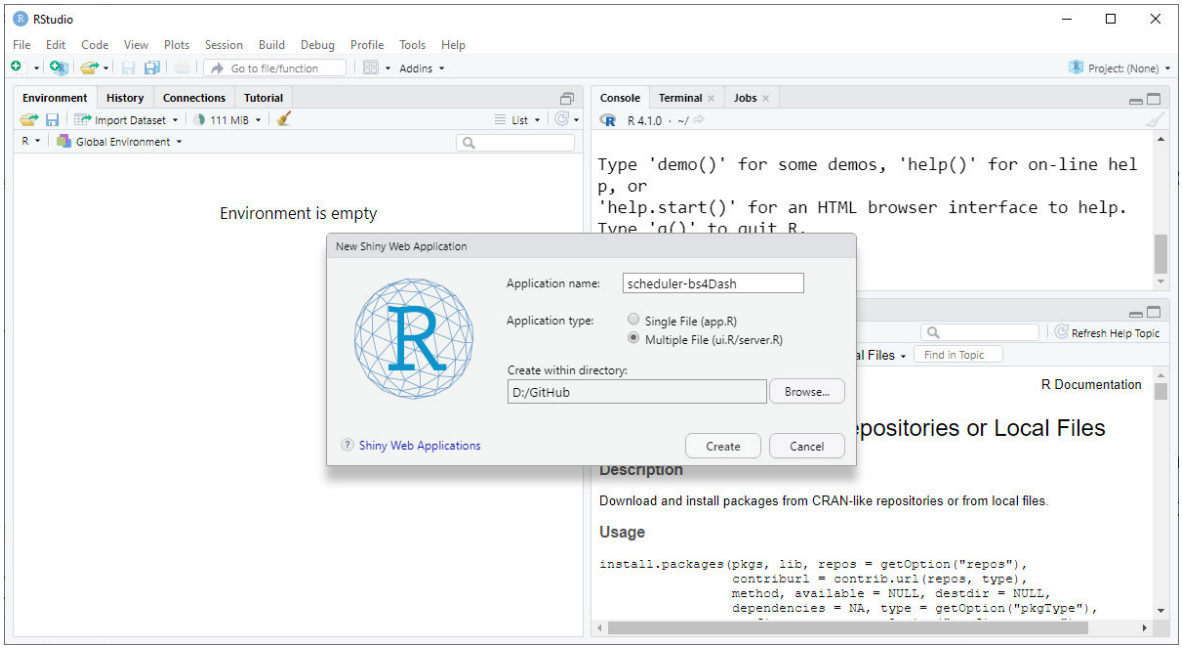


**Figura 2.12.** Creación del proyecto scheduler-bs4Dash

RStudio Desktop, de forma automática crea el directorio del proyecto con los dos scripts R, ui.R, que controla el diseño y la apariencia del prototipo, y server.R, que contiene la lógica del prototipo. Si bien se puede tener un solo archivo app.R, es mucho más fácil de administrar cuando se distribuye en varios archivos. La Figura 2.13 ilustra la estructura del prototipo.

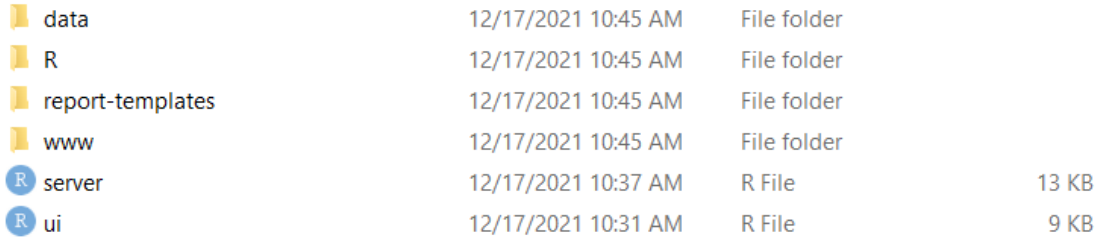


**Figura 2.13.** Estructura del proyecto

Esta es una estructura de carpetas muy básica y se adapta a las necesidades del proyecto. En el directorio principal del prototipo hay disponibles cuatro subdirectorios: *data*, *R*, *report-templates* y *www*. En la Tabla 2.9 se muestra una breve descripción del propósito de cada subdirectorio.

**Tabla 2.9.** Subdirectorios del proyecto

| Directorio | Descripción |
|---|---|
| `data` | La carpeta *data* es el directorio para los archivos predeterminados. |

| | |
|---|---|
| `R` | La carpeta R contiene varios scripts R, los cuales implementan las funcionalidades de los componentes del prototipo. |
| `report-templates` | La carpeta `report-templates` contiene las plantillas rmarkdown a partir de las cuales se generarán los reportes dinámicos. |
| `www` | La carpeta www contiene archivos CSS, HTML y scripts de JavaScript. |

### 2.2.4. CODIFICACIÓN

Una vez finalizado la instalación y configuración de las herramientas de desarrollo, se muestra a continuación la implementación del *back-end* y *front-end* del prototipo de aplicación web. Para la implementación se han definido múltiples módulos: configuración de parámetros, disciplinas de planificación y gráficos avanzados. A continuación, se presentará un breve resumen de la implementación de cada módulo.

#### 2.2.4.1. Estructura general

Para la codificación de la interfaz gráfica de usuario se utiliza bs4Dash, un paquete desarrollado por el equipo de RinteRface. Este paquete ofrece una plantilla tipo *dashboard* construida sobre Bootstrap[2] 4. El paquete bs4Dash ofrece elementos adicionales que permiten desarrollar aplicaciones Shiny con una apariencia más profesional [43].

Las interfaces del prototipo de aplicación web se las implementó mediante código R de acuerdo con lo establecido en la etapa de diseño. En la Figura 2.14 se puede observar la apariencia predeterminada del *dashboard* proporcionado por el paquete bs4Dash.


**Figura 2.14.** Estructura del prototipo

[2] Bootstrap: Bootstrap es un framework HTML, CSS y JavaScript gratuito y de código abierto para crear rápidamente sitios web responsivos.

El Código 1.1. muestra la implementación de la estructura del prototipo de aplicación web, el cual está compuesto por tres funciones principales:

- **`dashboardHeader`:** La función `dashboardHeader` crea una barra de navegación horizontal que contiene el título de la aplicación y una imagen (opcional).
- **`dashboardSidebar`:** La función `dashboardSidebar` crea la barra lateral principal, en esta sección se implementan las opciones del menú a través de la función `sidebarMenu`.
- **`dashboardBody`:** La función `dashboardBody` crea el contenedor principal del dashboard, en donde se ubica todo el código R necesario para la implementación de las interfaces gráficas de los menús definidos en la barra lateral.

```r
# PAQUETES ########################################################################
library(shiny)

# ENCABEZADO ######################################################################
header <- bs4Dash::dashboardHeader(
  title = bs4Dash::dashboardBrand(title = "SIMULADOR", color = "lightblue"),
  status = "lightblue",
  fixed = TRUE
)

# MENÚ LATERAL ####################################################################
sidebar <- bs4Dash::dashboardSidebar(
  skin = "light",
  status = "lightblue",
  # Menú barra lateral
  bs4Dash::sidebarMenu()
)

# CUERPO ##########################################################################
body <- bs4Dash::dashboardBody(
  # Módulos y componentes del prototipo
  bs4Dash::tabItems()
)

# Define la interfaz de usuario para la aplicación
shiny::shinyUI(
  bs4Dash::dashboardPage(
    title = "SIMULADOR",
    header = header,
    sidebar = sidebar,
    body = body
  )
)
```

**Código 1.1.** Archivo ui.R

El desarrollo del prototipo se lo realizó con las funcionalidades que proveen los diversos paquetes de R, por lo que es necesario importar cada uno de ellos. Sin embargo, los desarrolladores de paquetes R tienden a utilizar los mismos nombres de función, lo que puede provocar un enmascaramiento de nombres. Para evitar este enmascaramiento, se decidió especificar la ruta completa de las funciones de los diferentes paquetes, de acuerdo con la sintaxis `package::function_name`; donde `package` representa el nombre del paquete y `funcion_name` la función requerida.

### 2.2.4.2. Módulo configuración de parámetros

Al iniciar el prototipo de aplicación web, se muestra una interfaz gráfica de usuario similar a la que se observa en la Figura 2.15. Esta interfaz está compuesta por varios *widgets Shiny*, como entradas numéricas, selectores, botones de acción, tablas y más, que permiten al usuario establecer y modificar los parámetros de simulación. Los *widgets* utilizados están disponibles en los paquetes Shiny, bs4Dash y shinyWidgets.

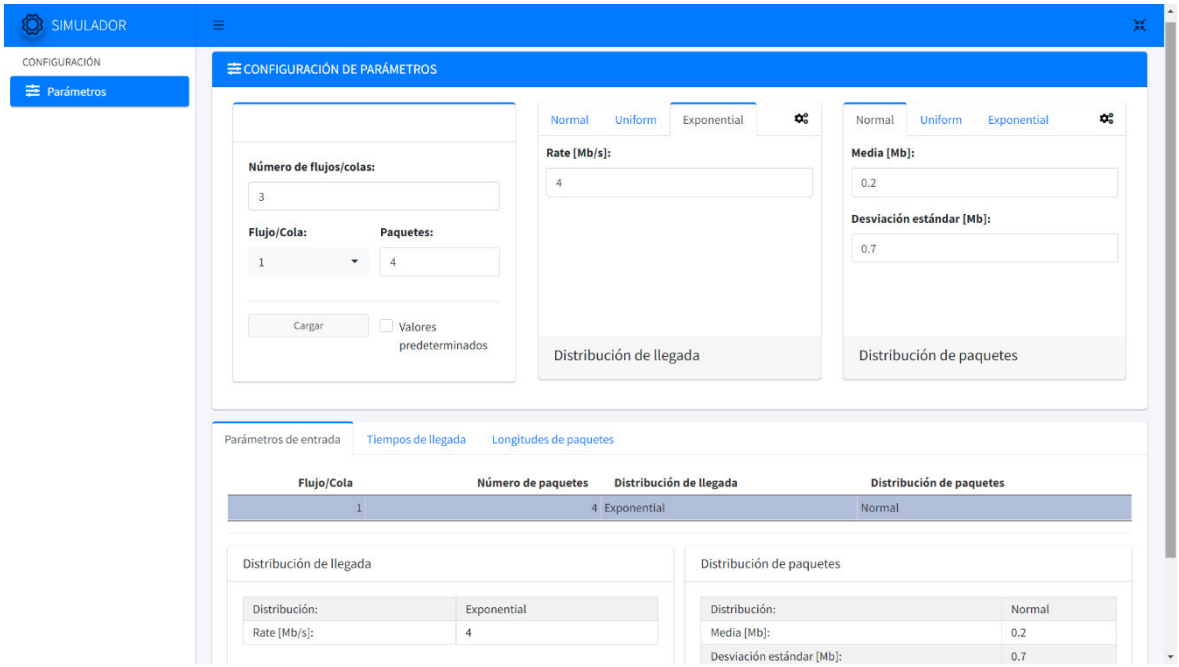


**Figura 2.15.** Interfaz del módulo configuración de parámetros

A continuación, el Código 1.2 muestra las funciones utilizadas para la implementación del primer ítem del menú de la barra lateral, que corresponde a la configuración de parámetros. La función `sidebarHeader` crea un encabezado para los menús. La función `menuItem` crea una pestaña personalizada con los valores que se pasan como argumentos. Es importante tener en cuenta que por cada pestaña (`menuItem`) disponible en el menú de la barra lateral, se debe implementar un elemento de pestaña (`tabItem`) en el cuerpo principal. Finalmente, la función `tooltip` agrega información adicional sobre la pestaña.

```
# Menú configuración de parámetros
bs4Dash::sidebarHeader(title = "CONFIGURACIÓN"),
bs4Dash::menuItem(
  text = "Parámetros",
  tabName = "settings",
  icon = icon("sliders-h")
) %>% bs4Dash::tooltip("Configuración de parámetros", placement = "bottom"),
```

**Código 1.2.** Menú de configuración de parámetros

A modo de ejemplo, el Código 1.3 muestra la implementación del botón “Cargar” y del *CheckBox* “Valores predeterminados” de la interfaz gráfica del módulo de configuración de parámetros.

```
shiny::fluidRow(
  # Botón cargar
  customBtn(
    id = "loadParams",
    label = "  Cargar",
    tooltip = "Cargar parámetros"
  ) %>% shiny::column(width = 6),

  # checkbox valores predeterminados
  shinyWidgets::awesomeCheckbox(
    inputId = "defaultParams",
    label = "Valores predeterminados"
  ) %>%
    shiny::tags$div(`data-tooltip` = awTooltip, `data-flow` = "bottom") %>%
    shiny::column(width = 6)
)
```

**Código 1.3.** Widgets de entrada

Una vez finalizado la codificación de la interfaz gráfica, se procede con la codificación de la lógica del servidor de acuerdo con lo establecido en la etapa de diseño. Como se puede observar en el Código 1.4, el bloque `eventReactive` depende solo de la entrada reactiva `input$loadParams`, esto asegura que *Shiny* haga la mínima cantidad de trabajo. De esta forma, el bloque reactivo solo se ejecuta cuando el valor de `input$loadParams` cambia (el botón “Cargar” es presionado). Cuando una salida requiere un valor reactivo y se presiona el botón “Cargar”, el bloque reactivo toma las entradas y produce una salida (`fmtParams`). Cuando el contenido de `fmtParams` cambia, la salida se actualiza automáticamente.

```
# Módulo 1: Configuración de parámetros ###################################

# Variable auxiliar
aux <- reactiveValues(sp = list(), dp = list())

# Obtener los datos de la interfaz (bloque reactivo)
fmtParams <- eventReactive(input$loadParams, {
  shiny::req(paramsVal$is_valid())
  params <- setData(input)  # Establece los parámetros de entrada
  index <- params[[1]][[1]]
  aux$sp[[index]] <- params[[1]]
  aux$dp[[index]] <- params[[2]]
  showCustomNotification(paste0("Cola: ", index, ". ", msg1))
  aux
}, label = "formatData")

# Módulo Shiny: Resumen de parámetros (salida)
observeEvent(fmtParams(), {
  shiny::req(fmtParams())
  summaryServer(
    id = "paramsSummary",
    simulationParams = fmtParams()$sp,
    distParams = fmtParams()$dp,
    arrivalTime = arrivalTime(),
    packetSize = packetSize()
  )
}, label = "paramsSummary")
```

**Código 1.4.** Configuración de parámetros (lógica del servidor)

### 2.2.4.3. Módulo de disciplinas de planificación

Al seleccionar una disciplina de planificación de paquetes, en la pantalla se muestra una interfaz gráfica de usuario similar a la que se observa en la Figura 2.16 (sección de simulación), en donde el usuario debe establecer un valor para el campo “Velocidad de transmisión” previo a la simulación de la disciplina seleccionada. En esta sección también está disponible un botón de descarga para generar el reporte de la disciplina seleccionada.

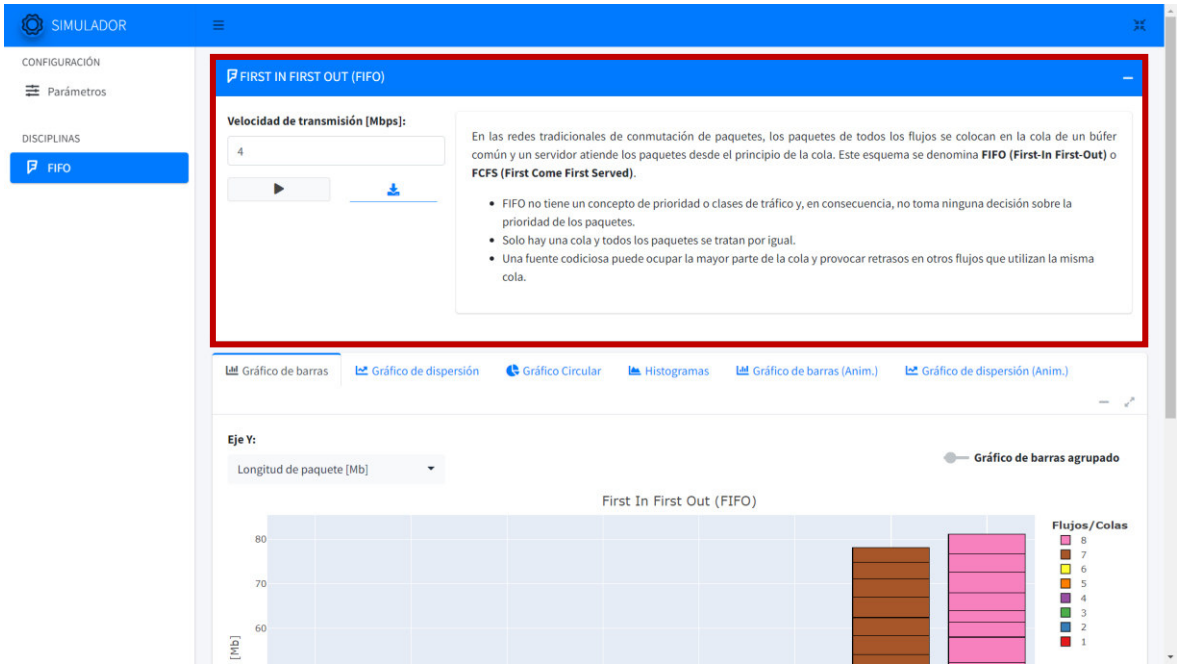


**Figura 2.16.** Interfaz del módulo disciplinas de planificación (sección de simulación)

El Código 1.5 muestra la codificación de la función `disciplineSettingsUI`, la cual genera los *widgets* de la sección de simulación (cuadro rojo). Esta función contiene cuatro parámetros, que se describen a continuación:

- **`title`**: título del encabezado.
- **`id`**: identificador de función.
- **`icon_`**: icono del encabezado.
- **`htmlPath`**: ruta del archivo HTML que contiene una breve información sobre la disciplina de planificación seleccionada.

```r
disciplineSettingsUI <- function(title, id, icon_, htmlPath) {
  fluidRow(
    bs4Dash::box(
      width = 12,
      title = title,
      solidHeader = TRUE,
      status = status,
      icon = icon_,
      elevation = elevation,

      # Interfaz de configuraciónde disciplina
      fluidRow(
        column(
          width = 3,
          # Tasa de transmisión
          shiny::numericInput(
            inputId = paste0(id, "Tr"),
            label = "Velocidad de transmisión [Mbps]:",
            min = minVtx,
            max = maxVtx,
            value = 4
          ) %>% bs4Dash::tooltip(tsMsg, placement = "bottom"),

          shiny::fluidRow(
            # Boton simular
            customBtn(id = paste0(id, "Simulation"), icon = icon("play")) %>%
              column(width = 6),
            # Módulo Shiny: Reportes
            reportUI(id = paste0(id, "Report")) %>% column(width = 6)
          )
        ),
        # Información adicional
        column(width = 9, includeHTML(path = htmlPath))
      )
    )
  )
}
```

**Código 1.5.** Disciplinas de planificación (Sección de simulación)

A continuación, el Código 1.6 muestra el código utilizado para la implementación de la lógica del servidor del módulo de disciplinas de planificación (FIFO). El bloque reactivo solo depende del valor de la variable reactiva `input$fifoSimulation`, por lo que el computo de la disciplina empezará solo cuando el valor de esta variable cambie, es decir, cuando se presione el botón “Simular”. Las salidas se actualizarán inmediatamente después de que finalice el procesamiento del bloque reactivo.

```
## Disciplina FIFO ############################################################

# Figuras FIFO
fifoFigs <- reactiveValues(
  bar = NULL,
  scatter = NULL,
  pie = NULL,
  hist = NULL
)

# Computo de la disciplina FIFO
fifo <- eventReactive(input$fifoSimulation ,{

  # Validaciones de campo
  shiny::req(fifoValidation$is_valid())
  check(input$defaultParams, input$queuesNumber, arrivalTime(), session)

  # Desactivar botones
  shinyjs::disable(id = "fifoSimulation")
  shinyjs::disable(id = "fifoReport-report_bttn")

  out <- fifoScheduler(arrivalTime(), packetSize(), input$fifoTr)

  # Activar botones
  shinyjs::enable(id = "fifoSimulation")
  shinyjs::enable(id = "fifoReport-report_bttn")

  out  # Salida
}, label = "computeFifo")

# Salida de los resultados de simulación
observeEvent(fifo(), {
  iplotServer("fifoPlots", fifoTooltip, fifo(), fifoFigs) # Gráficos
  dataframeServer("fifoData", fifo()) # Tablas
}, label = "fifoDataOut")
```

**Código 1.6.** Disciplinas de planificación (lógica del servidor)

Una vez que el proceso de simulación ha finalizado, el usuario tiene a su disposición varios *widgets* de salida para la exploración y visualización de los resultados generados en la simulación.

Para la generación de los reportes dinámicos se utiliza un módulo *Shiny* (`reportUI`/`reportServer`). El módulo *Shiny* report permite que los resultados de la simulación estén disponibles para el usuario como archivos descargables. El Código 1.7 muestra la implementación del módulo report.

```r
reportUI <- function(id) {
  shinyWidgets::downloadBttn(
    outputId = NS(id, "report"),
    label = "", style = "minimal", size = "sm", block = TRUE
  ) %>%
    tags$div(`data-tooltip` = "Generar reporte", `data-flow` = "bottom") %>%
    shinyjs::disabled()
}

reportServer <- function(id, type, params, data, ts, figs, w = NULL) {
  moduleServer(id, function(input, output, session) {
    # Generar reportes
    output$report <- downloadHandler(
      filename = function() {
        paste0(type, "-", Sys.Date(), ".", "html")
      },
      content = function(file) {
        # Ruta temporal del archivo a renderizar
        temp_report <- file.path(tempdir(), paste0(type, ".Rmd"))

        # Copia la plantilla del reporte a una dirección temporal
        file.copy(
          from = paste0("./report-templates/", "html", "/", type, ".Rmd"),
          to = temp_report, overwrite = TRUE
        )

        # Parámetros a pasar al reporte
        params <- list(
          parameters = params$sp, distributions = params$dp,
          ts = ts, data = data, figs = figs, w = w
        )

        # Renderizar reporte
        rmarkdown::render(
          input = temp_report, output_file = file,
          params = params, envir = new.env(parent = globalenv())
        )
      }
    )
  })
}
```

**Código 1.7.** Módulo Shiny `report`

#### 2.2.4.4. Módulo de gráficos avanzados

Cómo se puede observar en la Figura 2.17, el paquete Esquisse proporciona una interfaz gráfica que cuenta con varios *widgets* para la creación, exploración y visualización interactiva de los datos.


**Figura 2.17.** Interfaz del módulo gráficos avanzados

A continuación, en el Código 1.8 se puede observar la implementación de la interfaz gráfica del módulo `esquisse_ui`. Para lo cual, se establece un identificador (id), un vector de caracteres (`controls`), que especifica los controles que se mostrarán en el menú de la interfaz de Esquisse. Adicionalmente se establece la altura del contenedor del módulo.

```
# Módulo 3: Gráficos avanzados ############################################
bs4Dash::tabItem(
  tabName = "advancedGraphs",
  esquisse::esquisse_ui(
    id = "esquisse",
    controls = c("labs", "appearance", "filters"),
    container = esquisse::esquisseContainer(height = "90vh")
  )
)
```

**Código 1.8.** Implementación de la interfaz gráfica de Esquisse

En el Código 1.9, se muestra la implementación de la lógica del módulo de gráficos avanzados. La variable reactiva `data_rv` almacena un conjunto de datos predeterminado. La función `esquisse_server` define la lógica del módulo. En esta función se define un identificador (`id`), un conjunto de datos de entrada (`data_rv`), y la fuente de los datos (`file`).

```
# Módulo 3: Gráficos avanzados ############################################
# Conjunto de datos pretederminado
data_rv <- reactiveValues(data = defaultData, name = "wfq")

# Configuración del servidor de esquisse
esquisse::esquisse_server(
  id = "esquisse",
  data_rv = data_rv,
  import_from = "file"
)
```

**Código 1.9.** Implementación del servidor

Los códigos anteriores representan solo una parte del código total necesario para la implementación de los módulos y sus componentes. El código completo junto a la documentación de los módulos y funciones está disponible en el ANEXO E.

### 2.2.5. DESPLIEGUE DE LA APLICACIÓN WEB

Como ya se mencionó en secciones anteriores, el prototipo de aplicación web estará alojado en la plataforma shinyapps.io, esto con el fin de hacer que el prototipo esté disponible para todos los usuarios a través de internet.

Para el despliegue del prototipo en la plataforma shinyapps.io, se necesita que el paquete rsconnect esté autorizado a una cuenta de shinyapps.io mediante un *token* y una llave secreta. Para acceder a los tokens se debe iniciar sesión en shinyapps.io, y en el menú del *dashboard* seleccionar *Tokens* | *Show*. La Figura 2.18 muestra el *token* y la llave secreta de la cuenta.


**Figura 2.18.** Tokens

El siguiente paso consiste en la configuración de la información de la cuenta de shinyapps.io en el IDE RStudio Desktop. Para lo cual, se utiliza el comando `setAccountInfo` con los argumentos especificados en la Figura 2.19.


**Figura 2.19.** Configuración de la información de la cuenta

Una vez que se ha configurado la información de la cuenta en RStudio, se ejecuta el comando `rsconnect::deployApp()` desde el directorio de trabajo principal. En la Figura 2.20 se ilustra el proceso de despliegue del prototipo en shinyapps.io.

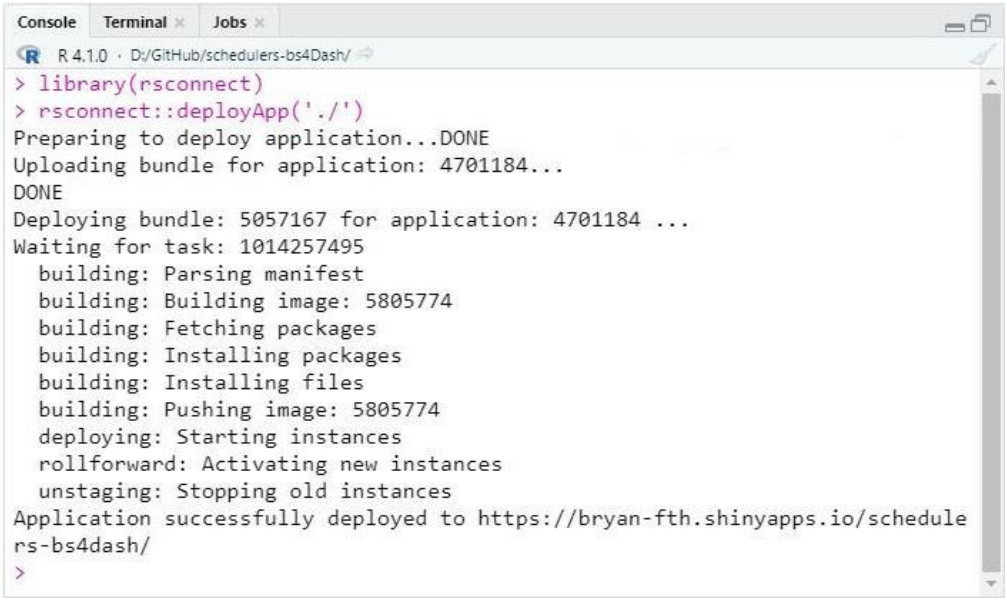


**Figura 2.20.** Despliegue del prototipo de aplicación

Cuando finaliza el proceso de despliegue, el prototipo recién implementado inicia automáticamente en el navegador predeterminado del sistema, como se ilustra en la Figura 2.21.

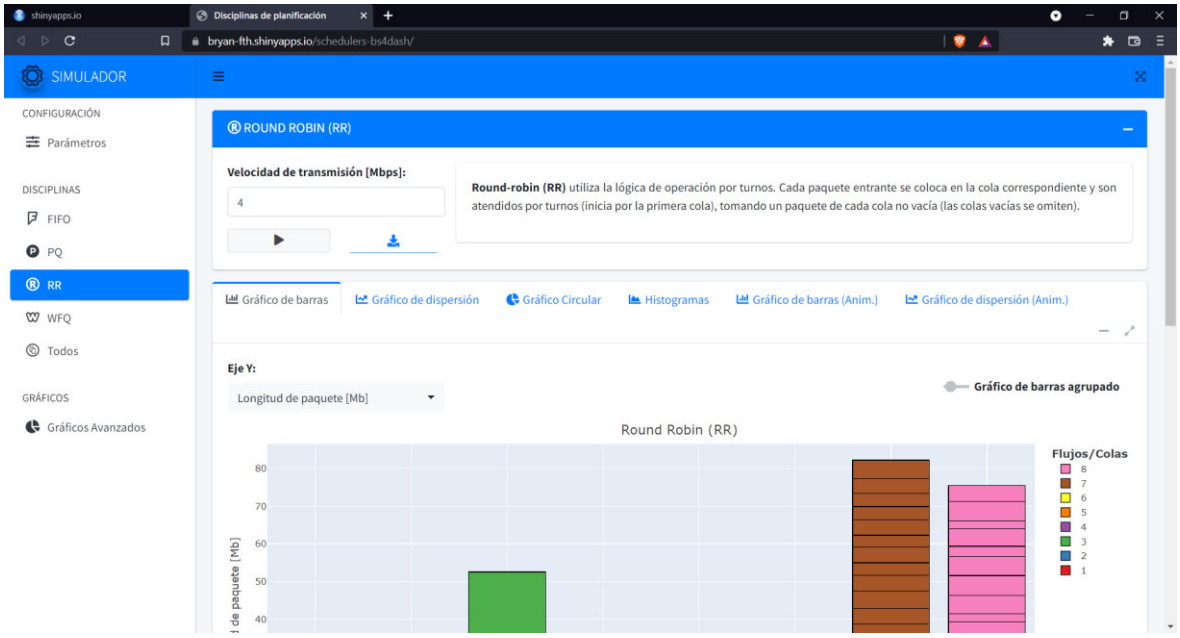


**Figura 2.21.** Prototipo de aplicación web

# 3. RESULTADOS Y DISCUSIÓN

En esta sección se presentan las validaciones de los requerimientos funcionales y no funcionales, validación de resultados y pruebas automatizadas (pruebas de rendimiento) realizadas al prototipo. Por último, se detallan los errores que se encontraron y sus respectivas correcciones.

En el ANEXO F se incluye un enlace a dos videotutoriales, en los que se muestra la estructura del prototipo y un ejemplo de simulación.

## 3.1. ACTUALIZACIÓN DEL TABLERO KANBAN

Las actividades del 8 al 13, de la Tabla 2.1, relacionadas con la implementación del prototipo han finalizado, por lo que pasan de la columna “En Proceso” a la columna “Finalizado”. El siguiente paso consiste en la realización de las pruebas al prototipo y la corrección de errores, por lo que las actividades del 14 al 18, de la Tabla 2.1, pasan de la columna “Por Hacer” a la columna “En Proceso” del tablero Kanban.

## 3.2. PRUEBAS DE VALIDACIÓN DE LOS REQUERIMIENTOS FUNCIONALES

En esta sección se realiza la validación de los requerimientos funcionales del prototipo de aplicación web descritos en la sección 2.1.2.3.1.

### 3.2.1. PRUEBA DE VALIDACIÓN DEL RF01: CONFIGURACIÓN DE PARÁMETROS DE SIMULACIÓN

Se procede a verificar el cumplimiento del requerimiento funcional RF01. La interfaz gráfica principal del prototipo (configuración de parámetros) cuenta con varios *widgets* de entrada para el establecimiento de los parámetros de simulación: número de flujos / colas, número de paquetes y distribuciones de probabilidad. El usuario establece los valores de los diferentes campos de acuerdo con sus necesidades y da clic en cargar. Si los valores establecidos son correctos, una notificación indicará al usuario que los parámetros se han establecido correctamente para el flujo o cola seleccionado. En la Figura 3.1 se puede observar el cumplimiento del requerimiento RF01.

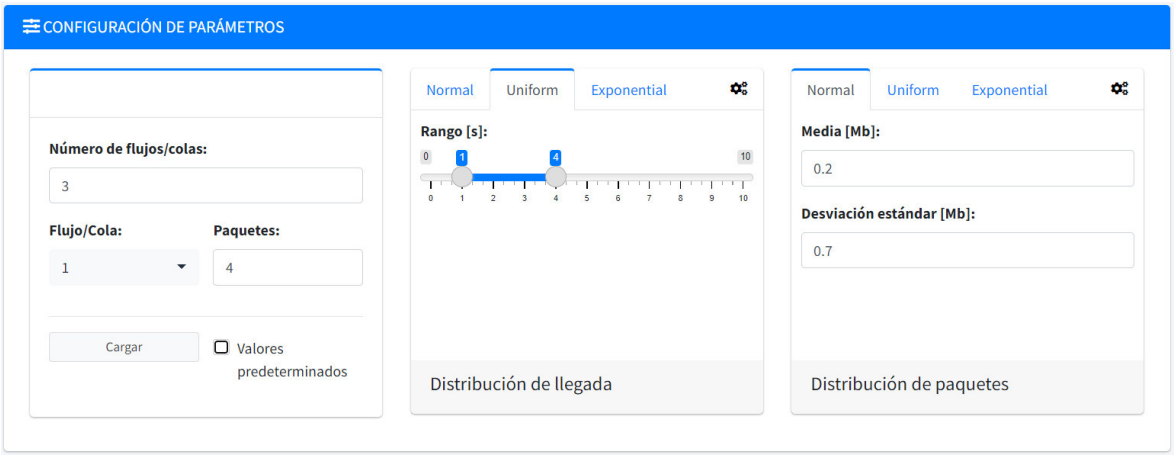


**Figura 3.1.** Configuración de los parámetros de simulación

Los requerimientos funcionales especifican que la velocidad de transmisión del enlace debe ser un parámetro configurable por el usuario, sin embargo, se ha decidido no incluirlo en esta interfaz. En su lugar, este parámetro estará disponible de forma independiente para cada disciplina de planificación, como se observa más adelante.

### 3.2.2. PRUEBA DE VALIDACIÓN DE RF02: ACTUALIZACIÓN DE LOS PARÁMETROS DE SIMULACIÓN

Cuando el usuario configura con éxito los parámetros de un flujo o cola, la tabla de resumen de parámetros se actualiza automáticamente para mostrar estos nuevos valores. Adicionalmente, se puede seleccionar una fila para mostrar información adicional sobre las distribuciones de probabilidad de cada flujo/cola. La Figura 3.2 muestra el cumplimiento del requerimiento RF02, el cual consiste en la visualización de los parámetros configurados.

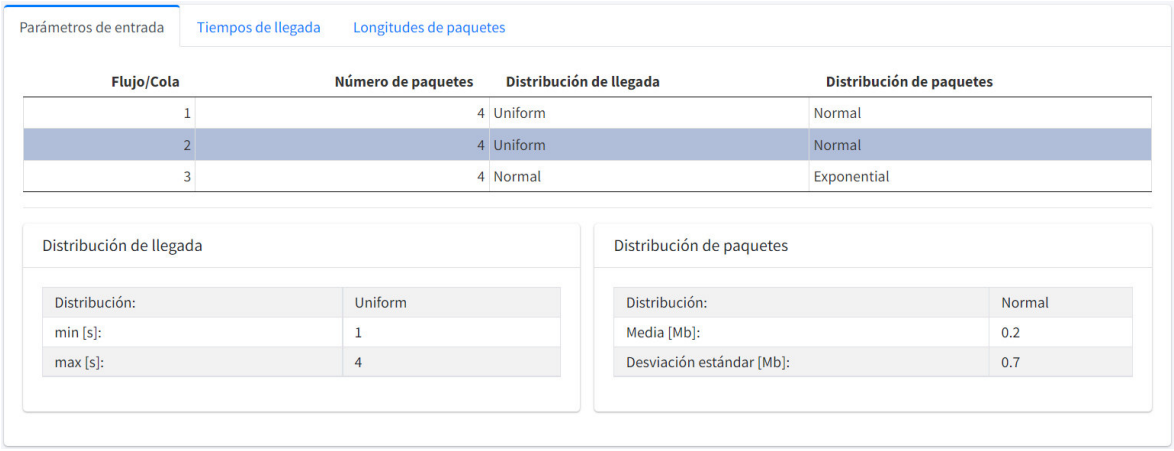


**Figura 3.2.** Visualización de parámetros configurados

Los usuarios pueden modificar los parámetros previamente definidos en cualquier momento. Para lo cual se debe seleccionar el flujo o cola a modificar, actualizar los valores

de los campos requeridos y cargar nuevamente. La Figura 3.3 muestra que los datos se han modificado con éxito.

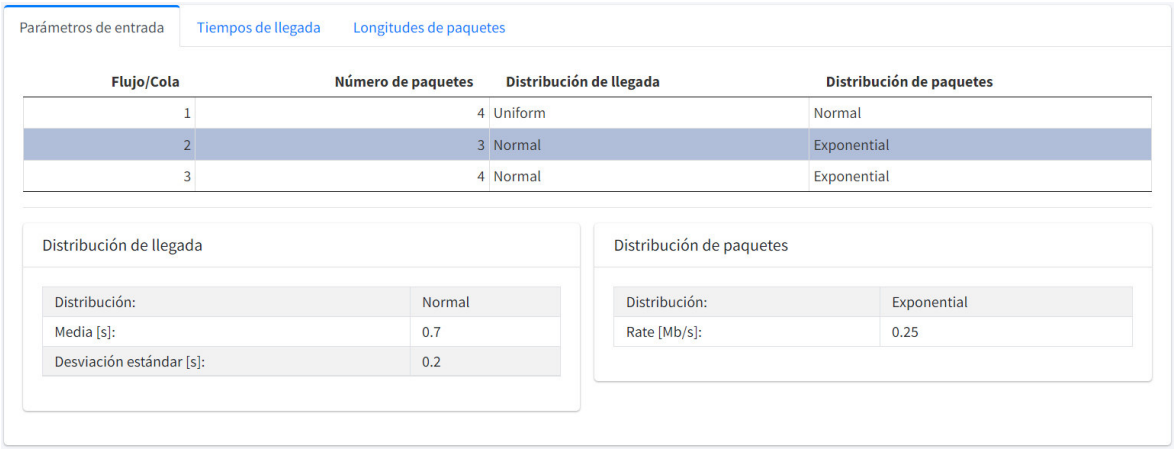


**Figura 3.3.** Actualización de los parámetros de simulación

### 3.2.3. PRUEBA DE VALIDACIÓN DE RF03: VISUALIZACIÓN DE RESULTADOS

La Figura 3.4 muestra la interfaz gráfica de simulación para la disciplina de planificación FIFO. Esta interfaz se divide en tres secciones: control de simulación, gráficas interactivas y datos de simulación. Una vez que se ha configurado con éxito los parámetros iniciales, el usuario selecciona la disciplina que se desea simular, establece el valor de la velocidad de transmisión del enlace y da clic en el botón “Simular”.


**Figura 3.4**. Interfaz de simulación de la disciplina FIFO

En función de los datos de la simulación, hay diferentes tipos de gráficos interactivos (gráficos de barras, dispersión, circulares e histogramas) para ofrecer una mejor visualización de los datos. En la Figuras 3.5 se observan los gráficos que se obtienen en la simulación de la disciplina de planificación FIFO.

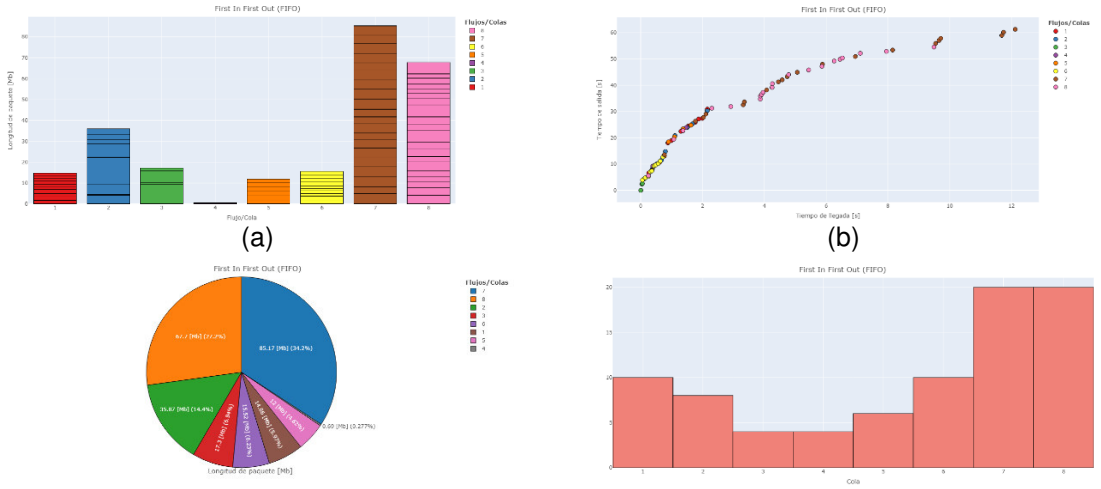


**Figura 3.5.** Gráficos interactivos

Como se observa en la Figura 3.6, en la tabla de datos dinámica se muestran los valores que se generaron durante la simulación de la disciplina.

Datos

| Cola | Longitud de paquete [Mb] | Tiempo de llegada [s] | Tiempo de salida [s] | Tiempo de espera [s] | Tiempo de servicio [s] | Tiempo final [s] |
|---|---|---|---|---|---|---|
| 3 | 9.48357 | 0.00293 | 0.00293 | 0 | 2.37089 | 2.37382 |
| 3 | 0.84211 | 0.03552 | 2.37382 | 2.3383 | 0.21053 | 2.58435 |
| 3 | 5.389 | 0.05372 | 2.58435 | 2.53063 | 1.34725 | 3.9316 |
| 6 | 1.92795 | 0.05786 | 3.9316 | 3.87374 | 0.48199 | 4.41358 |
| 3 | 1.5892 | 0.11668 | 4.41358 | 4.2969 | 0.3973 | 4.81088 |
| 6 | 1.93132 | 0.14952 | 4.81088 | 4.66137 | 0.48283 | 5.29372 |
| 6 | 1.22688 | 0.24857 | 5.29372 | 5.04515 | 0.30672 | 5.60044 |
| 8 | 4.10402 | 0.24873 | 5.60044 | 5.35171 | 1.02601 | 6.62644 |
| 1 | 1.71632 | 0.25977 | 6.62644 | 6.36667 | 0.42908 | 7.05552 |
| 6 | 1.42356 | 0.30518 | 7.05552 | 6.75034 | 0.35589 | 7.41141 |

Mostrando registros del 1 al 10 de un total de 82 registros

Anterior 1 2 3 4 5 … 9 Siguiente

**Figura 3.6.** Visualización de resultados (tabla de datos de simulación)

### 3.2.4. PRUEBA DE VALIDACIÓN DE RF04: DESCARGA DE DATOS DE SIMULACIÓN

El usuario puede descargar un archivo con los datos de los resultados de la simulación de la disciplina de planificación seleccionada. Para su descarga, los datos están disponibles en formato CSV y TXT, como se observa en la Figura 3.7.

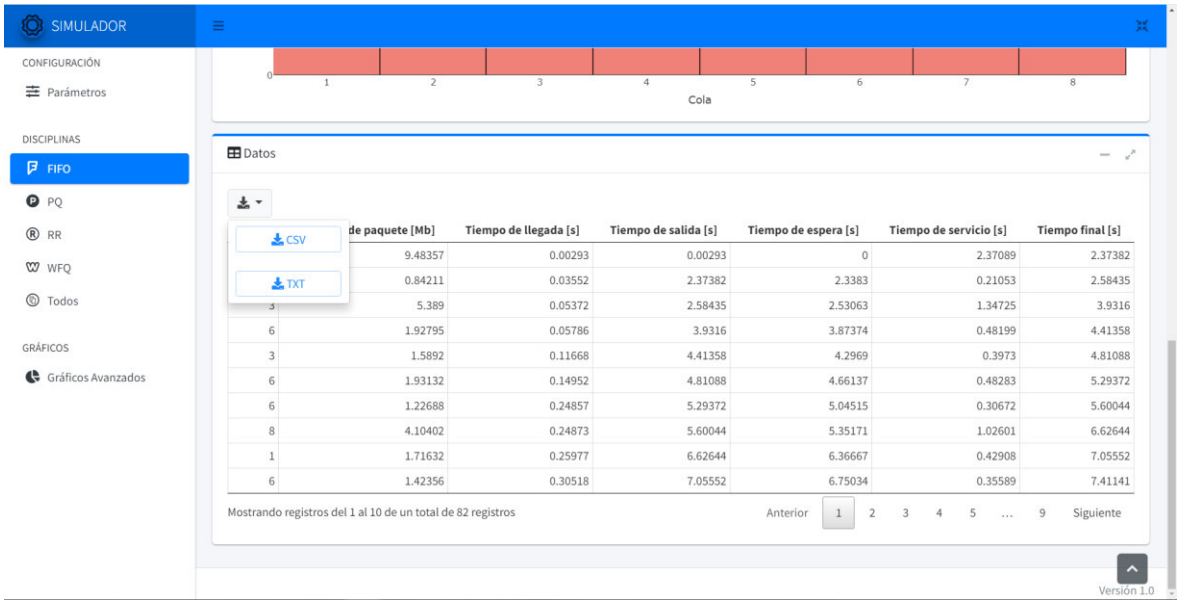


**Figura 3.7.** Descarga de datos de simulación (controles de descarga)

A continuación, a modo de ejemplo, la Figura 3.8 muestra los archivos CSV y TXT que se generan al descargar los datos de simulación.

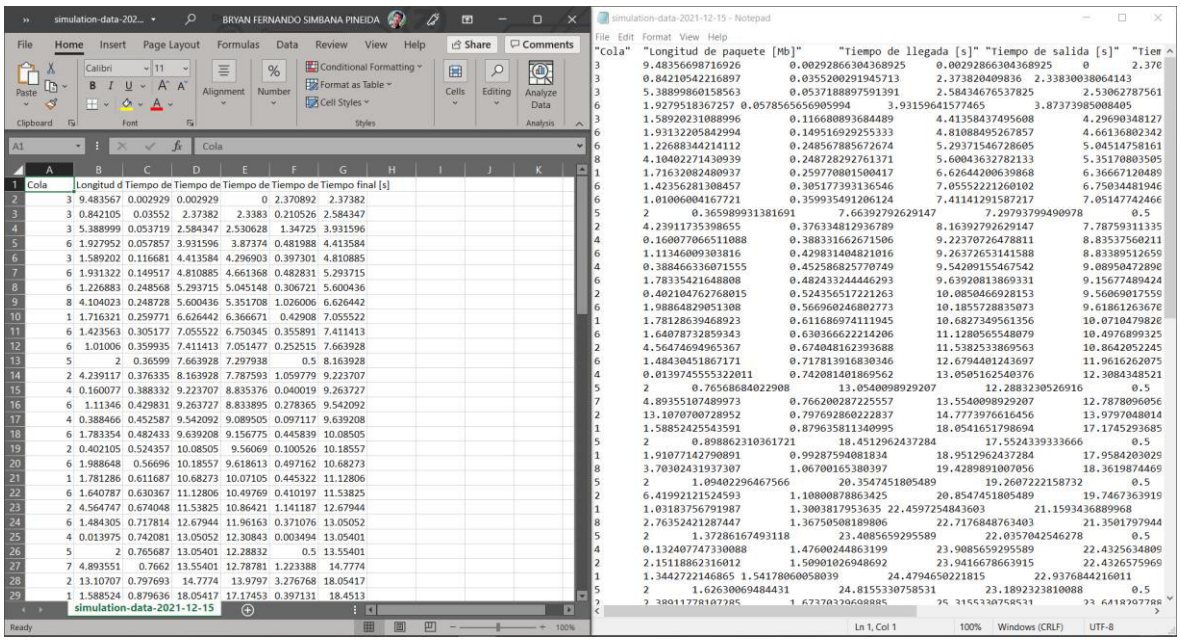


**Figura 3.8.** Datos de simulación (CSV y TXT)

### 3.2.5. PRUEBA DE VALIDACIÓN DE RF05: REPORTES DE SIMULACIÓN

El usuario puede descargar un reporte de los resultados de la simulación de la disciplina de planificación de paquetes seleccionada. El informe generado incluye un resumen de los parámetros configurados y los resultados obtenidos (gráficos y tablas). La Figura 3.9

muestra una ventana emergente para elegir la ubicación de almacenamiento del reporte dinámico generado.

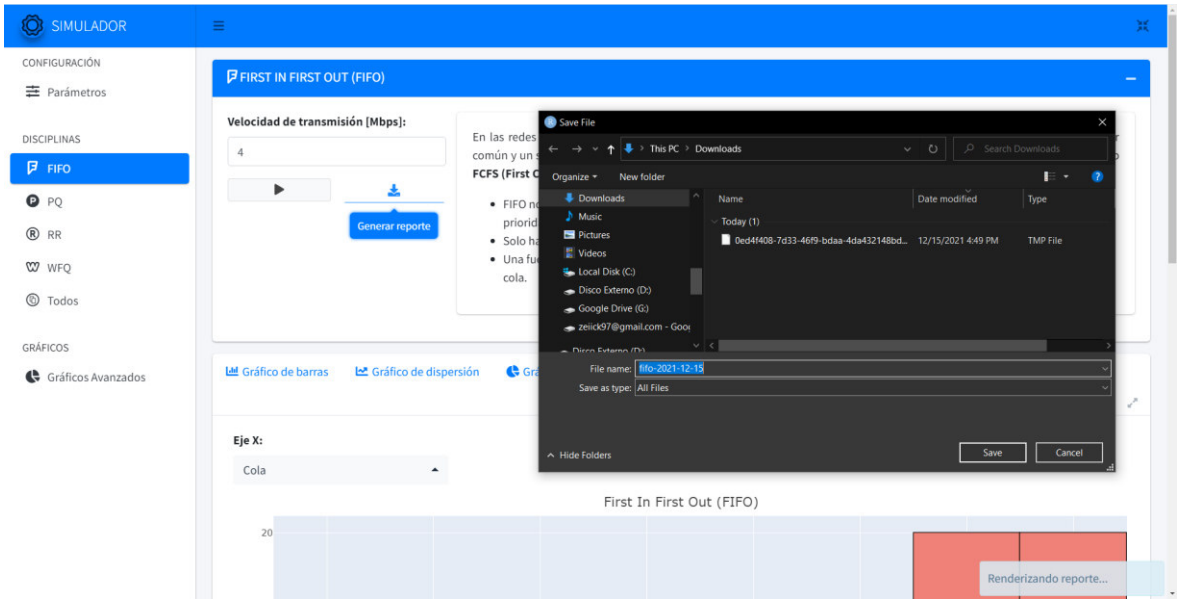


**Figura 3.9.** Generación de deportes

En el ANEXO G se incluye a modo de ejemplo, un reporte generado para la disciplina de planificación FIFO.

### 3.2.6. PRUEBA DE VALIDACIÓN DE RF06: GRÁFICOS AVANZADOS

En la pestaña de gráficos avanzados se tiene la posibilidad de crear gráficos personalizados mediante un archivo de datos de simulación externo. El primer paso consiste en dar clic en el botón “Cargar datos”. En pantalla aparece una ventana modal para seleccionar la ubicación de los datos a importar. En esta ventana también se tiene la posibilidad de modificar los atributos de los datos, como el nombre de las columnas, el tipo de datos, entre otros.

Una vez se ha hecho los cambios necesarios, se da clic en “Importar datos”. El siguiente paso consiste en mapear las variables en los campos disponibles (ejes, color, tamaño, entre otros). Esquisse selecciona de forma automática el tipo de datos que mejor se adapte con las características seleccionadas. Esquisse cuenta con varias funcionalidades, entre la que destacan las siguientes: establecimiento de título, subtitulo, etiquetas de los ejes, y colores de los gráficos. La Figura 3.10 muestra el cumplimiento del requerimiento RF06.

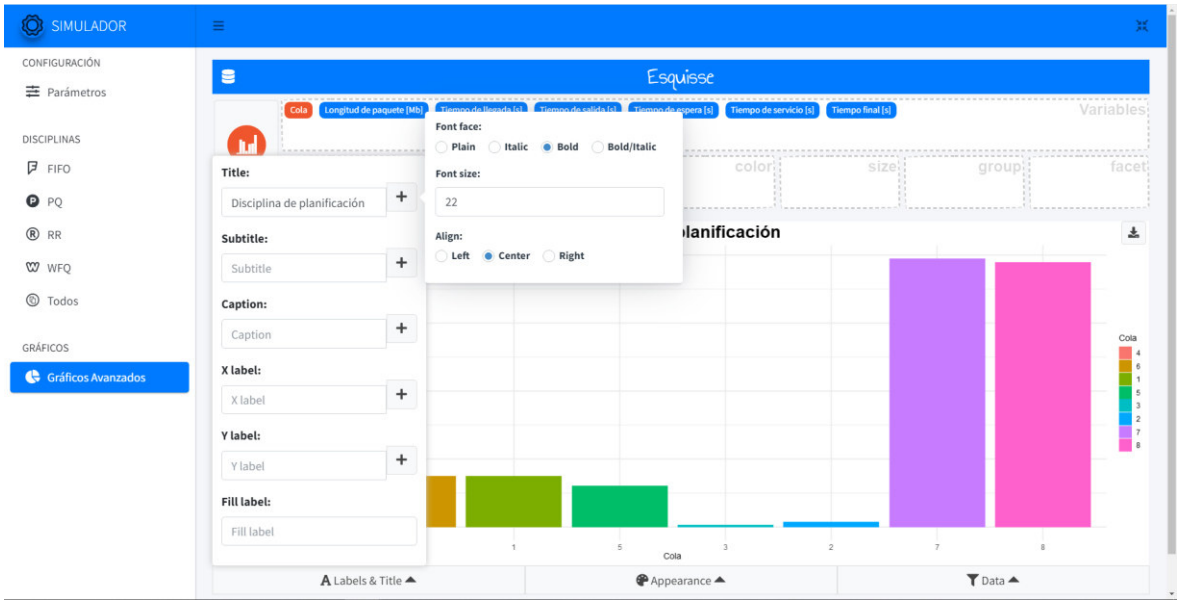


**Figura 3.10.** Gráficos avanzados

### 3.2.7. PRUEBA DE VALIDACIÓN DE RF07: DESCARGA DE GRÁFICOS DE SIMULACIÓN

Una vez que el usuario ha finalizado la creación del gráfico personalizado, se puede descargar las imágenes generadas, para lo cual se selecciona el icono de descarga ubicado en la parte superior derecha del contenedor de la imagen. Como se observa en la Figura 3.11, los formatos de descarga disponibles son: PNG, JPG/JPEG y PDF.

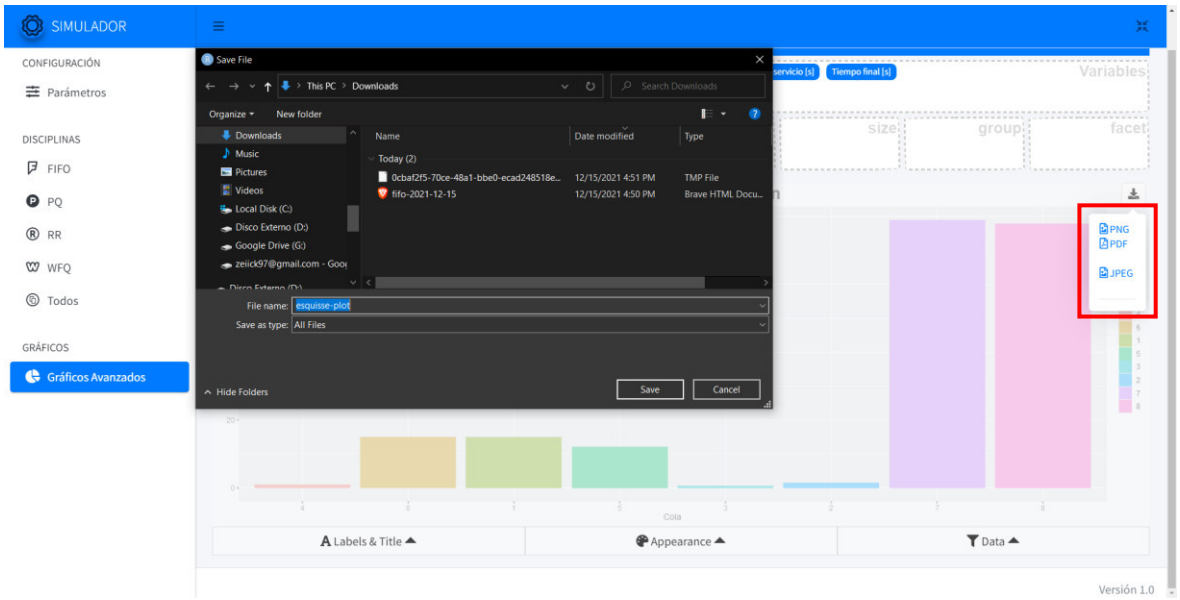


**Figura 3.11.** Descarga de gráficos personalizados

Una vez se ha finalizado las pruebas de validación de los requerimientos funcionales, se puede observar que el prototipo cumple con las especificaciones definidas en la sección de requerimientos funcionales.

## 3.3. PRUEBAS DE VALIDACIÓN DE LOS REQUERIMIENTOS NO FUNCIONALES

En esta sección se realiza la validación de los requerimientos no funcionales del prototipo de aplicación web descritos en la sección 2.1.2.3.2.

### 3.3.1. PRUEBA DE VALIDACIÓN DE RNF01: PRESENTACIÓN Y USABILIDAD

La interfaz del prototipo proporciona varios *widgets* distribuidos de forma cuidadosa. Cada *widget* proporciona una etiqueta con un mensaje claro, además de un *tooltip* (Información sobre herramientas) que proporciona información adicional sobre el propósito del *widget*. Todos los *widgets* proporcionan valores predeterminados, lo que ayuda a reducir la carga del usuario.

De acuerdo con los requerimientos no funcionales, el usuario necesita recibir notificaciones de información, advertencia y error. La Figura 3.12 muestra un ejemplo de alerta de error emitida por la aplicación cuando se intenta simular una disciplina de planificación sin configurar sus parámetros.

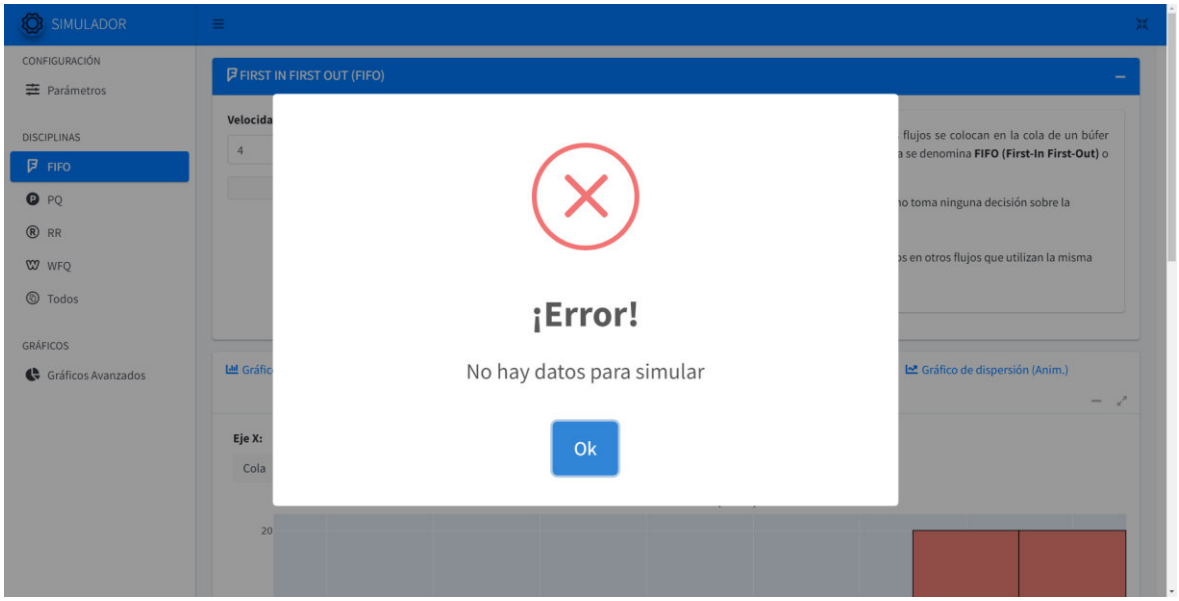


**Figura 3.12.** Notificación de error

Cabe destacar que las interfaces de usuario de los módulos comparten muchos elementos (*widgets*), como botones de acción, selectores y entradas numéricas. Estos elementos comunes hacen posible que los usuarios se familiaricen con su uso de una forma rápida y sencilla.

### 3.3.2. PRUEBA DE VALIDACIÓN DE RNF02: RENDIMIENTO

Site14x7 es un servicio de monitoreo del rendimiento de sitios web, servidores, y aplicaciones web. El servicio de monitoreo permite analizar los componentes de un sitio alojado en internet y en base a unas métricas medir su rendimiento. Al analizar un sitio o aplicación web, el servicio de monitoreo proporciona cuatro resultados: puntuación de PageSpeed, tiempo de carga de la página, número de solicitudes y tamaño del papel [44]. La Figura 3.13 ilustra los resultados obtenidos para el prototipo de aplicación web.

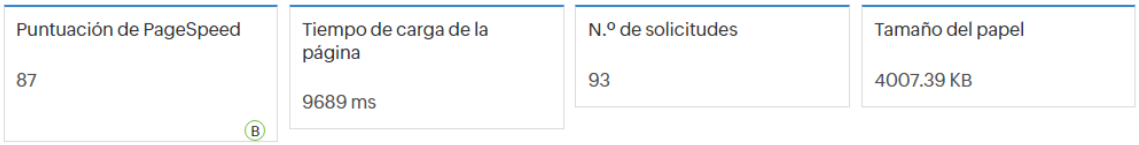


**Figura 3.13.** Métricas de rendimiento

PageSpeed es un puntaje otorgado por Site24x7, sobre 100, por sus herramientas de monitoreo, que toma las métricas sin procesar y las convierte en una puntuación de entre 1 y 100. El tiempo de carga de la página es el tiempo en segundos que la aplicación se demoró en estar disponible para el usuario. El número de solicitudes hace referencia a las peticiones realizadas por el navegador web (cliente) al servidor durante la carga de la página o aplicación web. Por último, el tamaño del papel es el tamaño en *Kilobytes* (KB) de todos los componentes que conforman el prototipo, como archivos de scripts, CSS, HTML, imágenes y más [44].

Adicionalmente, para comprobar el rendimiento del prototipo con múltiples usuarios simultáneos se realizó una prueba de carga con el paquete `shinyloadtest`. El paquete genera automáticamente un *dashboard* con varias pestañas de información sobre diferentes métricas de rendimiento del prototipo.

La Figura 3.14 ilustra la duración de varias sesiones simultáneas del prototipo. Como se puede observar en la figura, todas las sesiones de inicio (*Homepage* y JS/CSS) terminan aproximadamente al mismo tiempo que las demás, por lo que se tiene un comportamiento constante aun cuando el prototipo es utilizado por varios usuarios simultáneos.

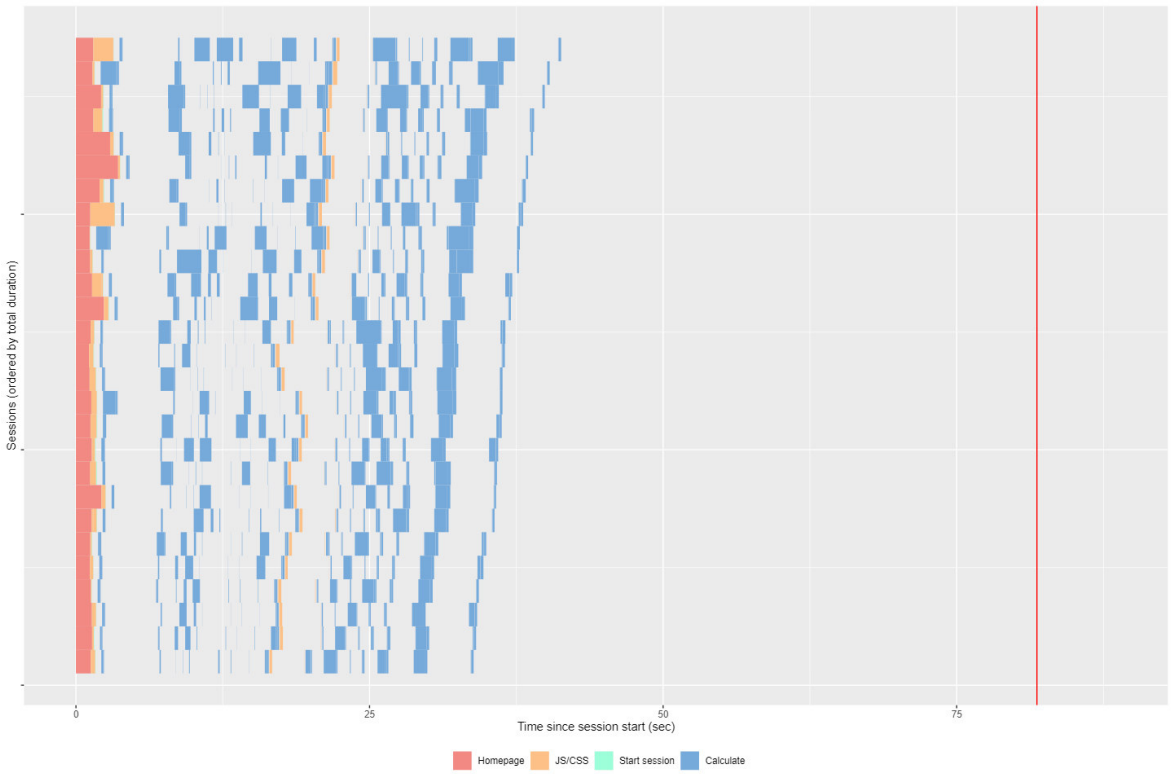


**Figura 3.14.** Duración de sesión

La Figura 3.15 muestra la latencia total (tiempo de carga) de HTTP y archivos auxiliares (JS/CSS). Las barras verticales representan la cantidad de tiempo que la aplicación *Shiny* de una sesión tuvo que esperar antes de poder solicitar información al servidor.

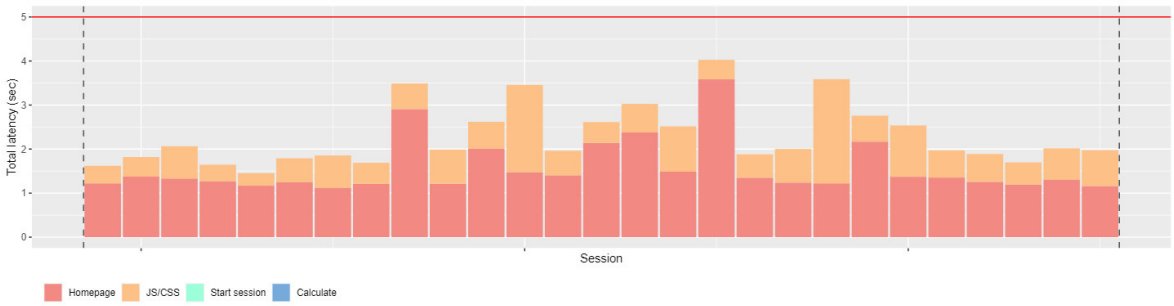


**Figura 3.15.** Latencia

El informe completo del rendimiento del prototipo web está disponible en el ANEXO H.

### 3.3.3. PRUEBA DE VALIDACIÓN DE RNF03: MANTENIBILIDAD

Para el desarrollo del prototipo de aplicación web se consideró el uso de módulos *Shiny*, lo que permitió simplificar el código del proyecto y mantener una estructura simple. El uso de módulos *Shiny* permite escribir, analizar y probar los componentes individuales de forma

aislada, por lo que la adición, modificación o eliminación de las funcionalidades del prototipo es sencilla.

Además, con el fin de proporcionar una explicación legible en el código fuente, cada función y módulo *Shiny* proporciona información sobre sus argumentos, una descripción de lo que hace y los resultados que retorna.

### 3.3.4. PRUEBA DE VALIDACIÓN DE RNF04: INTEGRIDAD Y VALIDEZ DE LOS DATOS

Los *widgets* numéricos de entrada implementan las siguientes reglas de validación:

- Campos obligatorios
- Valores mínimos y máximos.
- Tipos de datos

La Figura 3.16 muestra la validación de los campos de entrada de la interfaz gráfica principal.

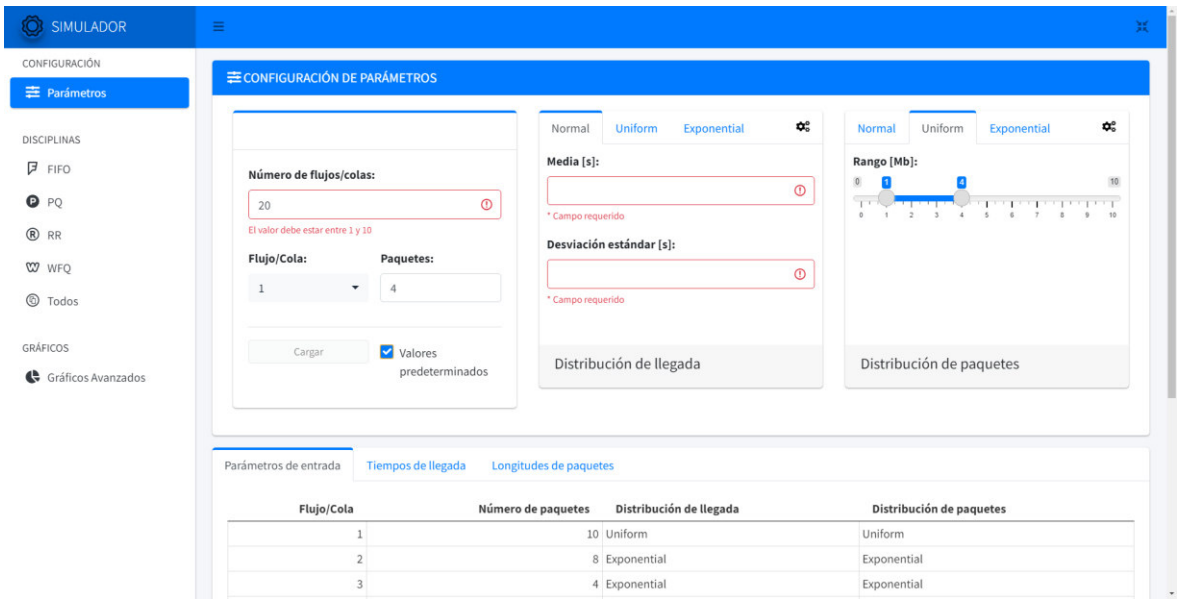


**Figura 3.16.** Validaciones de entrada

Una vez se ha finalizado las pruebas de validación de los requerimientos no funcionales, se puede observar que el prototipo cumple con las especificaciones definidas en la sección de requerimientos no funcionales.

## 3.4. PRUEBAS DE COMPATIBILIDAD

Para comprobar que el prototipo de aplicación web funciona correctamente en diferentes navegadores, se procede a ejecutar el prototipo en los navegadores: Google Chrome, Mozilla Firefox y Microsoft Edge. Las Figuras 3.17, 3.18 y 3.19 muestran que el prototipo se ejecuta sin problemas en los navegadores mencionados.

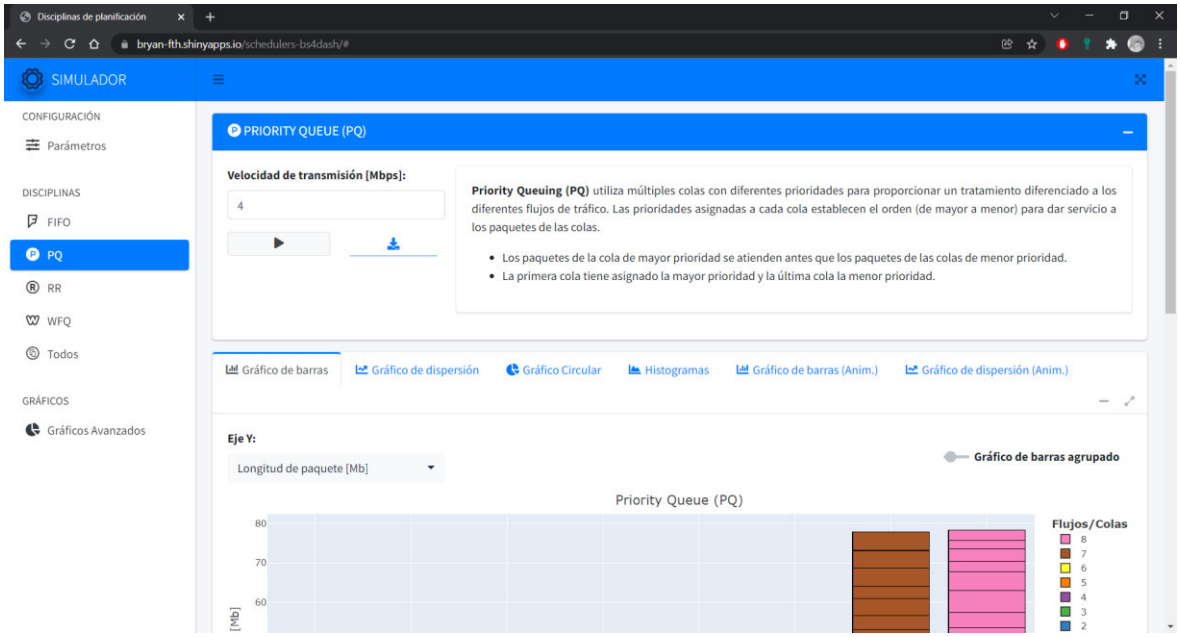


**Figura 3.17.** Ejecución del prototipo de aplicación web (Google Chrome)

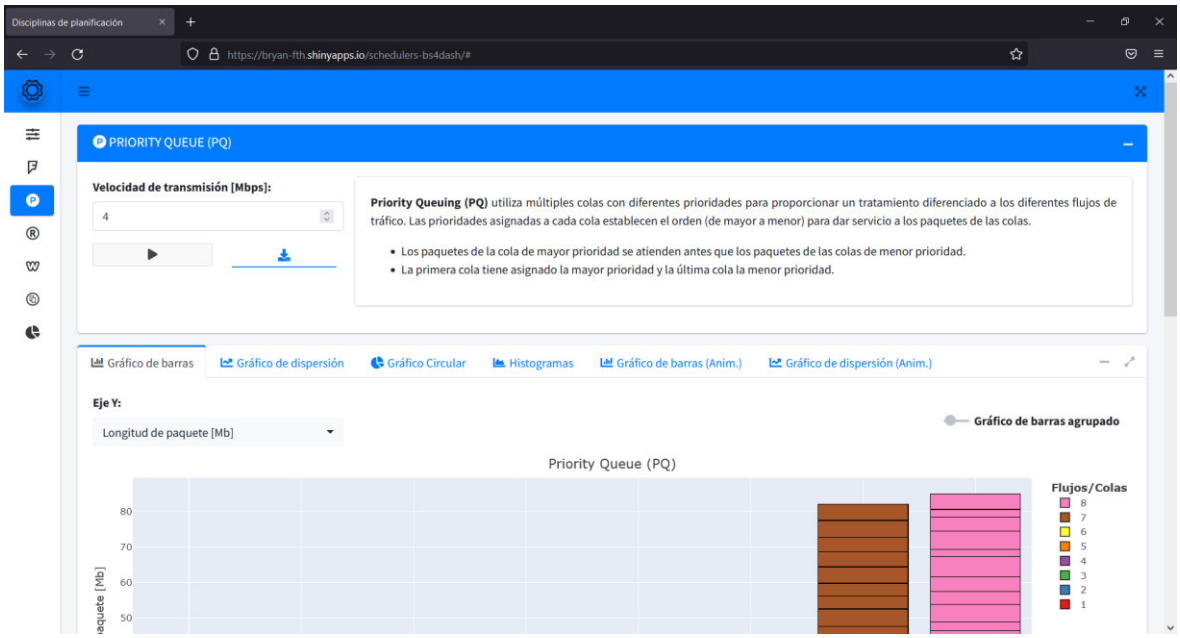


**Figura 3.18.** Ejecución del prototipo de aplicación web (Mozilla Firefox)

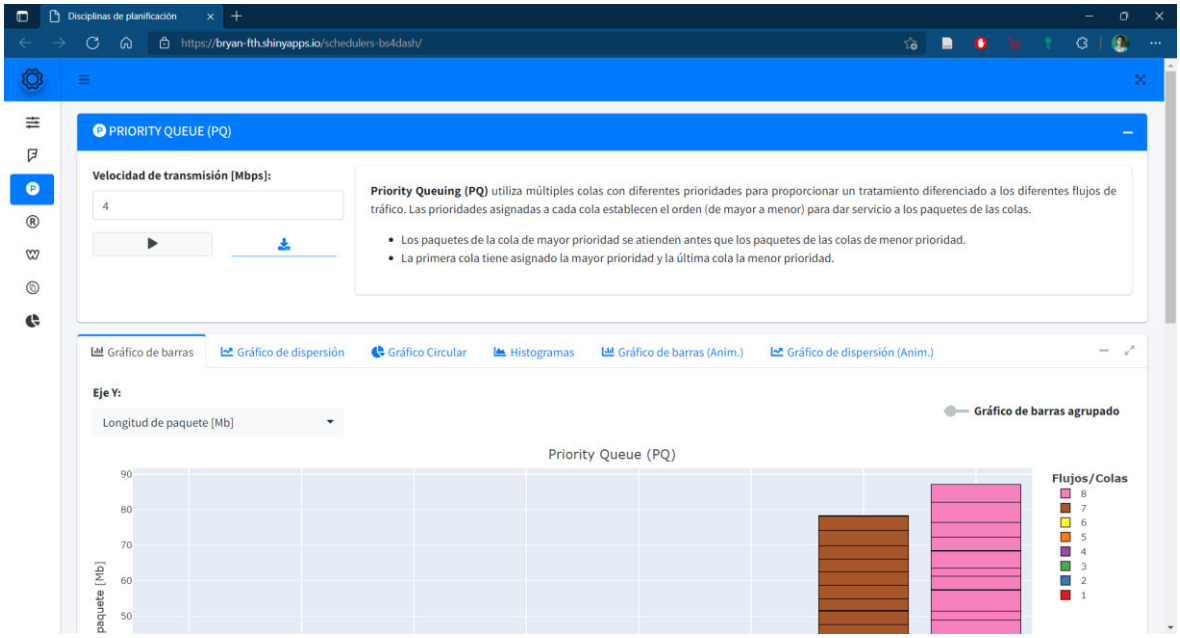


**Figura 3.19.** Ejecución del prototipo de aplicación web (Microsoft Edge)

## 3.5. PRUEBA DE VALIDACIÓN DE RESULTADOS

Para la simulación de las disciplinas de planificación se establecen 3 fuentes de paquetes, con una fuente para cada clase de servicio (3 flujos/colas). La tasa de transmisión del enlace de salida tiene un valor de 2 Mbps.

Para modelar la hora de llegada de los paquetes se utiliza las distribuciones de probabilidad de la Tabla 3.1.

**Tabla 3.1** Distribución de llegada de los paquetes

| Flujo/Cola | Distribución | Parámetros |
|---|---|---|
| 1 | Uniforme | mín. = 0.5 [s] |
| | | máx. = 1 [s] |
| 2 | Uniforme | mín. = 1 [s] |
| | | máx. = 2 [s] |
| 3 | Exponencial | rate = 3 [Mbps] |

Se considera 3 paquetes por cada flujo o cola, cada uno con diferente longitud. Para el modelado de las longitudes de los paquetes se utiliza las distribuciones de probabilidad de la Tabla 3.2.

**Tabla 3.2** Distribución de los paquetes (longitud de paquetes)

| Flujo/Cola | Distribución | Parámetros |
|---|---|---|
| 1 | Exponencial | rate = 4 [Mbps] |
| 2 | Uniforme | mín. = 1 [Mb] |
| | | máx. = 2 [Mb] |
| 3 | Normal | media = 0.8 [Mb] |
| | | sd = 0.2 [Mb] |

Para la disciplina de planificación WFQ, los pesos relacionados a cada clase son: $w_1 = 1$, $w_2 = 2$, $w_3 = 3$.

Para simplificar el proceso de validación de resultados, todas las disciplinas de planificación utilizarán los mismos datos.

### Configuración de parámetros

Como primer paso, se procede a establecer los parámetros de simulación, esta tarea es realizada por el usuario en la interfaz del módulo de configuración de parámetros. La Figura 3.20 muestra la pantalla principal de la aplicación, permitiendo al usuario establecer los diferentes parámetros de simulación. Para la validación de los resultados se utilizan los valores descritos al inicio de esta sección (validación de resultados).

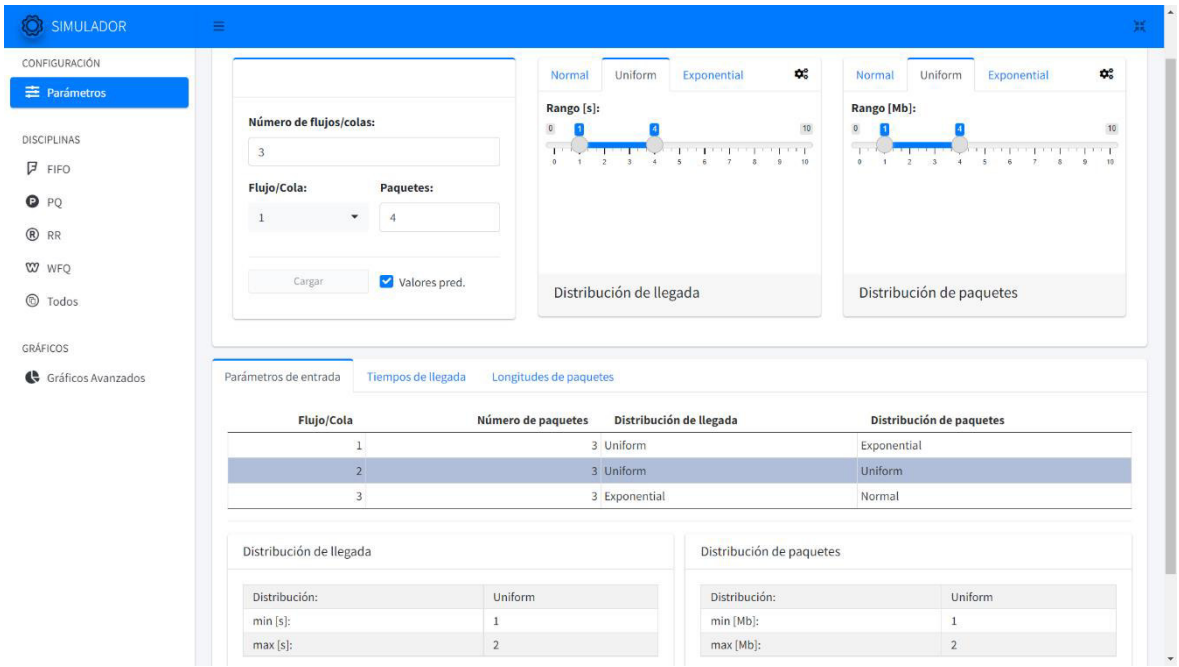


**Figura 3.20.** Configuración de parámetros

### 3.5.1. FIRST-IN FIRST-OUT

Una vez se ha establecido los parámetros de simulación, se procede a seleccionar la disciplina de planificación deseada y establecer los valores restantes, en este caso la velocidad de transmisión del enlace, el cual se establece en 2 [Mbps]. La Figura 3.21 muestra la interfaz gráfica de simulación para la disciplina de planificación FIFO.

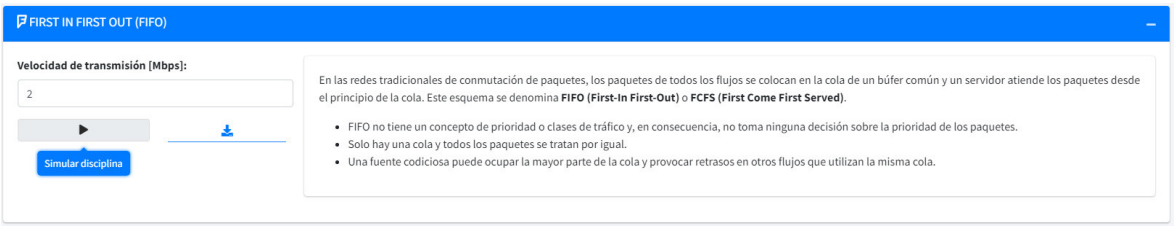


**Figura 3.21.** Configuración de la velocidad de transmisión del enlace (FIFO)

En la Figura 3.22 se observa un gráfico interactivo con los valores más importantes relacionadas a un paquete. La información que se visualiza en la siguiente figura corresponde al primer paquete en ser atendido, el cual pertenece al primer flujo (Flujo 1).

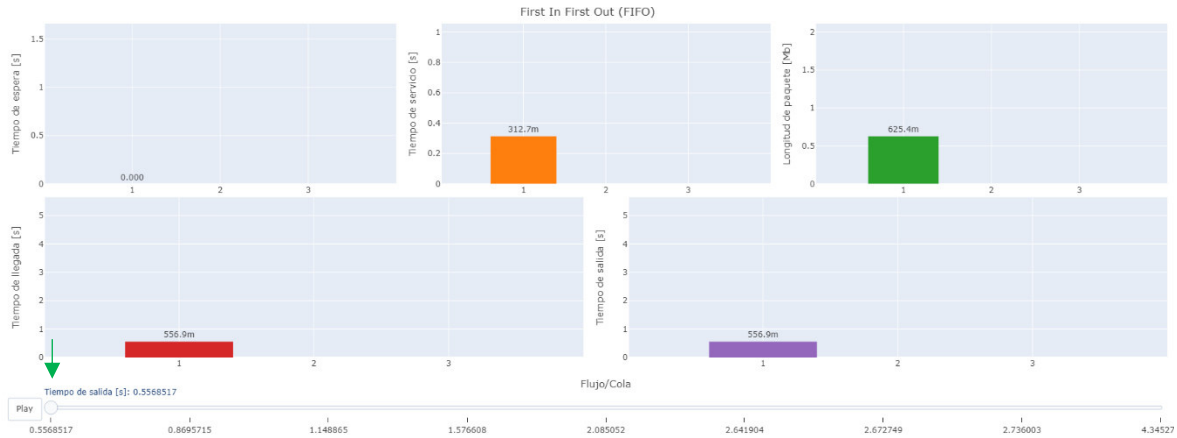


**Figura 3.22.** Simulación FIFO (Primer paquete)

Para observar la información del siguiente paquete, es necesario mover el deslizador a la siguiente posición. En la Figura 3.23 se puede observar el segundo paquete atendido y los valores de sus métricas.

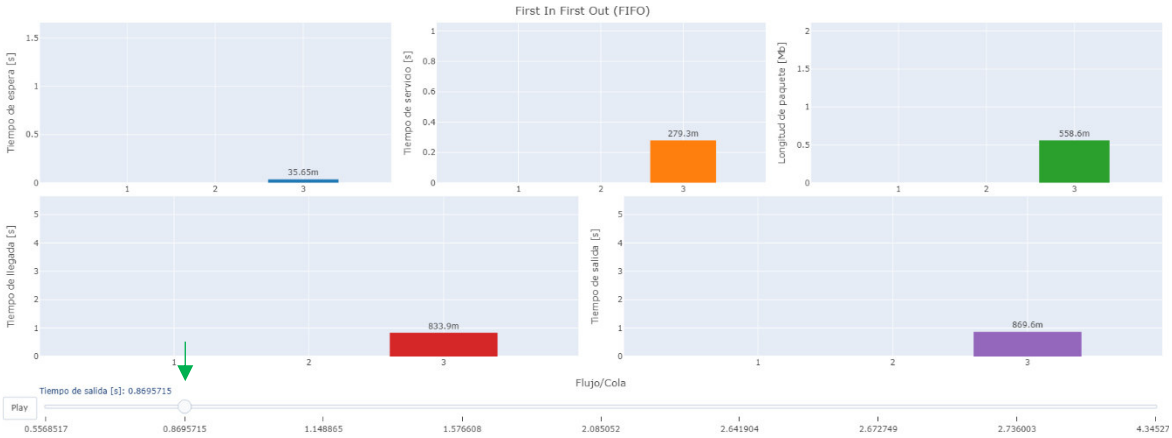


**Figura 3.23.** Simulación FIFO (Segundo paquete)

En la Figura 3.24 se observa una tabla de datos dinámica con un resumen de todos los valores generados en la simulación.

| Cola | Longitud de paquete [Mb] | Tiempo de llegada [s] | Tiempo de salida [s] | Tiempo de espera [s] | Tiempo de servicio [s] | Tiempo final [s] |
|---|---|---|---|---|---|---|
| 1 | 0.62544 | 0.55685 | 0.55685 | 0 | 0.31272 | 0.86957 |
| 3 | 0.55859 | 0.83392 | 0.86957 | 0.03565 | 0.27929 | 1.14886 |
| 3 | 0.85549 | 0.91617 | 1.14886 | 0.23269 | 0.42774 | 1.57661 |
| 3 | 1.01689 | 0.91837 | 1.57661 | 0.65824 | 0.50844 | 2.08505 |
| 2 | 1.1137 | 1.1137 | 2.08505 | 0.97135 | 0.55685 | 2.6419 |
| 1 | 0.06169 | 1.368 | 2.6419 | 1.2739 | 0.03084 | 2.67275 |
| 1 | 0.00165 | 2.17264 | 2.67275 | 0.50011 | 0.00082 | 2.67357 |
| 2 | 1.6223 | 2.736 | 2.736 | 0 | 0.81115 | 3.54715 |
| 2 | 1.60927 | 4.34528 | 4.34528 | 0 | 0.80464 | 5.14991 |

**Figura 3.24.** Datos de simulación de la disciplina FIFO

A continuación, se procede a verificar los resultados obtenidos en la simulación. Para lo cual, el primer paso consiste en generar los valores de los tiempos de llegada y las longitudes de los paquetes con ayuda de las funciones de distribución de R. Estas funciones generan valores pseudoaleatorios, así que, con el fin de reproducir los valores obtenidos en la simulación, se establece un mismo valor de semilla tanto en el simulador como en la nueva sesión de R que generará los valores antes mencionados.

La Tabla 3.3 muestra los valores correspondientes a los tiempos de llegada entre paquetes de la misma clase.

**Tabla 3.3.** Tiempos de llegada entre paquetes

| **Tiempo de llegada [s]** | | |
|---|---|---|
| **Flujo 1** | **Flujo 2** | **Flujo 3** |
| 0.55685 | 1.11370 | 0.83392 |
| 1.36800 | 2.73600 | 0.91617 |
| 2.17264 | 4.34528 | 0.91837 |

La Tabla 3.4 muestra los valores que corresponden a las longitudes de los paquetes de los diferentes flujos.

**Tabla 3.4.** Longitud de los paquetes

| Longitud [Mb] | | |
|---|---|---|
| **Flujo 1** | **Flujo 2** | **Flujo 3** |
| 0.62544 | 1.11370 | 0.55859 |
| 0.06169 | 1.62230 | 0.85549 |
| 0.00165 | 1.60927 | 1.01689 |

Por simplicidad y para facilitar la representación de cada paquete, se utiliza la notación $P_k(i)$; donde $i$ y $k$ representan el índice del paquete y el número de flujo/cola respectivamente. Por ejemplo, $P_1(3)$ hace referencia al tercer paquete del flujo número 1.

El menor tiempo de llegada corresponde a $P_1(1)$, por lo que es el primer paquete en ser atendido. Debido a que no hay paquetes previos, el tiempo de espera en la cola es de cero, y por lo tanto el tiempo de salida coincide con el tiempo de llegada. $P_1(1)$ tiene una longitud de $0.62544\ [Mb]$, por lo que el *router* se demora en procesar este paquete $0.62544\ [Mb]\ /\ 2\left[\frac{Mb}{s}\right] = 0.31272\ [s]$. En el instante $t = 0.55685\ [s] + 0.31272\ [s] = 0.86957\ [s]$ finaliza la ejecución de $P_1(1)$ y, por lo tanto, el paquete es removido de la cola.

El siguiente paquete es $P_3(1)$, el cuál llega en el instante $t = 0.83392$ [s], mientras el primer paquete aún sigue procesándose. $P_3(1)$ permanece en la cola $0.86957\ [s] - 0.83392\ [s] = 0.03565\ [s]$. Este paquete tiene una longitud de $0.55859\ [Mb]$, por lo que el *router* se demora en procesar este paquete $0.55859\ [Mb]/2\left[\frac{Mb}{s}\right] = 0.27929\ [s]$. En el instante $t = 0.86957\ [s] + 0.27929\ [s] = 1.14886\ [s]$ finaliza la ejecución de $P_3(3)$ y es removido de la cola.

Este proceso continúa hasta que se agoten los paquetes de todos los flujos. Al finalizar el proceso, se obtienen los resultados que se observan en la Tabla 3.5. Como se puede observar de los resultados anteriores, los valores calculados coinciden con los valores generados en la simulación de la disciplina FIFO (Figura 3.24).

**Tabla 3.5.** Datos calculados de la disciplina de planificación FIFO

| Cola | Tiempo de espera [s] | Tiempo de servicio [s] | Tiempo final [s] |
|---|---|---|---|
| 1 | 0.55685 - 0.55685 = 0 | 0.62544 / 2 = 0.31272 | 0.55685 + 0.31272 = 0.86957 |
| 3 | 0.86957 - 0.83392 = 0.03565 | 0.55859 / 2 = 0.27929 | 0.86957 + 0.27929 = 1.14886 |
| 3 | 1.14886 - 0.91617 = 0.23269 | 0.85549 / 2 = 0.42774 | 1.14886 + 0.42774 = 1.57661 |
| 3 | 1.57661 - 0.91837 = 0.65824 | 1.01689 / 2 = 0.50844 | 1.57661 + 0.50844 = 2.08505 |

| | | | |
|---|---|---|---|
| 2 | 2.08505 - 1.11370 = 0.97135 | 1.11370 / 2 = 0.55685 | 2.08505 + 0.55685 = 2.64190 |
| 1 | 2.64190 - 1.36800 = 1.27390 | 0.06169 / 2 = 0.03084 | 2.64190 + 0.03084 = 2.67275 |
| 1 | 2.67275 - 2.17264 = 0.50011 | 0.00165 / 2 = 0.00082 | 2.67275 + 0.00082 = 2.67357 |
| 2 | 2.73600 - 2.73600 = 0 | 1.62230 / 2 = 0.81115 | 2.73600 + 0.81115 = 3.54715 |
| 2 | 4.34528 - 4.34528 = 0 | 1.60927 / 2 = 0.80464 | 4.34528 + 0.80464 = 5.14991 |

### 3.5.2. PRIORITY QUEUEING

La Figura 3.25 ilustra la configuración de los parámetros necesarios para la simulación de la disciplina de planificación PQ.

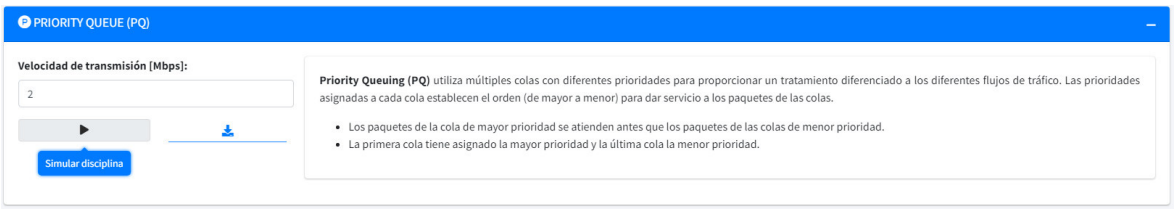


**Figura 3.25.** Configuración de la velocidad de transmisión del enlace (PQ)

En las Figuras 3.26 y 3.27 se observan los valores de la longitud, el tiempo de espera en la cola y el tiempo de servicio de los dos primeros paquetes en ser atendidos. En estas figuras se puede observar que el primer paquete en ser atendido pertenece a la cola 1 y el segundo paquete pertenece a la cola 3.

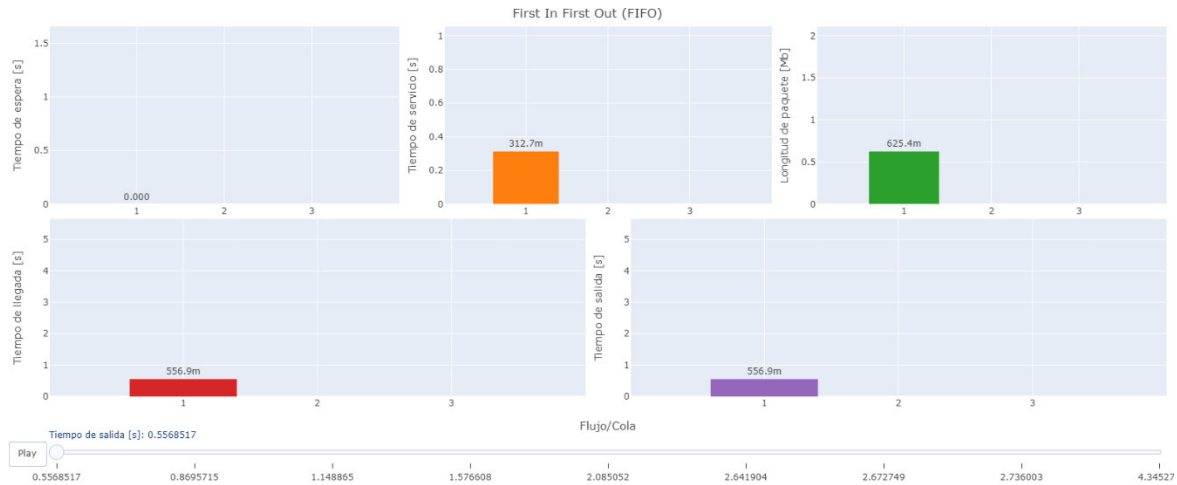


**Figura 3.26.** Simulación PQ (Primer paquete)

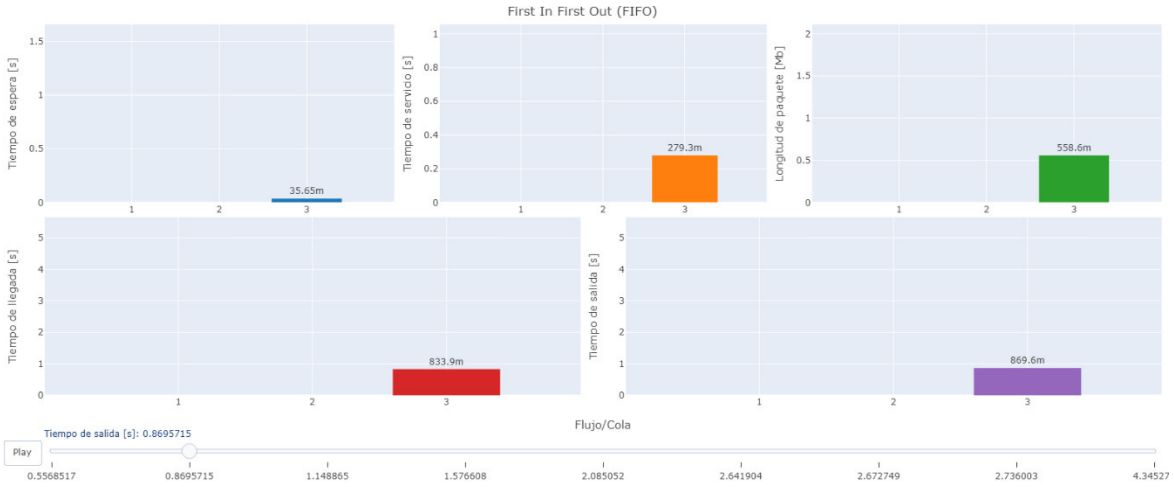


**Figura 3.27.** Simulación PQ (Segundo paquete)

En la Figura 3.28 se observan todos los valores de las métricas que se generaron durante la simulación. Los paquetes están ordenados de acuerdo con el orden en el que fueron atendidos.

| Cola | Longitud de paquete [Mb] | Tiempo de llegada [s] | Tiempo de salida [s] | Tiempo de espera [s] | Tiempo de servicio [s] | Tiempo final [s] |
|---|---|---|---|---|---|---|
| 1 | 0.62544 | 0.55685 | 0.55685 | 0 | 0.31272 | 0.86957 |
| 3 | 0.55859 | 0.83392 | 0.86957 | 0.03565 | 0.27929 | 1.14886 |
| 2 | 1.1137 | 1.1137 | 1.14886 | 0.03516 | 0.55685 | 1.70572 |
| 1 | 0.06169 | 1.368 | 1.70572 | 0.33772 | 0.03084 | 1.73656 |
| 3 | 0.85549 | 0.91617 | 1.73656 | 0.82039 | 0.42774 | 2.1643 |
| 3 | 1.01689 | 0.91837 | 2.1643 | 1.24594 | 0.50844 | 2.67275 |
| 1 | 0.00165 | 2.17264 | 2.67275 | 0.50011 | 0.00082 | 2.67357 |
| 2 | 1.6223 | 2.736 | 2.736 | 0 | 0.81115 | 3.54715 |
| 2 | 1.60927 | 4.34528 | 4.34528 | 0 | 0.80464 | 5.14991 |

**Figura 3.28.** Datos de simulación de la disciplina PQ

A continuación, se procede a verificar los resultados obtenidos en la simulación. Para este ejemplo se tiene tres colas, por lo que se puede aplicar una de las tres prioridades para una clase: alta, media y baja. La prioridad alta corresponde a la primera cola, la prioridad media corresponde a la segunda cola, y la prioridad baja corresponde a la tercera cola. Los paquetes de la cola de mayor prioridad se atienden antes que los paquetes de las colas de menor prioridad. Además, las colas de menor prioridad se atienden solo si todas las colas de mayor prioridad están vacías.

En el instante $t = 0.55685\ [s]$, llega a la cola 1 el paquete $P_1(1)$. La cola 1 tiene la mayor prioridad, por lo que $P_1(1)$ es atendido de forma inmediata. $P_1(1)$ tiene una longitud de 0.62544 [Mb], por lo que el *router* se demora en procesarlo $0.62544\ [Mb]/2\left[\frac{Mb}{s}\right] =$

$0.31272\ [s]$. En el instante $t = 0.55685\ [s] + 0.31272\ [s] = 0.86957\ [s]$ finaliza la ejecución de $P_1(1)$, y el paquete es removido de la cola.

En el instante en el que finaliza la ejecución de $P_1(1)$, no hay paquetes disponibles en las colas 1 y 2, por lo que el planificador continúa con la siguiente cola de menor prioridad (cola 3). $P_3(1)$ llegó a la cola en el instante $t = 0.83392\ [s]$. $P_3(1)$ es atendido en el instante $t = 0.86957\ [s]$, por lo que el tiempo de espera en la cola es de $t = 0.86957\ [s] - 0.83392\ [s] = 0.03565\ [s]$. $P_3(1)$ tiene una longitud de $0.55859\ [Mb]$, por lo que el *router* se demora en procesarlo $0.55859\ [Mb]/2\left[\frac{Mb}{s}\right] = 0.27929\ [s]$. En el instante $t = 0.83392\ [s] + 0.27929\ [s] = 1.14886\ [s]$, $P_3(1)$ finaliza su ejecución y es removido de la cola.

En el instante $t = 1.1137\ [s]$ llega el paquete $P_2(1)$ a la cola de prioridad media (cola 2), por lo que el planificador pasa de la cola 3 a la cola 2 (aún no hay paquetes en la cola 1). $P_2(1)$ es atendido en el instante $t = 1.14886\ [s]$, por lo que espera en la cola $t = 1.14886\ [s] - 1.1137\ [s] = 0.55685\ [s]$. $P_2(1)$ tiene una longitud de $1.1137\ [Mb]$, por lo que el *router* se demora en procesarlo $1.1137\ [Mb]/2\left[\frac{Mb}{s}\right] = 0.55685\ [s]$. En el instante $t = 1.14886\ [s] + 0.55685\ [s] = 1.70572\ [s]$, $P_2(1)$ finaliza su ejecución y es removido de la cola.

El proceso anterior continúa hasta que se agoten los paquetes disponibles de todas las colas. Al finalizar este proceso, se obtienen los resultados que se observan en la Tabla 3.6. Como se puede observar, los valores calculados coinciden con los valores generados en la simulación de la disciplina PQ (Figura 3.28).

**Tabla 3.6.** Datos calculados de la disciplina de planificación PQ

| Cola | Tiempo de espera [s] | Tiempo de servicio [s] | Tempo final [s] |
|---|---|---|---|
| 1 | 0.55685 - 0.55685 = 0 | 0.62544 / 2 = 0.31272 | 0.55685 + 0 = 0.55685 |
| 3 | 0.83392 - 0.86957 = 0.03565 | 0.55859 / 2 = 0.27929 | 0.86957 + 0.27929 = 1.14886 |
| 2 | 1.11370 - 1.14886 = 0.03516 | 1.1137 / 2 = 0.55685 | 1.14886 + 0.42774 = 1.70572 |
| 1 | 1.36800 - 1.70572 = 0.33772 | 0.06169 / 2 = 0.03084 | 1.70572 + 0.50844 = 1.73656 |
| 3 | 0.91617 - 1.73656 = 0.82039 | 0.85549 / 2 = 0.42774 | 1.73656 + 0.55685 = 2.1643 |
| 3 | 0.91837 - 2.16430 = 1.24594 | 1.01689 / 2 = 0.50844 | 2.16430 + 0.03084 = 2.67275 |
| 1 | 2.17264 - 2.67275 = 0.50011 | 0.00165 / 2 = 0.00082 | 2.67275 + 0.00082 = 2.67357 |
| 2 | 2.73600 - 2.73600 = 0 | 1.6223 / 2 = 0.81115 | 2.76300 + 0.81115 = 3.54715 |
| 2 | 4.34528 - 4.34528 = 0 | 1.60927 / 2 = 0.80464 | 4.34528 + 0.80464 = 5.14991 |

### 3.5.3. ROUND ROBIN

La Figura 3.29 ilustra la configuración de los parámetros necesarios para la simulación de la disciplina de planificación RR.

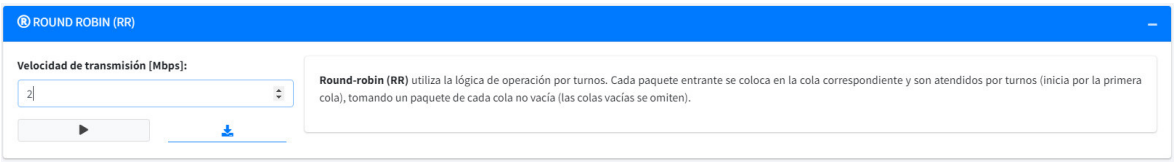


**Figura 3.29.** Configuración de la velocidad de transmisión del enlace (RR)

En las Figuras 3.30 y 3.31 se observan los valores de las métricas relacionadas con la longitud, el tiempo de espera en la cola y el tiempo de servicio de los paquetes. En estas figuras se puede observar que el primer paquete en ser atendido pertenece a la cola 1, y el segundo paquete pertenece a la cola 3.

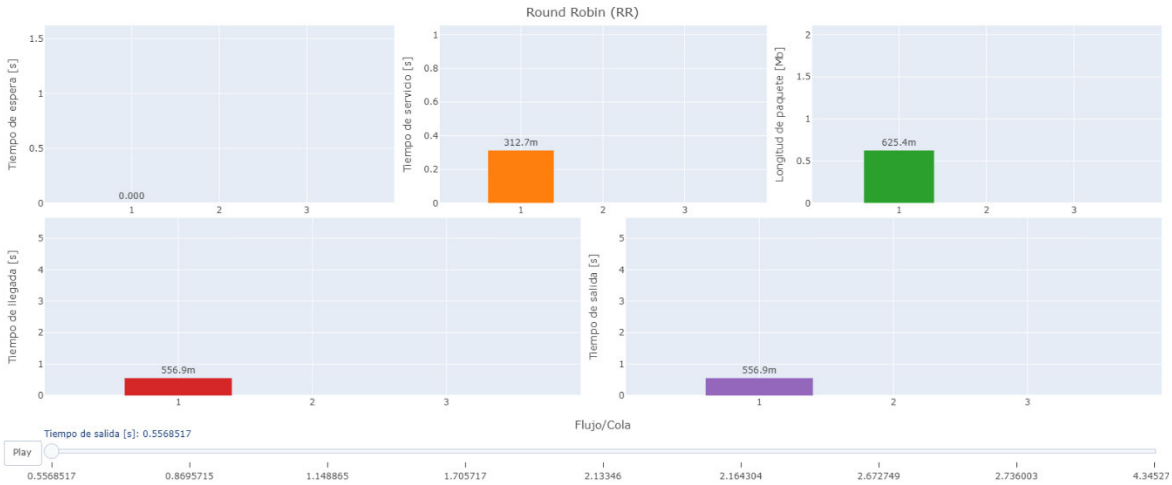


**Figura 3.30.** Simulación RR (Primer paquete)

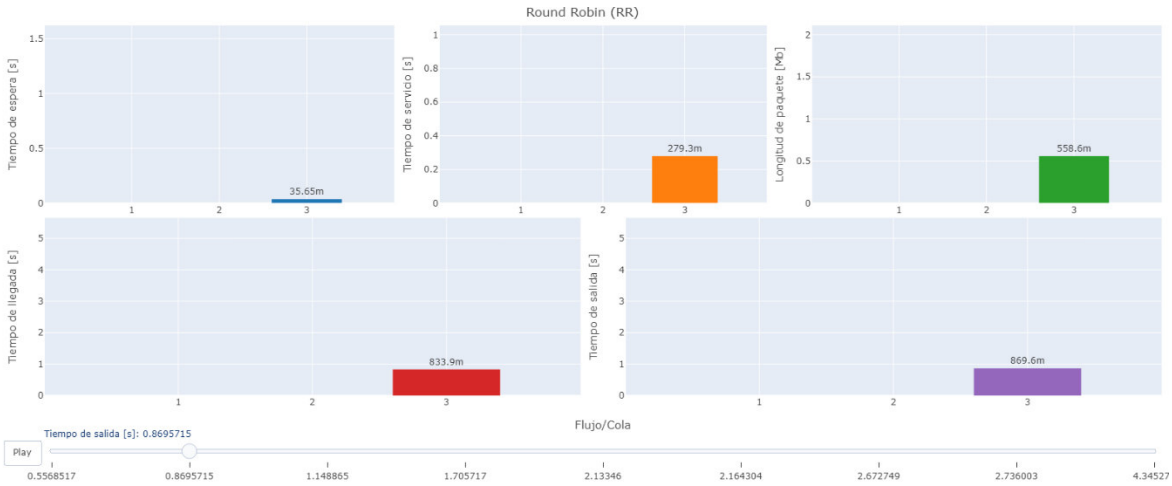


**Figura 3.31.** Simulación RR (Segundo paquete)

En la Figura 3.32 se observan todos los valores de las métricas que se generaron durante la simulación. En esta tabla, los paquetes están ubicados de acuerdo con el orden en el que fueron atendidos.

| Cola | Longitud de paquete [Mb] | Tiempo de llegada [s] | Tiempo de salida [s] | Tiempo de espera [s] | Tiempo de servicio [s] | Tiempo final [s] |
|---|---|---|---|---|---|---|
| 1 | 0.62544 | 0.55685 | 0.55685 | 0 | 0.31272 | 0.86957 |
| 3 | 0.55859 | 0.83392 | 0.86957 | 0.03565 | 0.27929 | 1.14886 |
| 2 | 1.1137 | 1.1137 | 1.14886 | 0.03516 | 0.55685 | 1.70572 |
| 3 | 0.85549 | 0.91617 | 1.70572 | 0.78954 | 0.42774 | 2.13346 |
| 1 | 0.06169 | 1.368 | 2.13346 | 0.76546 | 0.03084 | 2.1643 |
| 3 | 1.01689 | 0.91837 | 2.1643 | 1.24594 | 0.50844 | 2.67275 |
| 1 | 0.00165 | 2.17264 | 2.67275 | 0.50011 | 0.00082 | 2.67357 |
| 2 | 1.6223 | 2.736 | 2.736 | 0 | 0.81115 | 3.54715 |
| 2 | 1.60927 | 4.34528 | 4.34528 | 0 | 0.80464 | 5.14991 |

**Figura 3.32.** Datos de simulación de la disciplina RR

A continuación, se procede a verificar los resultados obtenidos en la simulación. RR utiliza la lógica de operación por turnos, por lo que todas las colas son atendidas periódicamente. Cada paquete entrante se coloca en la cola correspondiente y son atendidos por turnos, tomando un paquete de cada cola no vacía.

En el instante $t = 0.55685\ [s]$, llega a la cola 1 el paquete $P_1(1)$, el cuál es atendido de forma inmediata. $P_1(1)$ tiene una longitud de $0.62544\ [Mb]$, por lo que el *router* se demora en procesarlo $0.62544\ [Mb]/2\left[\frac{Mb}{s}\right] = 0.31272\ [s]$. En el instante $t = 0.55685\ [s] + 0.31272\ [s] = 0.86957\ [s]$ finaliza la ejecución de $P_1(1)$ y es removido de la cola. El planificador continúa con la siguiente cola.

En el instante de tiempo $0.86957\ [s]$, aún no hay paquetes disponibles en la cola 2, por lo que el planificador continúa con la cola 3. $P_3(1)$ llegó a la cola en el instante $t = 0.83392\ [s]$. $P_3(1)$ es atendido en el instante $t = 0.86957\ [s]$, por lo que el tiempo de espera en la cola es de $t = 0.86957\ [s] - 0.83392\ [s] = 0.03565\ [s]$. $P_3(1)$ tiene una longitud de $0.55859\ [Mb]$, por lo que el *router* se demora en procesarlo $0.55859\ [Mb]/2\left[\frac{Mb}{s}\right] = 0.27929\ [s]$. En el instante $t = 0.83392\ [s] + 0.27929\ [s] = 1.14886\ [s]$ finaliza la ejecución de $P_3(1)$ y es removido de la cola. En este instante de tiempo, el planificador completó el primero ciclo, por lo que empieza nuevamente con la cola 1. Sin embargo, en esta cola no hay paquetes por lo que el planificador continúa con la siguiente cola.

En la cola 2, $P_2(1)$ llegó en el instante $t = 1.1137\ [s]$. Este paquete es atendido en el instante $t = 1.14886\ [s]$, por lo que espera en la cola $t = 1.14886\ [s] - 1.1137\ [s] = 0.55685\ [s]$. $P_2(1)$ tiene una longitud de $1.1137\ [Mb]$, por lo que el *router* se demora en

procesarlo $1.1137\ [Mb]/2\left[\frac{Mb}{s}\right] = 0.55685\ [s]$. En el instante $t = 1.14886\ [s] + 0.55685\ [s] = 1.70572\ [s]$ finaliza la ejecución de $P_2(1)$, y es removido de la cola.

El ciclo se repite hasta que se agoten los paquetes de todas las colas. Al finalizar el proceso, se obtienen los resultados que se observan en la Tabla 3.7. Como se puede observar, los valores calculados coinciden con los valores generados en la simulación de la disciplina RR (Figura 3.32).

**Tabla 3.7.** Datos calculados de la disciplina de planificación RR

| Cola | Tiempo de espera [s] | Tiempo de servicio [s] | Tiempo final [s] |
|---|---|---|---|
| 1 | 0.55685 - 0.55685 = 0 | 0.62544 / 2 = 0.31272 | 0.55685 + 0.31272 = 0.86957 |
| 3 | 0.86957 - 0.83392 = 0.03565 | 0.55859 / 2 = 0.27929 | 0.86957 + 0.27929 = 1.14886 |
| 2 | 1.14886 - 1.11370 = 0.03516 | 1.11370 / 2 = 0.55685 | 1.14886 + 0.55685 = 1.70572 |
| 3 | 1.70572 - 0.91617 = 0.78954 | 0.85549 / 2 = 0.42774 | 1.70572 + 0.42774 = 2.13346 |
| 1 | 2.13346 - 1.36800 = 0.76546 | 0.06169 / 2 = 0.03084 | 2.13346 + 0.03084 = 2.16430 |
| 3 | 2.16430 - 0.91837 = 1.24594 | 1.01689 / 2 = 0.50844 | 2.16430 + 0.50844 = 2.67275 |
| 1 | 2.67275 - 2.17264 = 0.50011 | 0.00165 / 2 = 0.00082 | 2.67275 + 0.00082 = 2.67357 |
| 2 | 2.73600 - 2.73600 = 0 | 1.62230 / 2 = 0.81115 | 2.73600 + 0.81115 = 3.54715 |
| 2 | 4.34528 - 4.34528 = 0 | 1.60927 / 2 = 0.80464 | 4.34528 + 0.80464 = 5.14991 |

### 3.5.4. WEIGHTED FAIR QUEUEING

La Figuras 3.33 ilustra la configuración de los parámetros necesarios para la simulación de la disciplina de planificación WFQ.

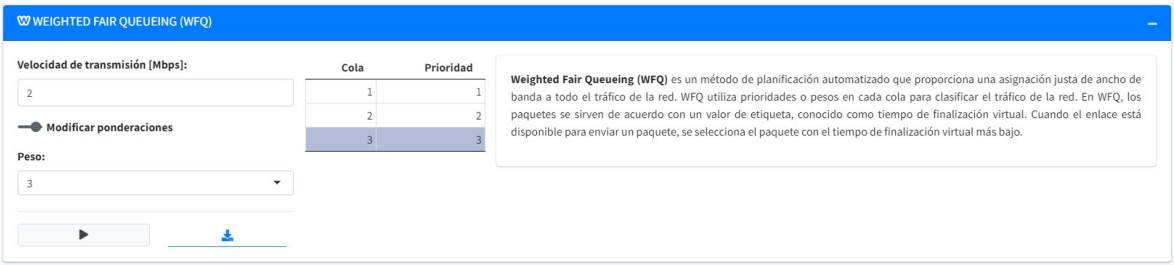


**Figura 3.33.** Configuración de la velocidad de transmisión del enlace y los pesos asociados a cada cola.

En las Figuras 3.34 y 3.35 se observan los valores de las métricas relacionadas con los paquetes.

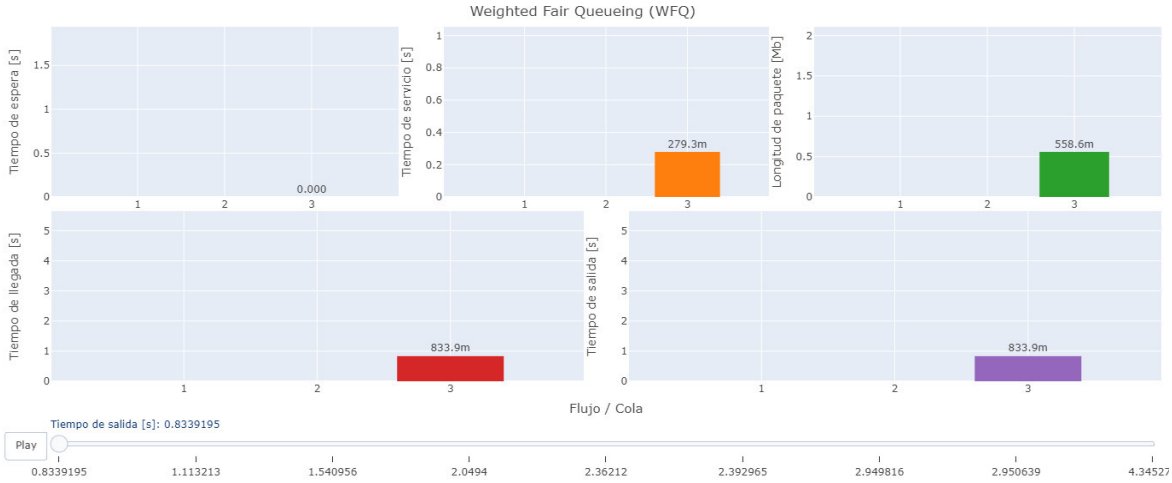


**Figura 3.34.** Simulación WFQ (Primer paquete)

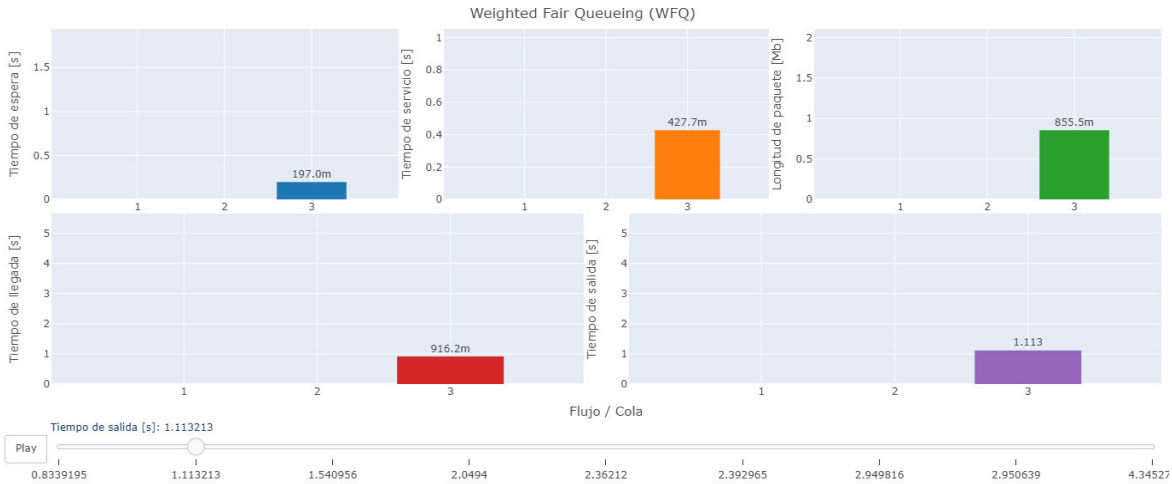


**Figura 3.35.** Simulación WFQ (Segundo paquete)

De acuerdo con la Figura 3.34, el primer paquete en ser atendido pertenece a la cola 3. Este paquete tiene una longitud de 0.55859 [Mb], con un tiempo de servicio o procesamiento de 0.27929 [s]. El segundo paquete en ser atendido pertenece igualmente a la cola 3 (Figura 3.35). En la Figura 3.36 un resumen de todos los valores generados en la simulación de la disciplina WFQ.

| Cola | Longitud de paquete [Mb] | Tiempo de llegada [s] | Tiempo de salida [s] | Tiempo de espera [s] | Tiempo de servicio [s] | Tiempo final [s] |
|---|---|---|---|---|---|---|
| 3 | 0.55859 | 0.83392 | 0.83392 | 0 | 0.27929 | 1.11321 |
| 3 | 0.85549 | 0.91617 | 1.11321 | 0.19704 | 0.42774 | 1.54096 |
| 1 | 0.62544 | 0.55685 | 1.54096 | 0.9841 | 0.31272 | 1.85368 |
| 1 | 0.06169 | 1.368 | 1.85368 | 0.48567 | 0.03084 | 1.88452 |
| 1 | 0.00165 | 2.17264 | 2.17264 | 0 | 0.00082 | 2.17346 |
| 2 | 1.1137 | 1.1137 | 2.17346 | 1.05976 | 0.55685 | 2.73031 |
| 3 | 1.01689 | 0.91837 | 2.73031 | 1.81195 | 0.50844 | 3.23876 |
| 2 | 1.6223 | 2.736 | 3.23876 | 0.50275 | 0.81115 | 4.04991 |
| 2 | 1.60927 | 4.34528 | 4.34528 | 0 | 0.80464 | 5.14991 |

**Figura 3.36.** Datos de simulación de la disciplina WFQ

A continuación, se procede a verificar los resultados obtenidos en la simulación. En WFQ, las colas tienen un valor de peso/prioridad ($w_k$) asignado a cada cola. El flujo de cada cola (cola $k$) alcanza una tasa de datos promedio de acuerdo con la Ecuación 1.1.

$$r_1 = \frac{1}{1+2+3} \cdot 2\ [Mbps] = \frac{1}{3}\ [Mbps]$$

$$r_2 = \frac{2}{1+2+3} \cdot 2\ [Mbps] = \frac{2}{3}\ [Mbps]$$

$$r_3 = \frac{3}{1+2+3} \cdot 2\ [Mbps] = 1\ [Mbps]$$

A continuación, para calcular el tiempo de finalización virtual de cada paquete se utiliza la Ecuación 1.2.

$$F_1(1) = \max(0, 0.55685) + \frac{0.62544\ [Mb]}{\frac{1}{3}\left[\frac{Mb}{s}\right]} = 2.43317\ [s]$$

$$F_1(2) = \max(2.43317, 1.36800) + \frac{0.06169\ [Mb]}{\frac{1}{3}\left[\frac{Mb}{s}\right]} = 2.61824\ [s]$$

$$F_1(3) = \max(2.61824, 2.17264) + \frac{0.00165\ [Mb]}{\frac{1}{3}\left[\frac{Mb}{s}\right]} = 2.62319\ [s]$$

Los valores calculados corresponden a todos los paquetes de la cola 1. La Tabla 3.8 muestra los tiempos de finalización virtual para todos los paquetes.

**Tabla 3.8.** Tiempos de finalización virtual

| Cola 1 | Cola 2 | Cola 3 |
|---|---|---|
| 2.43317 | 2.78426 | 1.39251 |
| 2.61824 | 5.21771 | 2.24799 |
| 2.62319 | 7.63162 | 3.26488 |

Para determinar el paquete a ser atendido se utiliza la Ecuación 1.3.

$$salida = \min\{F_k\} = \min\{2.43317, 2.78426, 1.39251\} = 1.39251$$

El paquete seleccionado para la salida es $P_3(1)$. Una vez finalizado el proceso, el paquete se elimina de la cola. Para el siguiente paquete se tiene:

$$salida = \min\{F_k\} = \min\{2.43317, 2.78426, 2.24799\} = 2.24799$$

El paquete seleccionado para la salida es $P_3(2)$. Una vez finalizado el proceso, el paquete se elimina de la cola. Este proceso se repite indefinidamente hasta que ya no queden paquetes en las colas. Como se puede observar en la Tabla 3.9, el orden en el que los paquetes son atendidos coincide con el orden de la Figura 3.36.

**Tabla 3.9.** Datos de la disciplina WFQ

| Cola | Tiempo de finalización virtual [s] |
|---|---|
| 3 | 1.39251 |
| 3 | 2.24799 |
| 1 | 2.43317 |
| 1 | 2.61824 |
| 1 | 2.62319 |
| 2 | 2.78426 |
| 3 | 3.26488 |
| 2 | 5.21771 |
| 2 | 7.63162 |

Los valores calculados coinciden con los valores generados por el prototipo (Figura 3.26).

## 3.6. CORRECCIÓN DE ERRORES

En esta sección se expone un resumen de los errores encontrados en el prototipo durante la fase de implementación y pruebas. En Tabla 3.10 se observan los errores que se encontraron y las soluciones.

**Tabla 3.10.** Corrección de errores

| Módulo | Error | Solución |
|---|---|---|
| Configuración de parámetros | No se presenta la tabla dinámica con el resumen de los parámetros. | *Shiny* dispone de funciones propias para la creación de tablas, sin embargo, estas no son compatibles con el paquete bs4Dash, por lo que se decidió utilizar el paquete DT. |
| Disciplinas de planificación | No se generan los reportes dinámicos. | Se instala el paquete knitr. |
| | No se visualizan las gráficas interactivas. | Se modifica el identificador del widget de salida para que coincida en el *front-end* y en el *back-end*. |
| Gráficos avanzados | Los *widgets* del módulo Esquisse no se visualizan de forma correcta. | Los widgets de Esquisse implementan las clases `pull-right` y `pull-left` (Bootstrap 3) que no están disponibles en Bootstrap 4. Para solucionar el error la clase `pull-right` se |

| | | |
|---|---|---|
| | | cambió por la clase `float-right` y la clase `pull-left` por la clase `float-left`. |
| | Las pestañas del módulo datamods (importar datos) no se visualizan correctamente. | El paquete datamods utiliza la función `tabsetPanel` de *Shiny* para implementar las pestañas. Esta función utiliza varias clases de la biblioteca Bootstrap 3 que no están disponibles en Bootstrap 4. El problema se corrigió al cambiar la función `shiny::tabsetPanel` por `bs4Dash::tabsetPanel`. |

*Shiny* no genera un error por escribir mal el nombre de una variable o de una función, por lo que para identificar los errores se utilizó del depurador de RStudio y del paquete reactlog para generar un gráfico reactivo. En el ANEXO I se adjunta el gráfico reactivo generado con el paquete reactlog.

## 3.7. TABLERO KANBAN FINAL

Finalmente, todas las actividades listadas en el tablero Kanban (Tabla 2.1) han culminado con éxito, por lo que todas las actividades están ubicadas en la columna “Terminado”.

# 4. CONCLUSIONES Y RECOMENDACIONES

## 4.1. CONCLUSIONES

- Al finalizar el desarrollo de este Proyecto de Titulación se consiguió desarrollar un prototipo de aplicación web interactiva para la simulación de las principales disciplinas de planificación de paquetes para la gestión de QoS. Este prototipo sirve como material de apoyo a los estudiantes para reforzar los conceptos sobre las disciplinas de planificación de paquetes, y a los maestros como un complemento de enseñanza a los métodos tradicionales.

- Para el desarrollo de este Proyecto de Titulación se adquirieron nuevos conocimientos sobre el análisis, procesamiento y visualización de los datos con el lenguaje de programación R, la creación de aplicaciones interactivas con el *framework Shiny*, y el despliegue de aplicaciones *Shiny*. Además, se repasaron conceptos fundamentales sobre la Calidad de Servicios (QoS) y disciplinas de planificación de paquetes.

- Para el desarrollo de este Proyecto de Titulación se utilizó la metodología ágil Kanban debido a su gran flexibilidad. La metodología ágil Kanban permitió desarrollar todas las tareas o actividades en orden y de forma efectiva.

- Las entrevistas realizadas a los docentes de la Facultad de Ingeniería Eléctrica y Electrónica de la Escuela Politécnica Nacional permitieron establecer tanto los requerimientos funcionales como los no funcionales. Con base en estos requerimientos se definieron los diferentes componentes de los módulos del prototipo y su funcionalidad.

- Se escogió el lenguaje de programación R para el desarrollo de este Proyecto de Titulación debido a su gran eficacia en el análisis de los datos. Además, R cuenta con el *framework Shiny*, el cual permite crear aplicaciones web interactivas con apariencia profesional sin conocimientos previos de desarrollo web.

- Se eligió shinyapps.io para el almacenamiento del prototipo en la nube ya que facilita el proceso de despliegue de aplicaciones *Shiny* en la web. Además, durante el despliegue de la aplicación el paquete `rsconnect` envía la versión de R, la lista de sus dependencias y la aplicación a shinyapps.io. Luego shinyapps.io compila e

instala todos los paquetes, por lo que no es necesario una configuración manual del servidor.

- La implementación de los planificadores FIFO, PQ y RR es simple, sin embargo, presentan ciertas limitaciones cuando se enfrentan a fuentes codiciosas o cuando se los utiliza en una red congestionada. Por otro lado, la implementación de WFQ es más compleja, pero este algoritmo gestiona de mejor manera los recursos de la red (ancho de banda) y soluciona las limitaciones de los algoritmos anteriores.

- Los resultados de los distintos escenarios simulados permitieron verificar varios conceptos de las disciplinas de planificación. En la simulación del planificador FIFO, se pudo observar que los tiempos de espera de los paquetes en la cola dependen de su orden de llegada. Además, los flujos con paquetes de gran tamaño consumen la mayor parte de los recursos de la red, lo que provoca un retraso a los otros flujos que utilizan la misma cola. Por otro lado, en la simulación del planificador PQ se pudo observar que la asignación de recursos no está garantizada para las colas de menor prioridad, ya que el planificador atenderá estas colas únicamente si las colas de mayor prioridad están vacías.

  En RR, se pudo observar que el planificador asigna los recursos en orden y, de este modo, los paquetes de todas las colas son atendidos periódicamente, solucionando así el problema de inanición. En cuanto al planificador WFQ, se pudo comprobar que distribuye los recursos de forma justa y precisa entre todas las colas disponibles. Sin embargo, debido a la complejidad de su algoritmo, WFQ (estándar) no es eficiente en redes de alta velocidad.

- Las pruebas de validación de los requerimientos realizadas al prototipo permitieron identificar y corregir los problemas de rendimiento y compatibilidad de los módulos del prototipo y sus elementos. Además, estas pruebas permitieron corroborar el cumplimiento de todos los requerimientos funcionales y no funcionales establecidos. Adicionalmente, las pruebas de validación de resultados nos permitieron verificar que los valores obtenidos en la simulación son los correctos y corresponden a cada uno de los planificadores implementados.

## 4.2. RECOMENDACIONES

- Se recomienda instalar una versión estable de R. Por lo general, shinyapps.io es compatible con la última versión de R después de tres días de su lanzamiento, sin

embargo, no todas sus características funcionan correctamente con la última versión disponible. Para el desarrollo de este Proyecto de Titulación se utilizó la versión 4.1.0 de R.

- RStudio actualiza los paquetes a la última versión compatible con el entorno R instalado, por lo que antes de iniciar un nuevo proyecto se recomienda actualizar los paquetes R, debido a que las versiones más recientes contienen actualizaciones de dependencias, adición de nuevas funcionalidades y mejoras en el rendimiento de las funciones integradas.
- Se recomienda el uso del servicio de almacenamiento shinyapps.io para aplicaciones *Shiny* ya que permite el despliegue de la aplicación sin la necesidad de configurar el servidor.
- Se recomienda seleccionar una metodología de desarrollo de software de acuerdo con las necesidades del proyecto a desarrollar debido que la metodología seleccionada será el punto de partida para la gestión del proyecto.
- Se recomienda la expansión del prototipo por medio de la adición de módulos y nuevas funcionalidades. Por ejemplo, la adición de nuevos módulos de disciplinas de planificación como *Class-Based Weighted Fair Queuing* (CBWFQ) o *Low-Latency Queuing* (LLQ).

# 5. REFERENCIAS BIBLIOGRÁFICAS


[1] S. Jha and M. Hassan, *Engineering Internet QoS*, 1st ed. Norwood, MA: Artech House, 2002.

[2] Y. Lin and J. Min, “Integrating Popular Web Applications in Classroom Learning Environments and Its Effects on Teaching, Student Learning Motivation and Performance,” *TOJECT*, vol. 12, no. 2, pp. 157–165, Apr. 2013.

[3] T. Szigeti, C. Hattingh, R. Barton, and K. Briley, *End-to-End QoS Network Design*, 2nd ed. Indianapolis: Cisco Press, 2013.

[4] ITU-T Study Group 12, “Definitions of terms related to quality of service,” *ITU-T Recommendations*, Sep. 2008. http://handle.itu.int/11.1002/1000/9524 (accessed Aug. 05, 2021).

[5] T. Braun, M. Diaz, J. Enríquez, and T. Staub, *End-to-End Quality of Service Over Heterogeneous Networks*, 1st ed. Heidelberg: Springer, 2008.

[6] S. Harpreet, *Implementing Cisco Networking Solutions*, 1st ed. Birmingham: Packt Publishing, 2017.

[7] A. Tanenbaum and D. Wetherall, *Redes de Computadoras*, 5th ed. México: Pearson Educación de México, 2016.

[8] A. Johnson, *31 Days Before Your CCNA Exam: A Day-By-Day Review Guide for the CCNA 200-301 Certification Exam*, 1st ed. Sydney: Cisco Press, 2020.

[9] M. Barreiros and P. Lundqvist, *QoS Enabled Networks - Tools and Foundations*, 2nd ed. United Kingdom: Wiley, 2016.

[10] S. Aidarous and T. Plevyak, *Managing IP Networks: Challenges and Opportunities*, 1st ed. 111 River Street, Hoboken: Wiley, 2003.

[11] C. Semeria, “Supporting Differentiated Service Classes: Queue Scheduling Disciplines,” *Juniper Networks, Inc.*, p. 27, 2001.

[12] O. Wendell, *CCNA 200-301 Official Cert Guide*, 1st ed., vol. 2. Sydney: Cisco Press, 2019.

[13] B. Mohamed, “QoS Concepts,” presented at the Enterprise Networking, Security, and Automation, Networking Academy. Accessed: Aug. 08, 2021. [Online]. Available: https://www.c02.co/apps/dnd/files/CCNA3/Instructor%20PowerPoints/

[14] S. Taniguchi, R. Kawate, K. Sato, E. Horiuchi, and T. Yokotani, “Performance evaluation of the simplified WFQ to multiplex a huge number of queues,” in *2012 IEEE International Workshop Technical Committee on Communications Quality and Reliability (CQR)*, May 2012, pp. 1–6. doi: 10.1109/CQR.2012.6267099.

[15] Atlassian, “Kanban: una breve introducción,” *Atlassian*. https://www.atlassian.com/es/agile/kanban (accessed Aug. 05, 2021).

[16] “Kanban Essentials Professional Certificate - KEPC | CertiProf.” https://certiprof.com/pages/kanban-essentials-professional-certificate-kepc (accessed Aug. 05, 2021).

[17] A. Stellman and J. Greene, *Learning Agile: Understanding Scrum, XP, Lean, and Kanban*, 1st ed. 1005 Gravenstein Highway North: O'Reilly Media, 2014.

[18] M. O. Ahmad, J. Markkula, and M. Oivo, "Kanban in software development: A systematic literature review," in *2013 39th Euromicro Conference on Software Engineering and Advanced Applications*, Santander, Sep. 2013, pp. 9–16. doi: 10.1109/SEAA.2013.28.

[19] M. O. Ahmad, D. Dennehy, K. Conboy, and M. Oivo, "Kanban in software engineering: A systematic mapping study," *Journal of Systems and Software*, vol. 137, pp. 96–113, Mar. 2018, doi: 10.1016/j.jss.2017.11.045.

[20] D. Bailey, "How to Get Started with Kanban in Software Development | Kanban Library." https://kanbantool.com/kanban-library/devops-kanban-basics/how-to-get-started-with-kanban-in-software-development (accessed Aug. 12, 2021).

[21] "R: What is R?" https://www.r-project.org/about.html (accessed Aug. 05, 2021).

[22] "Plotly R Graphing Library." https://plotly.com/r/ (accessed Dec. 04, 2020).

[23] "esquisse package - RDocumentation." https://www.rdocumentation.org/packages/esquisse/versions/1.0.2 (accessed Aug. 06, 2021).

[24] "RStudio | Open source & professional software for data science teams." https://rstudio.com/ (accessed Aug. 05, 2021).

[25] "RStudio." https://rstudio.com/products/rstudio/ (accessed Aug. 28, 2021).

[26] H. Wickham, *Mastering Shiny: Build Interactive Apps, Reports, and Dashboards Powered by R*, 1st ed. Sebastopol: O'Reilly Media, 2021.

[27] "Shiny - Function reference." https://shiny.rstudio.com/reference/shiny/1.6.0/ (accessed Sep. 09, 2021).

[28] "HTML basics - Learn web development | MDN." https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/HTML_basics (accessed Aug. 12, 2021).

[29] "CSS - Aprende sobre desarrollo web | MDN." https://developer.mozilla.org/es/docs/Learn/CSS (accessed Aug. 12, 2021).

[30] "¿Qué es JavaScript? - Aprende sobre desarrollo web | MDN." https://developer.mozilla.org/es/docs/Learn/JavaScript/First_steps/What_is_JavaScript (accessed Aug. 12, 2021).

[31] shinyapps io team, *shinyapps.io user guide*, 1st ed. 2020. Accessed: Nov. 30, 2020. [Online]. Available: https://docs.rstudio.com/shinyapps.io/index.html

[32] "shinyapps.io." https://www.shinyapps.io/ (accessed Aug. 28, 2021).

[33] "Shiny - Reactivity - An overview." https://shiny.rstudio.com/articles/reactivity-overview.html (accessed Sep. 05, 2021).

[34] "Shiny Reactlog." https://rstudio.github.io/reactlog/index.html (accessed Sep. 05, 2021).

[35] "Shiny - Gallery." https://shiny.rstudio.com/gallery/ (accessed Aug. 07, 2021).

[36] P. Laplante, *Requirements Engineering for Software and Systems*, 3rd ed. CRC Press.

[37] “The prototyping tool for teams | Framer.” https://www.framer.com/ (accessed Sep. 21, 2021).

[38] “Download R-4.1.0 for Windows. The R-project for statistical computing.” https://cran.r-project.org/bin/windows/base/ (accessed Aug. 08, 2021).

[39] “Download the RStudio IDE.” https://rstudio.com/products/rstudio/download/ (accessed Aug. 08, 2021).

[40] “Shiny - Getting started with shinyapps.io.” https://shiny.rstudio.com/articles/shinyapps.html (accessed Sep. 02, 2021).

[41] “rsconnect package.” https://rstudio.github.io/rsconnect/index.html (accessed Nov. 25, 2021).

[42] “Load Test Shiny Applications.” https://rstudio.github.io/shinyloadtest/ (accessed Nov. 22, 2021).

[43] D. Granjon, “A Bootstrap 4 Version of shinydashboard.” https://rinterface.github.io/bs4Dash/index.html (accessed Sep. 02, 2021).

[44] “Monitoreo de infraestructura y monitoreo en la nube: Site24x7.” https://www.site24x7.com/es/ (accessed Nov. 26, 2021).

# ANEXOS

**ANEXO A.** Widgets Shiny.

**ANEXO B.** Objetos y clases reactivas.

**ANEXO C.** Ejecución de reactividad.

**ANEXO D.** Modelo de entrevista y resultados.

**ANEXO E.** Código fuente del prototipo.

- **Enlace**: https://epnecuador-my.sharepoint.com/:f:/g/personal/bryan_simbana_epn_edu_ec/Ej9MTdEbZeJEg49Lwn6EvhABcUmiCEZQShQw6ClAHD8JBw?e=xrUFkT

**ANEXO F.** Videotutorial de la aplicación.

- **Enlace**: https://epnecuador-my.sharepoint.com/:f:/g/personal/bryan_simbana_epn_edu_ec/EmOHWGZ-kE1Hl9qqCAbEqL0B9V4DFwFr3JGFEYEUjcgvQQ?e=Y5Xcnp

**ANEXO G.** Ejemplo de reporte dinámico.

**ANEXO H.** Informe de rendimiento del prototipo.

**ANEXO I.** Gráficos reactivos

## ORDEN DE EMPASTADO