Ingeniería en Informática

Proyecto Final

TiX
Medición de Calidad de Servicio de los proveedores de Internet

---

Alumnos:

Karpovsky, Alan Ezequiel

Loreti, Nicolás Daniel

Tutor:

Alvarez-Hamelin, José Ignacio

Noviembre de 2014

Índice general

Capítulo 1

Introducción

1.1. Descripción del proyecto

1.1.1. Problemática

Generalmente cuando alguien contrata un enlace de Internet una de las características principales que evalúa es la capacidad del mismo, que puede ser o no simétrica, siendo simétrica en el caso en el que la capacidad de bajada como de subida sea la misma. En el caso particular de conexiones hogareñas usualmente la actividad principal se centra en consumir recursos de la red mientras se navega bajando textos, imágenes, videos, etc, y no en subir archivos o información a la misma. Por esto último es que generalmente se cuenta con conexiones asimétricas, es decir, más capacidad para la bajada de datos que para la subida de los mismos.

Una vez que ya se tiene contratado el enlace uno esperaría que la empresa cumpla con su parte y siempre dispongamos la capacidad máxima acordada. Lamentablemente, la calidad de este tipo de conexiones suele ser bastante pobre haciendo que el usuario no sepa si obtiene lo que efectivamente contrató.

Dado este escenario es bastante común que un usuario final tenga la necesidad de saber el estado de su conexión y para resolver este problema existen diversas soluciones en el mercado que pasaremos a repasar mostrando sus ventajas y desventajas.

1.1.2. Herramientas Disponibles

Las herramientas actuales para medir la calidad de un servicio se pueden clasificar en dos grandes grupos:

1. Soluciones de Hardware
2. Soluciones de Software

Soluciones de Hardware

Dentro de la más destacadas se encuentra la herramienta de Samknows[1] la cuál esta basada en un dispositivo que se conecta entre la computadora y el router desde el cuál se toma la conexión y se encarga de medir en tiempo real todos los datos que circulan por el cable o Wi-Fi. El nombre particular del dispositivo es *whitebox* y luego de una rápida y fácil instalación permite tener un panel administrador web con todos los datos de los tipos de paquete que circulan por la red, la capacidad del enlance en función del tiempo y otras estadísticas mucho más puntuales. También cuenta con un sistema de reporte mensual vía e-mail que el usuario puede disponer y archivar como así también una aplicación mobile para Android desde la cuál se pueden ver todas las estadísticas de la plataforma.

Whitebox además se asegura de realizar las pruebas cuando el usuario no esta consumiendo o subiendo recursos a Internet para no afectar la navegabilidad y realiza un buen manejo de la privacidad del usuario ya que no revisa en detalle de los paquetes que están circulando por el dispositivo.

A nivel técnico todas las Samknows Whitebox se encuentran basadas en una distribución de Linux derivada de OpenWrt y se requiere que varios puertos TCP y UDP estén habilitados para que el dispositivo pueda realizar las diferentes pruebas como así también que la computadora sea capaz de responder una solicitud de tipo "ICMP ping".

El almacenamiento de la información se realiza dentro de la Whitebox y se envía directamente a Samknows sin ocupar espacio de almacenamiento en los nodos a testear.

Dentro de las diferentes test que realiza se encuentran los siguientes por mencionar algunos:

1. Test de velocidad: Realiza pruebas de bajada y subida con conexiones

[1] http://www.samknows.com/broadband/index.php

simples TCP/UDP como así también múltiples, en el caso de TCP lo realiza generalmente con 3 conexiones concurrentes.

2. Navegar a través de Explorador: Mide el tiempo que se tarda en obtener el archivo HTML los recursos referencias por el mismo.

3. VoIP: Esta prueba se realiza sobre UDP usando las mismas características y propiedades (tamaño de paquetes, demoras, tasa de transferencia) que en el codec G.711. La configuración estándar realiza 500 paquetes para la subida y 500 para la bajada.

Como desventaja podemos comentar que esta solución solo está disponible en Estados Unidos y Reino Unido, resultando difícil su implementación en otros lugares, particularmente en Latinoamérica.

Soluciones de Software

Están basadas en medir la tasa de transferencia[bps] por un método de saturación del enlace. Esta metodología realiza un uso extensivo en el momento de la medición impidiendo al usuario navegar con normalidad durante ese instante de tiempo, y además sólo es un muestreo de lo que ocurre durante todo un día. Este es el caso de herramientas como:

1. Ookla: `http://www.speedtest.net/` genera múltiples conexiones TCP a través de solicitudes HTTP. Realiza un promedio de los valores obtenidos tomando hasta 30 muestras por segundo.

2. M-Lab: `http://www.measurementlab.net/` El mismo se ejecuta sobre flash por lo que es necesario tener instalada la extensión que permite correr este tipo de aplicaciones sobre el explorador. En caso de no tenerlo instalado puede resultar molesto tener que descargarlo para poder medir el estado del enlance.

3. Comscore: A través del envío de un fichero y el tiempo transcurrido en la transmisión calcula la capacidad del canal usando solo una conexión TCP.

Estos métodos hace que sea difícil poder determinar si el servicio que uno está obteniendo es bueno ya que la ventana de prueba es muy acotada y depende pura y exclusivamente de la situación particular y condiciones en las que se realiza la medición.

Conclusión

Podemos decir que si bien la solución por hardware es la más acertada está lejos de implementarse de forma local. En el caso particular de Latinoamérica no se cuenta con este tipo de soluciones y dada la complejidad que puede implicar la realización y la implementación de estos métodos resulta mucho más sencillo realizarlo a través de un software que cualquier usuario pueda usar o instalar en cualquier computadora.

Dentro de las soluciones de software disponibles como ya se mencionó lo hacen a través de saturación de enlance y por periodos de tiempo muy cortos haciendo que el resultado dependa del momento en el que se realiza y llevando tal vez a conclusiones equivocadas.

1.1.3. Objetivo y solución propuesta

En esta propuesta se diseñará e implementará el software para un nuevo paradigma de medición activa, cuya utilización del servicio de Internet es sólo el 1 % de la capacidad contratada. Además, permite el monitoreo las 24hs verificando la calidad del servicio otorgado por la empresa proveedora. Dicha herramienta implementará la metodología de medición y la presentación de resultados, tanto en el cliente (multiplataforma) como en el servidor (Linux). El almacenamiento de las mediciones podrá realizarse en una base de datos que permita su análisis en tiempo real.

También intervienen variables que agregan ruido a lo que uno está analizando como lo son distancias y cargas sobre la red que ya no son posibles de controlar por el proveedor de Internet, por ejemplo, no sabemos si la página que se está testeando esta ubicada en la misma ciudad desde la que estoy midiendo o se encuentra ubicada al otro lado del mundo. Otro punto que también resulta difícil de determinar son las diferencias se deben a congestión de la red por fuera del proveedor de Internet, ya que el protocolo sobre el cuál esta basado Internet realiza el mejor esfuerzo.

El presente proyecto quiere resolver la problemática que plantea determinar la calidad de una conexión de Internet sin saturar el enlace a lo largo del tiempo.

Como se comentó en la sección anterior existen varios problemas que están referidos a latencias, congestiones y seguimiento de una medición continua que permita establecer una conclusión fidedigna.

Para esto se desarrollará una herramienta que permita realizar mediciones de forma constante en el tiempo sin saturar la conexión que uno está intentando analizar. Esto va a permitir que el usuario pueda seguir navegando con total normalidad sin que noté el trabajo que está realizando la herramienta de forma transparente.

También se intentará reducir al mínimo los problemas de latencias por distancias y de congestión de la red haciendo que, los clientes en dónde se estén corriendo las pruebas y el servidor de medición, se encuentren relativamente de forma cercana (geográficamente).

En síntesis, el objetivo del presente proyecto es diseñar e implementar una herramienta para la medición de la calidad del servicio de Internet, basada en nuevas técnicas de medición activa con muy bajo impacto en uso del acceso. La misma será multiplataforma (Mac® y Linux).

Capítulo 2

Marco Teórico y Arquitectura Básica

2.1. Introducción Teórica

Para calcular la capacidad de la conexión que se está queriendo analizar bastaría con saber el tamaño del paquete a enviar y el tiempo que tarda ese paquete en transmitirse por el canal. Dentro de esta simplificación que reduce todo a un caso ideal aparecen, en la realidad, una serie de factores que complican un poco más el proceso. Entre ellos se encuentran:

1. Sincronización de relojes en maquinas aisladas.
2. Manejo de paquetes de protocolos intermedios.
3. Latencias en la red fuera de nuestro control.

Para solucionar estos inconvenientes se envía un paquete de una longitud adecuada, que contenga información de los tiempos de salida y llegada a cada extremo.

De esa forma si se tiene dos computadoras, llamémoslas C1 y C2, bajo el siguiente protocolo se podría determinar la capacidad del canal:

1. C1 envía un paquete a C2 con timestamp T1
2. C2 recibe el paquete y agrega el timestamp T2
3. C2 envía el paquete a C1 y agrega el timestamp T3

4. C1 recibe el paquete de C2 y agrega el timestamp T4

Por lo que $Tiempo\_ida\_vuelta = (T4 - T1) - (T3 - T2)$ y si los caminos virtuales establecidos en la comunicación son idénticos para la ida como para la vuelta, el tiempo de envío del paquete es el $Tiempo\_ida\_vuelta$ divido por dos.

Luego teniendo el tamaño del paquete y el tiempo de un tramo se tiene la capacidad del canal en forma teórica.

2.2. Arquitectura general del sistema de medición

La aplicación está concebida como tres módulos independientes: El **cliente de medición** es el aplicativo instalado en los dispositivos, el cual servirá para tomar registros sobre el estado del enlace y enviarlos periódicamente al servidor. El **servidor de medición** es el encargado de procesar los datos de los clientes, como así también es el punto hacia el cual los clientes envían sus paquetes para medir la calidad del enlace. Por último, la **aplicación web** es el medio por el cual los clientes pueden visualizar on-line los datos recabados en sus distintos dispositivos a través de los distintos clientes de medición instalados en ellos.

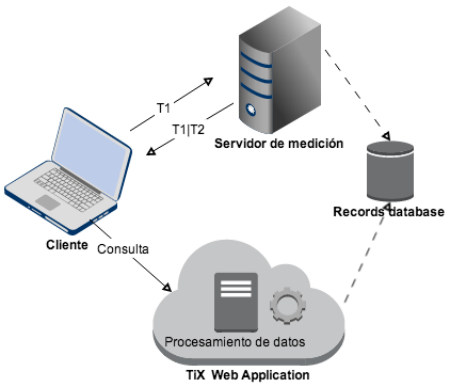

Figura 2.1: Arquitectura general del sistema

2.2.1. Servidor de Medición

El servidor tiene tres tareas que resultan fundamentales:

1. Atender a cada uno de los clientes que esta corriendo el aplicativo.
2. Procesar los datos cuando sea necesario.
3. Persistir los resultados en la base de datos.

Atender pedido de Clientes

Para poder realizar esto con una alta eficiencia es necesario que el programa distribuya la carga en varios hilos de ejecución. Como el servidor también procesa los datos y se comunica con la base de datos, es de vital importancia que cuando un cliente se quiera comunicar este disponible para atenderlo y no se pierdan paquetes con información que puede resultar muy valiosa para la medición incurriendo en errores y retrasos.

Procesar Datos

Una vez que el servidor cuente, para un cliente dado, con una cantidad suficiente de datos para realizar el procesamiento estadístico de los mismos, luego de que este se encuentre terminado, el servidor deberá eliminar la información que ya no es necesaria para realizar el próximo cálculo. Esto hará que la información no se acumule y el sistema pueda seguir trabajando sin saturarse haciendo un uso eficiente del espacio de disco.

Persistir Resultados

Por último una vez que ya el servidor haya recibido los datos y los haya procesado será necesarios persistirlos en la base de datos. Este proceso es de vital importancia para poder mostrar con gráficas y estadísticas el estado de una conexión a lo largo del tiempo.

2.2.2. Cliente

Multiplataforma

El cliente de medición fue concebido para poder correr en diversas plataformas. Debido a esto se decidió utilizar el framework *Kivy* de desarrollo sobre *Python* para el frontend. *Kivy* es una biblioteca *Python* de código abierto para el desarrollo de interfaces de usuario. Una vez empaquetado de forma correcta, el cliente podría correr sobre Windows, Linux, Mac OSx, Android y iOS ya que

Kivy se encarga de proveerle todas las dependencias necesarias.

Al momento de escribir este documento el cliente corre únicamente en OSx y Linux. El desafío para portarlo en otras plataformas está en entender su estructura de directorios y la forma en la que son creados, modificados y guardados los archivos.

Guardado y Envío de las mediciones

La medición de la calidad del enlace se efectúa mediante el envío de paquetes de forma periódica entre cliente y servidor que luego, a través de algunos cálculos de diferencias de tiempos, dan como resultado un parámetro de calidad de enlace.

Debido a que es altamente probable que los relojes de cliente y servidor no estén en sincronía, cada paquete es sellado con un *timestamp* al entrar o salir del cliente/servidor. Esto permite luego obtener las diferencias de tiempos necesarias para el cálculo del estado del enlace.

Dentro del sistema de archivos del dispositivo que contiene instalado el cliente, una carpeta es creada en la que se van almacenando los registros de las mediciones. Cuando el aplicativo detecta que ya existe suficiente información como para ser transmitida al servidor, genera un mensaje largo que concatena el contenido de todos los registros existentes al momento, lo envía al servidor y luego procede con el borrado local de los mismos. Este proceso tiene varias justificaciones: se necesitan un mínimo de datos para ser procesados; carece de sentido enviar cada uno de las mediciones al servidor en tiempo real ya que agregaría un gasto de recursos innecesario; asimismo se optimizan las escrituras en el servidor, enviando datos sólo cuando estos son significantes.

Privacidad y Cifrado

Al crearse una nueva instalación, se crean un par de claves públicas y privadas en el dispositivo que ha instalado el cliente de medición. La clave pública es luego enviada al servidor de medición, quien aloja esta información en su base de datos.

Utilizando un algoritmo de clave asimétrica de clave pública y privada podemos asegurar la confidencialidad e integridad de los mensajes enviados entre los clientes y los servidores a través de una red no segura como lo es Internet.

Este punto resulta crítico para poder dar seguridad a cada uno de los usuarios sobre la privacidad de sus datos y el manejo que se hace de los mismos.

Una vez que estas claves son creadas, el cliente está listo para empezar su ciclo de medición.

2.2.3. Aplicación Web

Manejo de Credenciales

La aplicación web utiliza un simple formulario de registro para la carga de nuevos usuarios. El mismo es protegido de *bots* mediante un código CAPTCHA.

Los nuevos usuarios sólo deben elegir una casilla de correo y una contraseña, los cuales serán los datos necesarios para el posterior inicio de sesión.

Visualización de Resultados

La principal función de la aplicación web es la de permitirle al usuario visualizar, de una forma cómoda, ordenada y agradable, los datos de las mediciones recabadas por los clientes instalados en sus dispositivos.

La aplicación web presenta al usuario, luego del inicio de sesión, un dashboard en el que se puede observar un gráfico de calidad y utilización junto con algunos botones para trabajar sobre los datos expuestos. Ellos son:

- Filtro por fecha: Rango de fechas para el cual se desea ver las mediciones.
- Filtro upstream / downstream: Permite la graficar sólo las curvas de subida o bajada.
- Instalaciones por proveedor de Internet: Permite mostrar el agregado de todos los para una determinada instalación (bajo la etiqueta *General*) o filtrar los datos para un determinado proveedor de Internet.
- Calidad y utilización: Permite ver sólo la calidad, sólo la utilización o ambas.

Por último, pensando en la usabilidad, el usuario tiene la posibilidad de elegir una instalación como *default* para que ésta sea mostrada al comienzo de cada nuevo inicio de sesión.

Exportación de información

La aplicación web le da la posibilidad al usuario de exportar un archivo *CSV* (valores separados por comas) conteniendo todos los resultados históricos de la calidad y utilización de su enlace. El archivo CSV separa estos datos por fecha y por instalación.

Asimismo, en caso de poseer permisos de administrador del sistema, se ofrece la posibilidad de exportar un archivo *CSV* con la medición agregada de todos los usuarios e instalaciones existentes; obteniendo datos agrupados por proveedor de Internet y fecha.

2.2.4. Base de datos

Concurrencia, Persistencia y Recuperación de Datos

Es el punto del sistema en el que se concentra toda la información procesada por el servidor en base a los paquetes enviados por los clientes como así también toda la información personal de los usuarios que han creado una cuenta.

Dado que la cantidad de consultas a la base de datos crece conjunto con el aumento de usuarios utilizando el sistema. Esta componente en particular se vuelve un punto crítico y requiere un alto grado de confidencialidad frente a eventos concurrentes.

Por último, también es posible recuperar los datos para realizar futuros análisis por afuera de la plataforma y los gráficos mostrados, de esta forma se tiene un seguimiento continuo e histórico de cada cliente que se puede extraer en cualquier momento.

2.3. Proceso de medición

Como se mostró en las secciones anteriores la medición en si esta compuesta por un paquete fragmentado en 4 por la capa IP que ayuda a determinar la capacidad del enlance asumiendo que las latencias y rutas elegidas por los routers intermedios son simétricas e iguales para cada uno de ellos.

Particularmente, los paquetes enviados por el cliente tienen dos tipos de rellenos: existe el **relleno largo** y el **relleno corto**. Esto se hace para simular distinto tipo de carga o stress sobre el enlace. Ambos rellenos son simplemente caracteres elegidos en forma aleatoria con la diferencia de que el relleno largo,

como su nombre lo indica, es de mayor longitud que el relleno corto, haciendo así variar el tamaño del paquete.

El proceso que corresponde a esta etapa de la medición se podría enumerar en los siguientes pasos:

1. El cliente enviará por UDP al servidor de medición un paquete el cual, básicamente, contiene un *timestamp* T1.
2. El servidor, al recibir el paquete, le pondrá un *timestamp* T2 y luego, antes de enviárselo nuevamente al cliente, le pondrá un *timestamp* T3.
3. El cliente recibirá la respuesta del servidor y le colocará un *timestamp* de recepción T4.
4. Finalmente, la tupla conformada por

   `T1|T2|T3|T4`

   será almacenada localmente por el cliente en un archivo de registros

Cuando el archivo de registros local es suficientemente grande (actualmente 1 minuto de medición), el cliente aprovecha el espacio existente en los paquetes denominados "largos" y le envía al servidor vía UDP los datos de ese minuto de medición.

De esta forma el servidor obtiene las mediciones del cliente para que puedan ser procesados y mediante un proceso estadístico (que trabaja con ventanas de 1 hora de datos) obtener un valor cuantitativo sobre la calidad de la medición.

2.3.1. Calidad del tráfico según el modelo Autosimilar

El proceso estadístico se ejecuta una vez que el servidor recibe 60 minutos de registros consecutivos por parte del cliente. La matemática asociada a ese proceso viene dada por lo siguiente:

El tráfico de datos que se observa en las redes puede modelarse como *autosimilar*, el cual es un modelo fractal del tráfico que permite mostrar y caracterizar las correlaciones de largo alcance. Si bien se ha utilizado durante mucho tiempo el modelo de Poisson (por ejemplo en las llamadas telefónicas), éste no posee correlaciones de largo alcance. El modelo que usamos en este trabajo posee tres

parámetros:

- El valor medio m, medido en una ventana de tiempo T;
- el coeficiente de varianza a, en la misma ventana T;
- y el parámetro de Hurst $H \in [0.5, 1)$ también en T.

El último representa el grado de *autosimilaridad* del tráfico, que siendo $H = 0.5$ converge al tráfico del modelo de Poisson, y cuando $H \to 1$ las correlaciones tienden a presentarse en ventanas con $T \to \infty$. Un ejemplo interesante es el de la altura de los ríos que sigue este modelo, descubierto por el hidrólogo Hurst cuando analizaba los registros históricos de más de 5000 años del Nilo. Dado la complejidad del problema se recomienda la lectura de: Self-Similar Network Traffic and Performance Evaluation, Kihong Park and Walter Willinger (editors), Wiley-Interscience, 2000.

En nuestra herramienta nos valemos de la estimación de H y del porcentaje la utilización del enlace para determinar si el proveedor nos puede entregar la capacidad contratada. Determinamos en cada minuto si el tráfico supera un umbral de H cuando la utilización es baja, en ese caso decimos que no se cumple con la calidad ya que muestra que no se podrá alcanzar un elevado porcentaje de utilización del enlace; en caso contrario se cumple con la calidad. Entonces, agrupando de cada diez minutos, establecemos el porcentaje de tiempo durante el cual se cumple con la calidad (por supuesto en una escala de diez valores, o saltos del 10 %).

Capítulo 3

Requerimientos y alcance

De todo lo discutido en la sección anterior se desprenden los siguientes requerimientos y casos de usos del proyecto.

3.1. Especificación del alcance

En la siguiente sección se presentará la especificación de requerimientos y el análisis funcional del sistema.

3.1.1. Especificación de requerimientos

Requerimientos funcionales - Cliente

RF1 - Validación de credenciales

Requerimiento funcional	
Relevancia	ALTA
Especificación	El cliente no debe iniciar el proceso de instalación hasta no haber iniciado una sesión válida. El usuario debe proveer las credenciales y el cliente deberá validar las mismas contra la base de datos presente en el servidor.

RF2 - Instalaciones duplicadas

Requerimiento funcional	
Relevancia	ALTA
Especificación	El cliente debe verificar que no exista una instalación previa en el equipo en el que se está intentando iniciar una nueva instalación. En caso de encontrarse evidencia de instalaciones previas, preguntar al usuario si desea proceder con la eliminación de la misma.

RF3 - Nombre de las instalaciones

Requerimiento funcional	
Relevancia	ALTA
Especificación	El cliente le dará al usuario la posibilidad de elegir el nombre de la nueva instalación. En caso de que este nombre entre en conflicto con alguna instalación previa en otro dispositivo bajo la misma cuenta, se mostrará una pantalla de error y se le pedirá al usuario que elija un nuevo nombre.

RF4 - Permisos de administrador

Requerimiento funcional	
Relevancia	ALTA
Especificación	Dado que parte del proceso de instalación requiere privilegios elevados, el cliente le pedirá al usuario que ingrese su contraseña de administrador cuando se requiera realizar alguna de estas tareas.

Requerimientos funcionales - Aplicación Web

RF5 - Registrar una cuenta

Requerimiento funcional	
Relevancia	ALTA
Especificación	El sistema debe permitir la registración de múltiples cuentas para distintos usuarios. La misma se utilizará para poder descargar los diferentes clientes y seguir los datos históricos de las instalaciones

RF6 - Descargar un cliente

Requerimiento funcional	
Relevancia	ALTA
Especificación	El sistema debe ser capaz de proveer una versión descargable de los clientes para los distintos sistemas operativos

RF7 - Creación y administración de una instalación

Requerimiento funcional	
Relevancia	ALTA
Especificación	Se debe poder crear una instalación del software en una máquina en concreto y que se permita su administración desde el panel de cuenta

RF8 - Elaboración de reportes

Requerimiento funcional	
Relevancia	MEDIA
Especificación	Se debe proveer reportes y gráficas de las distintas instalaciones del usuarios

RF9 - Exportación de datos

Requerimiento funcional	
Relevancia	BAJA
Especificación	El sistema debe ser capaz de exportar los datos en diferentes formatos

RF10 - Administración de la cuenta de Usuario

Requerimiento funcional	
Relevancia	MEDIA
Especificación	El sistema deberá darle la posibilidad al usuario de administrar su cuenta editando su información personal, password e instalaciones

RF11 - Cuenta de administrador

Requerimiento funcional	
Relevancia	MEDIA
Especificación	El sistema deberá contar con la posibilidad de existencia de un usuario de tipo super administrador que pueda eliminar usuarios existentes, como así también consultar sus mediciones

`RF12 - Filtrado de datos`

Requerimiento funcional	
Relevancia	BAJA
Especificación	El sistema deberá soportar filtros sobre los gráficos. Se deberá poder visualizar la data dentro de cierto rango de fechas especificado por el usuario como así también se deberá poder encender o apagar las distintas curvas desplegadas por los gráficos

Requerimientos no funcionales

Servidor

`Soporte de múltiples conexiones`

Requerimiento no funcional	
Relevancia	ALTA
Especificación	El servidor debe ser capaz de atender múltiples clientes en paralelo sin que esto afecte su performance.

`Alta disponibilidad`

Requerimiento no funcional	
Relevancia	ALTA
Especificación	El servidor debe contar con algún sistema automático de monitoreo que reaccione proactivamente ante fallos inesperados del aplicativo.

`Verificación de mensajes`

Requerimiento no funcional	
Relevancia	ALTA
Especificación	El servidor debe poder validar mediante el uso de la clave pública de cada cliente la integridad de los mensajes recibidos. En caso de detectar anomalías dentro del mismo, el mensaje debe ser descartado por considerarse malicioso.

`Procesamiento de datos`

Requerimiento no funcional	
Relevancia	ALTA
Especificación	El servidor debe iniciar un proceso paralelo (hijo) al identificar que un determinado cliente posee suficiente cantidad de datos como para realizar los cálculos. Durante este procesamiento de datos, la comunicación con los clientes no debe interrumpirse.

`Plataforma Linux`

Requerimiento no funcional	
Relevancia	ALTA
Especificación	El servidor debe poder correr sobre una plataforma Linux.

`Comunicación con Base de Datos`

Requerimiento no funcional	
Relevancia	ALTA
Especificación	El servidor deberá verificar que una instancia de una base de datos PosgreSQL está corriendo y permitiendo la correcta persistencia de datos. En caso de que esto no fuera así, el servidor retornará un error y detendrá su funcionamiento.

`Trasabilidad de registros`

Requerimiento no funcional	
Relevancia	ALTA
Especificación	El servidor deberá contar con un sistema de *logging* que permita un cómodo seguimiento del status del mismo.

`Escalabilidad`

Requerimiento no funcional	
Relevancia	BAJA
Especificación	La aplicación debe poder manejar más de 100 usuarios concurrentes

Cliente

`Portabilidad`

Requerimiento no funcional	
Relevancia	ALTA
Especificación	La aplicación tiene que ser capaz de correr en entornos MacOS y Linux

`Arranque automático`

Requerimiento no funcional	
Relevancia	ALTA
Especificación	El cliente se encargará, dentro del proceso e instalación, de crear los registros necesarios para que el ordenador inicialice el aplicativo durante el booteo.

`Creación de directorios`

Requerimiento no funcional	
Relevancia	ALTA
Especificación	El cliente creará toda la estructura de directorios necesaria para el correcto funcionamiento del sistema de medición.

Aplicación Web

`Usabilidad`

Requerimiento no funcional	
Relevancia	ALTA
Especificación	Las interfaces en lo que se refiere a creación de cuenta, instalación, visualización de reportes debe ser sumamente intuitiva para el usuario y fácil de realizar

`Medición y persistencia de datos`

Requerimiento no funcional	
Relevancia	MEDIA
Especificación	La aplicación deberá recabar información sobre el enlace de Internet y luego persistirla asociándola a una cuenta de usuario

`Performance`

Requerimiento no funcional	
Relevancia	MEDIA
Especificación	La generación de reportes no debe tardar más de 15 segundos y la creación de una cuenta los 20 segundos

3.1.2. Definición de actores

A continuación se describen los dos tipos de actores principales que interactuarán con la aplicación:

Usuario Final: Este actor corresponde a los usuarios que desean medir la calidad de su enlace de banda ancha por curiosidad o para estudio del mismo. Solo puede ver reportes de instalaciones que el mismo realizó y no puede ver las de nadie más.

Usuario Administrador: Este actor corresponde a un usuario que tiene una visibilidad completa de la plataforma, los diferentes usuarios, instalaciones y reportes. Cualquiera de estos datos están disponibles en cualquier momento y a demanda del mismo.

3.1.3. Casos de uso

Cliente

`Creación de una Instalación`

Caso de uso	
Trazabilidad	RF1 - RF2 - RF3
Complejidad	MEDIA
Descripción	1. El usuario final abre el cliente descargado para su plataforma. 2. El cliente ejecuta el aplicativo descargado desde su cuenta en la Aplicación Web. 3. Ingresa su usuario y contraseña que son validadas por el aplicativo. 4. En caso de haber una instalación existente, la aplicación avisa al usuario y pregunta si quiere borrarla para instalar una nueva. 5. Luego de seleccionar nueva instalación. Ingresa un nombre para la nueva instalación que no debe estar duplicado con alguna de las que ya posee ese usuario. 6. El programa realiza todas las actividades necesarias para dejar el programa corriendo.

`Ingreso de contraseña de administrador`

Caso de uso	
Trazabilidad	R4
Complejidad	BAJA
Descripción	1. El usuario selecciona el botón de instalar una vez seteado los parámetros de la misma y de estar logueado correctamente. 2. La aplicación solicita el ingreso de la contraseña de administrador del sistema operativo dónde se está realizando la instalación. 3. el usuario ingresa correctamente la contraseña y el aplicativo prosigue con la instalación hasta su finalización.

Aplicación Web

`Crear una cuenta`

Caso de uso	
Trazabilidad	RF5
Complejidad	MEDIA
Descripción	1. El usuario final accede a la página de creación de cuenta. 2. El usuario final ingresa todos los datos requeridos. 3. El sistema valida las entradas ingresadas por el usuario. 4. Si todo es válido el sistema crea la cuenta y devuelve un mensaje al usuario.

`Descargar del primer cliente`

Caso de uso	
Trazabilidad	RF6
Complejidad	BAJA
Descripción	1. El usuario final ingresa con su usuario y contraseña. 2. El sistema ofrece los distintos clientes disponibles. 3. El usuario selecciona el que corresponde a su sistema operativo. 4. El sistema envía el cliente seleccionado al usuario final.

`Descargar de un Cliente con instalaciones ya existentes`

Caso de uso	
Trazabilidad	RF6
Complejidad	BAJA
Descripción	1. El usuario final ingresa con su usuario y contraseña 2. El usuario final selecciona la opción de nueva instalación. 3. El sistema ofrece todos los clientes disponibles. 4. El usuario selecciona el que corresponde a su sistema operativo.

`Editar una instalación`

Caso de uso	
Trazabilidad	RF7
Complejidad	MEDIA
Descripción	1. En el menú principal selecciona la opción para editar instalación 2. Selecciona la instalación que desea editar del listado 3. Puede cambiar el nombre en el campo, ver la clave pública de la misma o eliminarla.

`Marcar una instalación como predeterminada`

Caso de uso	
Trazabilidad	RF7
Complejidad	MEDIA
Descripción	1. Selecciona la instalación del menú principal 2. Luego selecciona el botón de marcar como default 3. Se refresca la pantalla y la instalación default queda configurada.

`Filtrar una fecha`

Caso de uso	
Trazabilidad	RF8
Complejidad	BAJA
Descripción	1. Selecciona una instalación del menú principal. 2. El sistema ofrece las opciones de filtrado. 3. El usuario selecciona el tipo de congestión que desea filtrar. 4. Los cambios se ven reflejados luego de que recargue la página.

`Filtrar un tipo de congestión`

Caso de uso	
Trazabilidad	RF8
Complejidad	BAJA
Descripción	1. Selecciona una instalación del menú principal. 2. El sistema ofrece las opciones de filtrado. 3. El usuario selecciona el tipo de congestión que desea filtrar. 4. Los cambios se ven reflejados luego de que recargue la página.

`Filtrar la instalación por proveedor de Internet`

Caso de uso	
Trazabilidad	RF8
Complejidad	BAJA
Descripción	1. El usuarios final selecciona una instalación del menú principal 2. El sistema ofrece las distintas mediciones para cada proveedor de Internet 3. El usuario selecciona el proveedor de Internet que requiere filtrar 4. El gráfico con los datos correspondientes a la instalación e proveedor de Internet seleccionados se muestran.

`Generar un reporte de usuario`

Caso de uso	
Trazabilidad	RF8
Complejidad	BAJA
Descripción	1. El sistema muestra la opción en el menú principal del usuario. 2. El usuario final selecciona la opción 3. El sistema muestra todas las estadísticas correspondientes agrupadas por proveedor de Internet.

`Exportar un CSV`

Caso de uso	
Trazabilidad	RF8
Complejidad	BAJA
Descripción	1. El usuario final selecciona una instalación determinada. 2. El sistema ofrece los distintos proveedor de Internet y la gráfica general por default. 3. El usuario final selecciona la opción de exportar en el gráfico. 4. El sistema realiza una descarga al ordenador del usuario final.

`Exportar toda la información a CSV`

Caso de uso	
Trazabilidad	RF9
Complejidad	BAJA
Descripción	1. El administrador se loguea con su cuenta. 2. El sistema ofrece el botón de descarga. 3. El administrador selecciona el botón y recibe la información del servidor.

`Cambiar la contraseña`

Caso de uso	
Trazabilidad	RF10
Complejidad	MEDIA
Descripción	1. El sistema ofrece acceder al estado de la cuenta en el menú principal. 2. El usuario final selecciona su cuenta. 3. El sistema ofrece los campos editables . 4. El usuario realiza los cambios correspondientes y selecciona guardar.

`Eliminar usuario como administrador`

Caso de uso	
Trazabilidad	RF11
Complejidad	BAJA
Descripción	1. El usuario ingresa al sistema como administrador. 2. El administrador busca por el usuario a borrar. 3. El administrador selecciona borrar sobre la fila que corresponde al usuario a eliminar.

`Filtrar fechas en las gráficas`

Caso de uso	
Trazabilidad	RF12
Complejidad	BAJA
Descripción	1. El usuarios final selecciona una instalación del menú principal 2. El sistema ofrece las distintas mediciones para cada proveedor de Internet. 3. El usuario selecciona el proveedor de Internet que requiere filtrar 4. El gráfico con los datos correspondientes a la instalación e proveedor de Internet seleccionados se muestran. 5. Por sobre el gráfico se muestran las fechas iniciales y finales para el filtrado, ingresarlas y presionar el botón filtrar. 6. Se muestra el gráfico en las fechas escogidas.

Capítulo 4

Detalles de Implementación

En este capítulo se explicará en detalle la implementación de las tres componentes principales del proyecto en cuestión: servidor, cliente y aplicación web.

4.1. Servidor

Como ya se ha mencionado anteriormente, la función del servidor es la de atender las conexiones entrantes de los distintos clientes como así también procesar los datos y persistir los las mediciones dentro de los registros de la base de datos.

El servidor consta de un *bucle* infinito que va aceptando nuevas conexiones y creando hilos para atender a cada una de ellas por separado sin bloquear la llegada de nuevas peticiones. Cada uno de esos hilos analiza el tipo de mensaje enviado por el cliente y actúa en consecuencia: si el mensaje no contiene datos que deban ser guardados, el servidor simplemente le pone su *timestamp* al final del mensaje enviado por el cliente y se lo devuelve. En cambio, si el servidor detecta que el mensaje sí contiene información relevante que debe ser procesada, además de enviar la respuesta con su *timestamp*, se encarga de extraer los datos y guardar los mismos dentro del sistema de archivos.

Por último, cuando el servidor detecta que un determinado cliente tiene, para una determinada instalación, suficientes datos, lanza el proceso de cálculo de parámetros sobre el enlace.

4.1.1. Base de Datos

Para el motor de base de datos se decidió utilizar **PostgreSQL** por su fácil instalación e integración con la aplicación web hecha en Java. No debemos olvidar que esta última tiene dos fuentes de datos: los datos creados por la aplicación en si misma (usuarios, passwords, configuraciones, etc) y los resultados del procesamiento y cálculo sobre los archivos de registro enviados por los clientes.

4.1.2. Sistema de Archivos

El servidor aloja, dentro de la carpeta */etc/TIX/records*, los registros de cada uno de los clientes. Para hacerlo, se sigue la siguiente convención: una carpeta es creada por cada tupla de (IP, id de cliente, instalación). Es decir, el id de cliente es un número unívoco que identifica a cada cliente dentro del sistema; dado que la IP puede variar y dado que un usuario puede poseer más de una instalación en distintos dispositivos, se decidió utilizar estos tres parámetros para agrupar los registros. A continuación se muestra la estructura genérica del sistema de archivos:

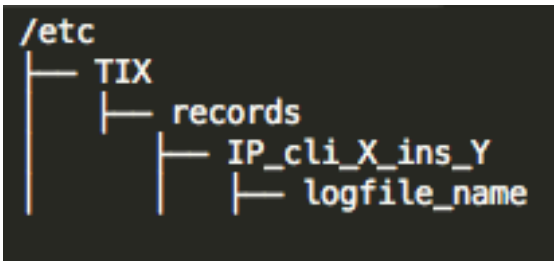

Figura 4.1: Sistema de archivos

En la siguiente imagen puede verse un ejemplo concreto del sistema de archivos:

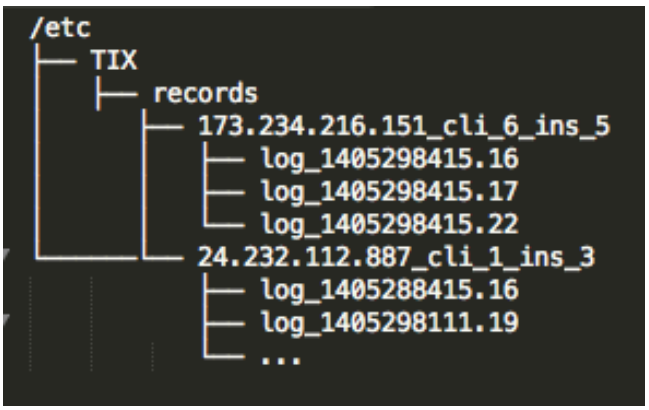

Figura 4.2: Ejemplo del sistema de archivos con datos

Recordar que cada carpeta de registros tendrá, como máximo, 60 archivos, ya que una vez alcanzado ese número la data es procesada y la ventana de medición corrida, por lo que los archivos más viejos son eliminados.

4.1.3. Cifrado

Para el cifrado se utilizó el módulo **rsa** de *Python* que contiene todas las funciones necesarias para el manejo de la criptografía.

Los paquetes con datos enviados del cliente hacia el servidor tienen el siguiente formato:

```
DATA|publicKeyPlain|signedMeessage|msg
```

Dado que llegan codificados en *base64*, el servidor los decodifica, separa sus partes y utiliza la clave pública para verificar el mensaje firmado por el cliente. Esto se realiza mediante el uso de la función *verify* del módulo *rsa*:

```
rsa.verify(client_plain_msg, client_signed_msg, publicKeyPlain)
```

Si la verificación de la integridad es satisfactoria, el servidor de medición hará una consulta a la base de datos de la aplicación web para obtener el número de cliente y la instalación en base a la clave privada con la que el mensaje ha sido firmado.

4.1.4. Procesamiento de datos

El cliente de medición simplemente mide diferencias de tiempos y va enviándole sus registros al servidor; este último es el que hace el trabajo de procesamiento y cálculo.

El servidor va alojando los registros de los clientes en una estructura de directorios diseñada especialmente para ello ya explicada anteriormente. Cada vez que recibe un nuevo registro, chequea si el directorio ya contiene una cantidad de registros suficiente como para procesar los datos y, en tal caso, dispara un subproceso y en paralelo le responde al cliente como es habitual. El subproceso se encarga de levantar toda la información disponible en los registros y hacer los cálculos necesarios para poder extraer los parámetros de calidad y estado de enlace relevantes.

Finalizado el procesamiento el servidor aloja los resultados en la base de datos para que éstos puedan ser consultados y graficados desde la aplicación web.

Se utiliza unas rutinas en R para estimar el parámetro H, según el estimador R/S y Wavelet, ver: R. G. Clegg, "A practical guide to measuring the hurst parameter," 21st UK Performance Engineering Workshop, School of Computing Science Technical Report Series, CSTR-916, University of Newcastle, pp. 43-55, 2006. Entonces, tal como se describió en la sección 2.3.1, se analiza por cada minuto si H supera un umbral cuando al agilización del enlace es baja; en dicho caso se considera que no hay calidad (porque hay congestión), y sino lo contrario. Luego se agrupa por tramos de diez minutos mostrando el resultado en porcentaje de calidad; por ejemplo 80 % de calidad significa que sólo el 20 % del tiempo estuvo congestionado (dos minutos de los diez).

4.1.5. Multi-Threading

Para poder atender todas solicitudes de forma eficiente y sin alterar la permaneced del sistema es necesario poder manejar una arquitectura de varios *hilos* en parles especialmente en los siguientes dos casos:

- Paquete entrante de un cliente: En caso de que el servidor este realizando otra tarea y un paquete arrúe al servidor es necesario que este lo pueda entendiere antes de que el *times* del lado del cliente finalice interrumpiendo la comunicación y perdiendo el paquete
- Procesamiento de datos: Dado que este proceso puede demorar varios seguros es importante que durante ese tiempo el servidor no este enfocado en esa sola tarea.

Creando un thread distinto para cada una de estas acciones nos aseguramos que el servidor pueda distribuir mejor la carga de trabajo y se encuentre dispo-

nible la mayoría del tiempo para atender todos los pedidos que sean necesarios.

4.1.6. Registros

El servidor maneja en simultáneo muchas conexiones y procesos; es por esto que es importante poder llevar un control de qué está haciendo el servidor y, en caso de error, poder detectaré qué produjo la falla. Pensando en esto se optó por utilizar el módulo de **registros** nativo de Python el cual permite, de manera muy sencilla, crear registros cuyos mensajes tienen distinta severidad (crítica, advertencia, informativa, etc).

Dentro del código del servidor, se especificaron mensajes puramente para *debuging* y otros mensajes informativos o de error presentes en la aplicación de producción. Basta cambiar el parámetro *level* del *log* para informarle qué nivel de *logging* se desea obtener. A continuación un ejemplo de configuración del sistema de *logging*:

```
imparto logging
# crearte logro
logra = logging.galguee('udpServerTiempos')
hdlr = logging.FileHandler('/var/tmp/tixUDPServerTiempos.log')
logger.setLevel(logging.DEBUG)

# create console handler and set level to debug
hdlr.setLevel(logging.DEBUG)

# create formatter
formatter = logging.Formatter('%(asctime)s %(levelname)s %(message)s')

# add formatter to ch
hdlr.setFormatter(formatter)

# add ch to logger
logger.addHandler(hdlr)

# Logger examples
# logger.debug("debug message")
# logger.info("info message")
# logger.warn("warn message")
# logger.error("error message")
```

```
# logger.critical("critical message")
```

4.1.7. Monitoreo

Debido a que el servidor de medición debe tener alta disponibilidad, se decidió utilizar un *daemon* llamado MONIT para delegar sobre él el monitoreo. Este aplicativo básicamente chequea de forma regular que el servidor esté levantado y, encaso de que esto no fuera así, registra la falla en un archivo de registro y procede a levantarlo nuevamente.

Logs de Monit:

```
$ tail -f set logfile /var/log/monit.log
```

Luego de levantar cinco veces consecutivas al servidor caído, *monit* dejará de repetir esta acción para evitar daños de hardware.

4.2. Cliente de medición

4.2.1. Validación de Credenciales

Para poder identificar correctamente a un usuario el cliente esta conectado con la base de datos por lo que este es capaz de validar a un usuario a través de las credenciales que ingresa.

El usuario y la contraseña a utilizar deben ser las mismas con las cuáles se tiene acceso a la interfaz web y fue creada la cuenta, no hay claves o usuarios duplicados.

El proceso de conexión con la base de datos se realiza a través del módulo *psycopg2* de Python y esta se puede ver en el siguiente fragmento de código la llamada:

```
self.INSTANCE = psycopg2.connect(host=databaseHost,
port=databasePort,user=databaseUsername,
password=databasePassword,database=databaseName)
```

Si las credenciales no son correctamente ingresadas el usuario es notificado con un mensaje de error y le pide que las ingrese nuevamente.

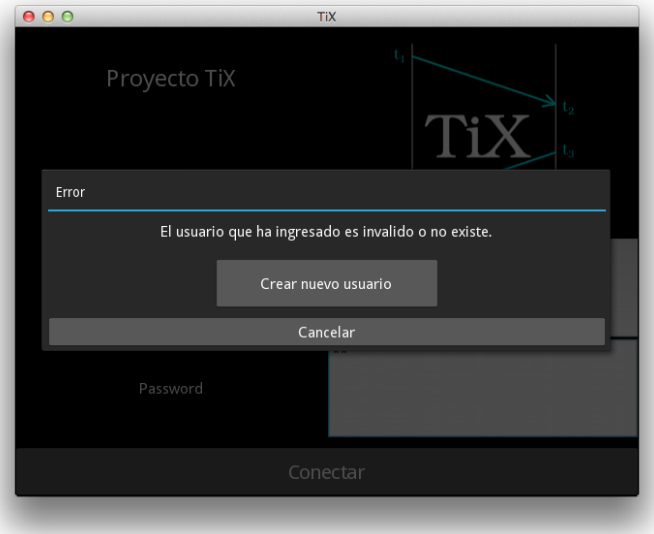

Figura 4.3: Credenciales inválidas.

4.2.2. Detección de Instalaciones previas

Una vez validadas las credenciales el cliente obtiene una lista de todas las instalaciones. Estas son comparadas contra la ingresada como nueva instalación y el usuario es notificado para que las vuelva a ingresar.

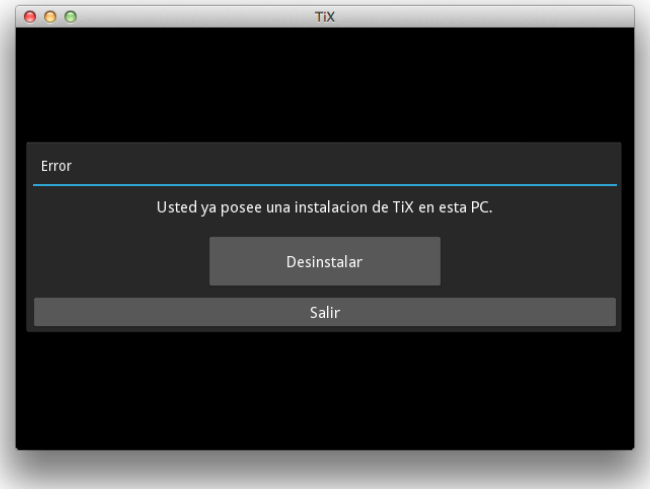

Figura 4.4: Instalación ya existente en el sistema.

4.2.3. Inicio Automático

Mediante las funciones dentro del instalador:

```
def darwin_install_client():
  # Set as root, copy to launchers, and tell osx we're inside
  os.system('sudo chown root %s' % src_path(darwinLaunchFile))
  cp_rf('%s' % src_path(darwinLaunchFile), darwin_launch_path(darwinLaunchFile))
  sys_return = os.system('sudo launchctl load %s' % darwin_launch_path(darwinLaunchFile))

def linux_install_client():
  print "about to copy"
  chmod_x(src_path(startupAppCaller))
  cp_rf(src_path(startupAppCaller),init_path(startupAppCaller))
```

Figura 4.5: Funciones dentro del instalador para inicio automático

La aplicación se inciará automáticamente cada vez que se encienda la PC haciendo este proceso totalmente transparente para el usuario.

4.2.4. Armado de Paquetes

Cuando el archivo de registros local es suficientemente grande, cada un minuto, el cliente aprovecha el espacio existente en los paquetes denominados "largos" y le envía al servidor vía UDP los de medición.
Para lograr esto, se antepone el prefijo `DATA;;` dentro del relleno, avisándole al servidor que éste paquete no tiene una secuencia de bytes aleatorios como relleno, sino que contiene datos que necesitan ser almacenados.
A continuación se muestra un ejemplo del contenido de un paquete con relleno largo que contiene datos:

```
"DATA;;" + base64.b64encode(publicKeyPlain) +
";;" + base64.b64encode(signedMessage) + ";;"
+ filereg.name + ";;" + base64.b64encode(msg) + ";;"
+ TO_BE_FILLED_WITH_RANDOM_CHARS
```

Como puede verse, luego del prefijo que anuncia los datos, se envía, codificado en base64, la clave pública, el mensaje (registros de medición) firmado, el nombre del archivo de registro a usar por el servidor, el mensaje original y por último se completa con caracteres seleccionados al azar hasta llegar al tamaño deseado de mensaje.

Cabe destacarse que cada paquete enviado por el cliente es codificado y luego firmado con la clave privada, asegurando así la integridad de los datos. El servidor, al recibirlo, hace la verificación y, en caso de no verificar, el paquete

es desestimado.

4.2.5. Instalador

El instalador esta desarrollado bajo Kivy, un proyecto comunitario liderado por desarrolladores profesionales. La biblioteca soporta múltiples plataformas permitiendo que corra sobre Linux, Windows, OS X, iOS y Android.

Es totalmente gratuito y puede ser usado para productos comerciales, cuenta con una API muy bien documentada y tutoriales básicos para empezar a utilizarla.

El motor gráfico está construido sobre OpenGL ES 2, escrito en C usando Cython y probado con pruebas de regresión.

4.3. Aplicación web

4.3.1. API

Dado que diversas aplicaciones necesitan consultar datos de la aplicación web, se especificó una API para manejar esta comunicación:

- **bin/api/newISPPost**: Recibe como **payload** un archivo JSON con la clave *isp_name* cuyo valor debe ser el nombre de un proveedor de Internet. Lo que hará es crear un nuevo proveedor de Internet bajo el nombre especificado y retornar el id asignado al mismo. En caso de que el proveedor de Internet ya exista, no lo crea nuevamente.
- **bin/api/authenticate**: Recibe un JSON con dos parámetros: **name** y *password*. Si los mismos matchean con algún usuario válido, retorna un archivo JSON con el ID del usuario y una lista de las instalaciones para el mismo.
- **bin/api/newInstallationPost**: Recibe un JSON con cuatro parámetros: *user_id*, *password*, *installation_name* y *encryption_key*. A diferencia del método anterior, éste recibe el id del usuario, la contraseña, el nombre de una nueva instalación y la clave pública a la que ésta quedará asociada.

4.3.2. Validación de permisos de usuario

Existen algunas páginas dentro de la aplicación web que sólo deberían poder ser accedidas por usuarios administradores. Por ejemplo, un usuario administrador puede acceder al *admin panel* desde donde puede eliminar usuarios, ver estadísticas para un determinado usuario y/o descargar archivos con el agregado de las mediciones para todo el sistema.

Para evitar problemas de seguridad, los *controladores* (modelo MVC) de la aplicación web hacen una verificación solicitando que el usuario logueado en la sesión sea de tipo *ADMIN* como se muestra a continuación:

```
User me = getSessionUser(session);

if (!me.getType().equals(UserType.ADMIN)) {
    mav.addObject("errorDescription",
        "No tiene permisos para acceder aquí.");
    mav.setViewName("error");
    return mav;
}
```

Como se puede observar en este extracto de código, si no sucede que el usuario sea de tipo administrador, se establece un mensaje de error y se retorna.

4.3.3. Desafíos de diseño

Cifrado de mensajes

Por razones de seguridad es importante asegurarse de que los paquetes enviados del cliente a servidor no sean modificados en la mitad del camino. En caso de no validar esto un atacante podría intencionalmente modificar los paquetes y hacer que, por ejemplo, las mediciones de calidad de enlace para un determinado ISP se vean afectadas (tanto positiva como negativamente).

Para llevar adelante esta tarea se utilizó el módulo de código abierto **Python-RSA** implementado por Sybren Stuvel[1]. Éste permite, de manera muy sencilla, encriptar, desencriptar, firmar y verificar mensajes como así también generar pares de claves públicas y privadas bajo el estándar RSA.

Cabe destacarse que la encriptación/firma se hace de acuerdo con el estándar *PKCS*, haciendo a este módulo compatible con *OpenSSL*. Asimismo las claves son salvadas en formato *PEM* y *DER*.

[1] https://bitbucket.org/sybren/python-rsa

Manejo de claves públicas

Uno de los desafíos encontrados durante el desarrollo fue el hecho de tener que transmitir desde el cliente al servidor la clave pública una vez generada. Como se explicó anteriormente, la clave pública es alojada en la base de datos del servidor y es un identificador unívoco de cada usuario: cuando llega un nuevo paquete con datos al servidor, éste último utiliza la clave pública presente en el mensaje para conocer a qué usuario se le deben asignar dichos datos una vez procesados.

Luego de hacer algunas pruebas y al ver que muchas veces esta identificación de usuario fallaba nos dimos cuenta de que las claves públicas suelen tener espacios y enters dentro de ellas. Debido a posibles errores en la transmisión o en el salvado de este *string* en la base de datos, sumado a problemas de codificación de caracteres, la validación solía fallar. El problema se resolvió cuando se decidió empezar a enviar todos los mensajes con la codificación *base64* como así también utilizar esta codificación para el salvado de las claves en la base de datos.

Funcionalidades del servidor

`Monitoreo de daemon`

Para solucionar el problema que resulta de tener el servicio que atiende a los clientes en el servidor detenido o caído ante cualquier eventualidad de forma automática se decidió implementar una herramienta de monitoreo.

El daemon estará revisando cada dos minutos el servicio y en caso de que este no este corriendo lo iniciará nuevamente durante un total de 5 veces consecutivas. En ese caso, es indicio de que hay un error grave y que requiere ser revisado en forma particular.

`Eliminación de registros: ventana de 1 hora`

En caso de que llegue un paquete de un cliente con información para ser almacenada el servidor revisa que en la carpeta haya menos de 60 archivos. Cuando se llega a 60 archivos significa que una hora de resultados se ha acumulado y hay que proceder a realizar el cálculo estadístico de la información almacenada en ese lapso de tiempo.

Una vez que ha sido calculado se procede a borrar los últimos 10 minutos para hacer espacio a que nuevos archivos lleguen al servidor y se pueda realizar la próxima medición.

En caso de que se pierda la conexión con el cliente y no se tenga una hora continua de datos se procederá a remover 10 archivos hasta que se alcance una ventana de 1 hora completa y continua de mediciones.

El manejo de esta lógica se realiza a través de comandos de sistema de archivos dependientes del sistema operativo.

Biblioteca gráfica HighCharts

La aplicación TiX se basa en proveer a los usuarios de gráficas sobre la calidad y el estado del enlace a Internet, es por ello que uno de los principales desafíos es la búsqueda de una herramienta para graficar grandes volúmenes de datos. Para esto se decidió utilizar la biblioteca gráfica **HighCharts**[2].

HighCharts es una biblioteca gratuita para aplicaciones no comerciales que permite graficar datos utilizando las tecnologías *JS* y *HTML5*. Ésta se encarga de agregar una capa de abstracción y permite al desarrollador simplemente enviar los puntos a graficar, el formato del gráfico, los títulos para los ejes y el framework se encarga del resto (ej. compatibilidad entre distintos navegadores).

Si bien resuelve la mayoría de los problemas, no resuelve el filtrado de datos por fechas[3]. Debido a esto se decidió implementar esta funcionalidad en nuestro backend de la siguiente forma: Por defecto el sistema gráfica todos los *records* que existen en la base de datos para ese determinado usuario/instalación. Cuando el usuario ingresa una rango de fechas específicas entre las que quiere ver los datos, el servidor realiza una consulta a la base de datos solicitando sólo los puntos necesarios y éstos son luego enviados al *frontend* de HighCharts para su posterior graficación.

Cabe destacarse que HighCharts ofrece gráficas dinámicas y no estáticas. Es suficientemente flexible como para permitir al programador definir reglas y acciones según distintos disparadores (ej. acción de click en un determinado punto del gráfico). Por último cabe destacarse que también ofrece la posibilidad de imprimir el gráfico o exportarlo en distintos formatos: imagen (PNG/JPEG) y PDF (SVG).

[2] http://www.highcharts.com
[3] La versión paga de HighCharts sí implementa esta funcionalidad

Detección de un proveedor de Internet

El servidor de medición recibe los paquetes con los datos de la calidad del enlace y a su vez tiene la IP desde la cual esas mediciones fueron hechas. Dado que la aplicación web permite al usuario visualizar sus mediciones filtrando por ISP, es necesario poder obtener, a partir de la IP, el sistema autónomo a la que ésta pertenece.

Para hacer la conversión entre la dirección IP y el sistema autónomo se utilizó la base de datos del proyecto *I am here*[4]. Puede encontrar más información sobre esta temática en el anexo.

[4] `http://lanet-vi.fi.uba.ar/i_am_here_/buscar.py`

Capítulo 5

Herramientas

5.1. Documentación técnica

5.1.1. Plan de avance

- **Etapa preliminar**
  - Definición de Requerimientos
  - Investigación de Herramientas
- **Etapa 1**
  - Definición de plataformas a utilizar
  - Definición del modelo de datos
  - Definición preliminar de las vistas
- **Etapa 2**
  - Maquetado HTML
  - ABM de usuarios
- **Etapa 3**
  - ABM de Instalaciones
  - Creación de la vista de reportes
- **Etapa 4**
  - Implementación y configuración del Servidor de Medición
  - Implementación y configuración de los Clientes de Medición (Linux y Mac)

- **Etapa 5**
  - Conexión del servidor de medición con la aplicación web
  - Exportación de reportes
- **Medición, documentación y presentación**
  - Elaboración del informe final y documentación técnica.
  - Presentación

5.1.2. Herramientas y plataformas utilizadas

Para el desarrollo de la aplicación web y el cliente/servidor de medición se utilizaron las siguientes herramientas y plataformas.

- **Ubuntu**[1]
- **MacOS**[2]
- **Java:** Es un lenguaje de programación orientado a objetos que corre sobre una maquina virtual multiplataforma. Java es rápido, seguro y confiable y unos de lenguajes más utilizados en la actualidad.[3]
- **Python:** Es un lenguaje de programación interpretado cuya filosofía hace hincapié en una sintaxis que favorezca un código legible.[4]
- **Postgres:** Es un sistema de gestión de base de datos relacional orientado a objetos y libre.[5]
- **Spring:** Es una plataforma para el desarrollo de aplicaciones y contenedor de inversión de control, de código abierto para la plataforma Java.[6]
- **Jetty:** Se enfoca en crear un servidor web sencillo, eficiente, importable y pluggable. El tamaño tan pequeño de Jetty lo hace apropiado para ofrecer servicios Web en aplicaciones Java. En el caso de TiX, Jetty fue usado como servidor de staging y testing mientras que Apache Tomcat fue elegido como servidor de producción.[7]

[1] http://www.ubuntu.com
[2] http://www.apple.com/osx/
[3] http://www.java.com
[4] https://www.python.org/
[5] http://www.postgresql.org
[6] http://spring.io
[7] http://www.eclipse.org/jetty/

- **Apache Tomcat:** Es un contenedor web con soporte de servlets y JSPs. Incluye el compilador Jasper, que compila JSPs convirtiéndolas en servlets. En TiX se combina este motor de servlets con el servidor web Apache para servir el sitio a los clientes.[8]

- **Nginx:** Es un servidor web/proxy inverso ligero de alto rendimiento. El proyecto TiX utiliza nginx configurado como un proxy reverso el cual redirige el tráfico del puerto 80 al Apache Tomcat en el puerto 8080.[9]

- **Highcharts:** Highcharts es una biblioteca escrita enteramente en Javascript que ofrece una manera sencilla de construir gráficos interactivos a embeberse en aplicaciones web.[10]

- **Bootstrap:** Twitter Bootstrap es un *framework* de software libre para diseño de sitios y aplicaciones web. Contiene plantillas de diseño con tipografía, formularios, botones, cuadros, menús de navegación y otros elementos de diseño basado en HTML y CSS, así como, extensiones de JavaScript opcionales adicionales.[11]

[8] http://tomcat.apache.org
[9] http://nginx.org
[10] http://www.highcharts.com
[11] http://getbootstrap.com

Capítulo 6

Conclusiones

6.1. Resultados

Como se comentó en la introducción del presente trabajo, existen diversos tipos de herramientas para solucionar la necesidad de medir la calidad de un enlace de Internet, tanto por hardware como por software con sus ventajas y desventajas. El objetivo del presente trabajo consistió en construir una plataforma que permitiese tomar lo mejor de ambos mundos dando la posibilidad de realizar la medición directamente por software pero analizando los datos de forma continua y sin saturar el enlace.

Respecto a los temas técnicos es importante destacar el soporte de varios sistemas operativos, la implementación de claves públicas y privadas que proveen un nivel de seguridad sumamente alto y los diferentes tipos de plataformas y bibliotecas que intervienen en todo el proceso haciendo que su integración no resulte algo trivial.

Desde el punto de vista del usuario, se cuenta con una interfaz totalmente gráfica lo cuál hace que sea muy sencillo para el mismo todo el proceso de instalación y vista de gráficos.

La posibilidad de expansión de la plataforma hacia nuevas funcionalidades o cambios dentro de la misma siempre fue un punto a tener en cuenta para que el proyecto pueda seguir creciendo a futuro. Por esto mismo se cuenta con diferentes módulos bien separados y que pueden ser trabajados de forma totalmente independiente.

Por último cabe mencionar que el proyecto en si mismo es de particular in-

terés en el ámbito local y regional ya que no existen alternativas *Open Source* que realicen la misma tarea.

6.2. Agradecimientos

Agradecemos el financiamiento del proyecto **FRIDA 2012** "Propuesta para el establecimiento de un nodo del measurement lab (**m-lab**) en América Latina", de **LANIC** (`http://www.lacnic.net`).

6.3. Futuras mejoras

Información estadística comparativa

Sería interesante analizar la posibilidad de que la plataforma proporcione a los usuarios con información estadística comparativa de su calidad de enlace respecto a la media de su zona, su ISP, etc. Una estrategia similar implementa SpeedTest[1] al informale al usuario cómo está posicionada su velocidad dentro del ranking de velocidades a nivel regional y mundial.

TiX Mobile

A futuro se podría realizar un aplicativo móvil *iOS / Android* que permita tomar el mismo tipo de mediciones sobre la calidad del enlace pero desde el dispositivo móvil. A diferencia de la señal WiFi convencional (o la red cableada), implementar TiX Mobile tiene sus complejidades a resolver dado la variación en la señal del teléfono en cada instante y el constante movimiento del dispositivo, como así también la baja disponibilidad de la red para enviar los datos.

Interface de status

Actualmente TiX corre en *background*. Una vez finalizado el proceso de instalación, el usuario de TiX no tiene ningún tipo de feedback sobre el estado del programa excepto que corra comandos avanzados sobre la consola para visualizar el proceso en ejecución. Podría ser útil desarrollar algún tipo de interfaz gráfica con un icono en la barra de tareas que le permita al usuario saber que TiX está tomando mediciones y, eventualmente, conocer algunos datos *on the fly* de la calidad del enlace. Asimismo este icono podría ser útil a la hora de informar errores en la recolección de datos, fallas en el envío de paquetes, etc.

[1] `http://www.speedtest.net/`

También permitiría darle más versatilidad al usuario para poder pausar y reanudar la medición cuando éste lo disponga sin tener que matar al proceso desde la consola.

Social media

Se podrían agregar botones de *share* dentro de la aplicación web TiX que le permitan al usuario compartir sus mediciones vía Twitter o Facebook.

Resumen de cuenta por Email

Mensualmente el usuario podría recibir un resumen de las mediciones recolectadas bajo su cuenta en el último mes.

Actualización remota

La aplicación debería contar con algún tipo de funcionalidad que le permita informarle a los usuarios sobre una nueva actualización disponible.

Capítulo 7

Anexo

7.0.1. Servidor de Medición

7.0.2. Dependencias del servidor

Para instalar la aplicación *udpServerTiempos.py* se necesitan algunas dependencias de Python que pueden instalarse como se explica a continuación:

```
$ wget http://peak.telecommunity.com/dist/ez_setup.py
$ sudo python ez_setup.py
$ sudo apt-get install python-psycopg2
$ sudo apt-get install libpq-dev
$ sudo easy_install Crypto
$ sudo easy_install rsa
$ sudo easy_install requests
$ wget https://ftp.dlitz.net/pub/dlitz/crypto/pycrypto/pycrypto-2.6.tar.gz
$ tar -xvzf pycrypto-2.6.tar.gz
$ cd pycrypto-2.6
$ sudo python setup.py install
```

7.0.3. Sistemas autónomos

Dado que el servidor TiX debe convertir direcciones IP en sistemas autónomos, es necesario mantener una base de datos local con estos datos. Para ello procedemos de la siguiente forma:

Instalamos **mysqlserver**:

```
$ sudo apt-get install mysql-server (configurar usuario root y password ******)
$ sudo service mysql start
```

Creamos la DB:

```
$ mysqladmin create ip_to_as --user=root --password=******
```

Descargamos de Internet la última DB e insertamos los records:

```
$ wget http://lanet-vi.fi.uba.ar/dbassr/last_20131215-iamheredb.sql.gz
$ gunzip last_20131215-iamheredb.sql.gz
$ mysql ip_to_as -uroot -p54bf1n6 < ./last_20131215-iamheredb.sql
```

Notar que en el primer comando *wget* se debe insertar la URL de la última DB disponible de *iamhere*.

7.0.4. Ejecución del servidor

Para correr el servidor, simplemente se debe acceder vía SSH al servidor TiX y ejecutar el siguiente comando. Nótese el uso de *screen* para separar el proceso de la shell y poder cerrar la conexión ssh sin mayores inconvenientes (-L indica logging):

```
$ ssh pfitba@tix.innova-red.net
$ sudo screen -A -m -L -d -S udpServer sudo python udpServerTiempos.py
```

En la misma carpeta en la que se encuentra el servidor TiX, se aloja un archivo de configuración con el siguiente formato:

```
[TiXServer]
tixBaseUrl = http://tix.innova-red.net/
databaseHost = localhost
databaseUsername=tix_user_db
databasePassword=***********
databaseName = tix_db
databasePort = 5432
installDirUnix = /etc/TIX
SERVER_HOST = 200.10.202.29
SERVER_PORT = 80
TEST_SERVER_HOST = 200.10.202.29
TEST_SERVER_PORT = 80
```

En él se definen los parámetros (host, usuario, contraseña, db y puerto) de la base de datos, como así también el puerto en el que recibirá conexiones el aplicativo y el directorio en el que alojará los records.//

7.0.5. Aplicación Web

7.0.6. Deployment

Ejecutar mvn package para generar el .WAR (quedará en el directorio *target*). En caso de que el esquema de la BD haya cambiado se deberá primero crear un WAR con el parámetro *setup.properties* en *CREATE* y luego de que se cree el esquema de la base de datos deberemos cambiarlo por *VALIDATE* para que no se borren todas las tuplas cada vez que se reinicie el servidor web.

setup.properties para el servidor de producción:

```
hibernate.show_sql              = false
hibernate.hbm2ddl.auto          = validate
hibernate.format_sql = true
hibernate.use_sql_comments = true
hibernate.dialect=org.hibernate.dialect.PostgreSQLDialect
jdbc.driverClassName=org.postgresql.Driver
jdbc.url=jdbc:postgresql://localhost:5432/tix_db
jdbc.username=tix_user_db
jdbc.password=*******
```

Copiar vía *scp* el WAR al servidor de TiX de la siguiente forma:

```
$ scp ./tix-1.0.war pfitba@tix.innova-red.net:/home/pfitba/tix.war
```

Mover el .WAR a la carpeta de **Tomcat7** *(/var/lib/tomcat/webapps)* para que sea explotado. Debe llamarse ROOT para que sea mapeado en */*. Si conserva el nombre tix.war, será mapeado en *http://tix.innova-red.net/tix*:

```
$ sudo rm -fr /var/lib/tomcat7/webapps/tix*
$ sudo mv tix.war /var/lib/tomcat7/webapps/ROOT.war
```

Si se desea generar el esquema de la DB de forma remota, hacer un tunel desde la PC local hacia pfitba@tix.innova-red.net de la siguiente forma para luego ejecutar el pgAdmin en la PC local para hacer los inserts:

```
$ ssh -L8082:localhost:5432 pfitba@tix.innova-red.net
```

Dejar Tomcat7 en el puerto 8080 y luego por iptables redirigir las conexiones entrantes al puerto 80 hacia el 8080. La última línea es para redirigir el tráfico INTERNO del servidor (por ejemplo para hacer el POST a la API y obtener el proveedor de Internet name dada una IP).

```
sudo iptables -F
sudo iptables -X
sudo iptables -t nat -F
sudo iptables -t nat -X
sudo iptables -t mangle -F
sudo iptables -t mangle -X
sudo iptables -P INPUT ACCEPT
sudo iptables -P FORWARD ACCEPT
sudo iptables -P OUTPUT ACCEPT
sudo iptables -t nat -A PREROUTING -p tcp --dport 80
-j REDIRECT --to-port 8080
sudo iptables -t nat -I OUTPUT -p tcp -d tix.innova-red.net
--dport 80 -j REDIRECT --to-port 8080
```

En caso de reinstalación del servidor, las reglas de iptables deben ser persistidas. Para ello utilizar:

```
sudo su -c 'iptables-save > /etc/iptables/rules.v4'
sudo su -c 'ip6tables-save > /etc/iptables/rules.v6'
```

Asimismo debe ser removido del startup el servicio de nginx para que no interceda con la ejecución de *Tomcat*:

```
sudo update-rc.d -f nginx remove
```

Reinicio del servidor

Si por algún motivo se quisiera reiniciar el servidor web, simplemente se debe ejecutar el siguiente comando:

```
$ sudo service tomcat7 restart
```

7.0.7. Extras

Lo primero que debe hacerse para poder utilizar el Cliente TiX es crearse un nuevo usuario entrando a la siguiente página web: http://tix.innova-red.net/tix/bin/account/register.

Una vez creada la cuenta, la aplicación web solicitará la descarga del empaquetado para la instalación del cliente. El usuario debe elegir el empaquetado basándose en su sistema operativo, descargarlo y luego instalarlo en su dispositivo.

Para mayor información sobre el proceso de descarga, instalación y uso del cliente de medición favor de consultar el **manual de usuario** adjunto.

7.1. Manual de Usuario

7.1.1. Creación de una cuenta

Para la creación de una cuenta se debe acceder a la *homepage* de TiX y luego presionar el botón *Instalar TiX* que se encuentra en la parte superior izquierda del contenido principal.

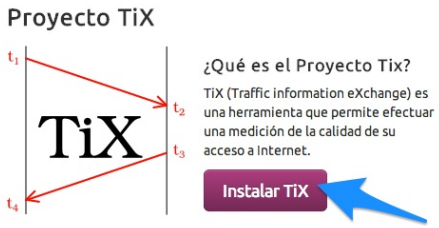

Figura 7.1: Botón de creación de cuenta en pantalla principal.

Inmediatamente luego será redirigido a la pantalla de creación de cuenta que se muestra a continuación:

Figura 7.2: Formulario de registro.

Simplemente deberá completar los campos del formulario y presionar el botón *Enviar*.
En caso de que alguno de los campos no cumpla con el formato exigido o que,

por ejemplo, el email indicado haya sido previamente registrado en la base de datos, se mostrará una caja de error describiendo las anomalías encontradas. El formulario cuenta con un código *captcha* para prevenir el registro de cuentas mediante bots.

Inicio de sesión

Una vez que se haya registrado satisfactoriamente en el sistema, podrá iniciar sesión con sus credenciales. Para hacerlo deberá ingresar éstas en la pantalla inicial tal como se muestra en la siguiente figura:

Figura 7.3: Inicio de sesión.

Si los datos ingresados son correctos, será automáticamente redirigido a su *dashboard* personal.

7.1.2. Descarga e instalación del aplicativo

En el caso en que no cuente con ninguna instalación activa, tan pronto como ud. inicie sesión, será automáticamente redirigido a la pantalla de descarga del cliente de medición.

Figura 7.4: Descarga de instaladores para los distintos sistemas operativos.

Deberá presionar sobre el instalador deseado dependiendo del sistema operativo con el que cuente y comenzará la descarga del cliente.

OSX

Una vez que haya descargado el *DMG* para OSX, simplemente debe abrirlo y ejecutarlo. El instalador le solicitará que ud. mueva a la carpeta */Applications* el archivo y luego, en caso de ser exitosa la instalación, la interfaz del cliente será visualizada.

Linux

La instalación en Linux es muy sencilla: Simplemente debe ejecutar el paquete con extensión *.deb* y presionar *Install package*. Si todo sale de forma correcta, la aplicación será lanzada automáticamente.

Instalación del cliente

Antes de comenzar una nueva instalación, el aplicativo verificará que no exista dentro del dispositivo una instalación previa. Esto se debe a que, un mismo ordenador no puede contener dos instalaciones en paralelo. En caso de que intentemos forzar una instalación en un equipo con otra previamente realizada, se visualizará un *popup* de notificación como el que se muestra a continuación, dándonos la opción de eliminarla:

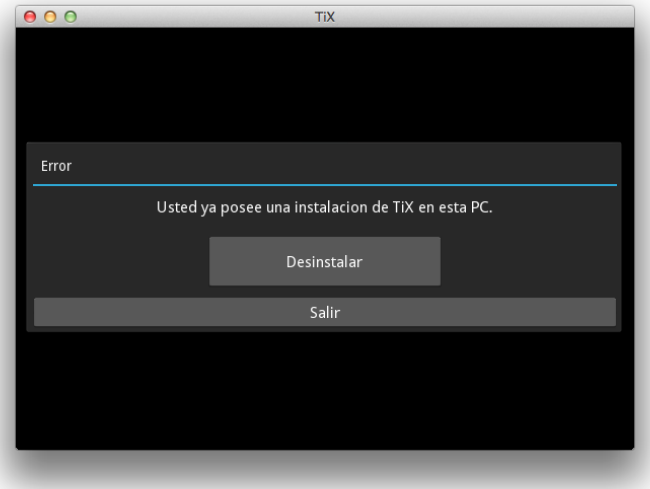

Figura 7.5: Instalación ya existente en el sistema.

Lo primero que veremos al abrir el aplicativo es una pantalla de inicio de sesión. En ella se deben introducir las credenciales con las que nos hemos registrado a *TiX*.

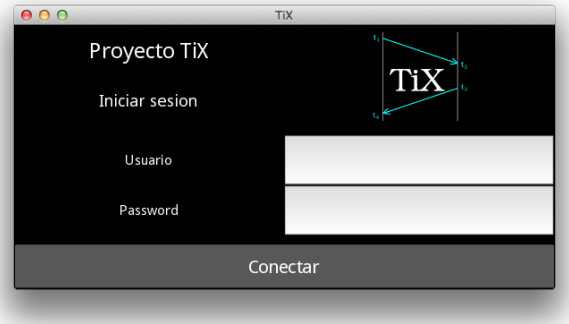

Figura 7.6: Inicio de sesión para instalación del cliente de medición.

Al ingresar satisfactoriamente nuestras credenciales, se desplegará un cuadro de diálogo pidiéndonos el nombre de la instalación que estamos creando. En caso de ya contar con instalaciones previas en otros dispositivos, las mismas serán mostradas justo debajo de la caja de texto. Tener en cuenta que no pueden existir instalaciones con nombres repetidos.

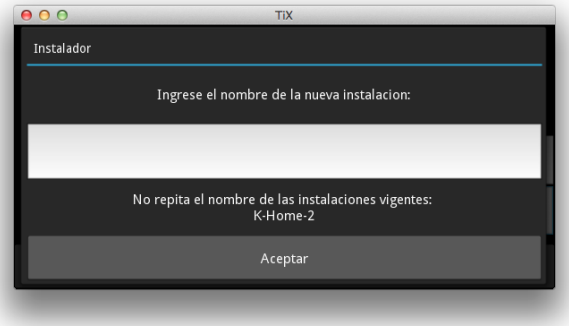

Figura 7.7: Ingreso de nombre para nueva instalación.

Una vez elegido el nombre, el aplicativo nos solicitará permisos de *administrador* en el sistema para poder generar los cambios necesarios como ser el registro de la aplicación dentro de la secuencia de *booteo*.

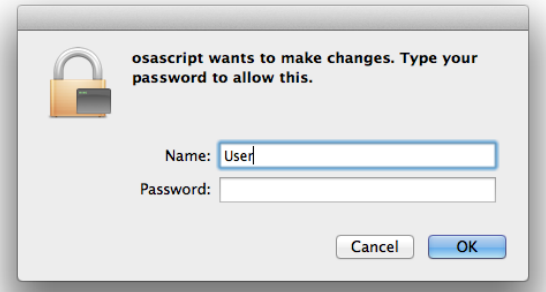

Figura 7.8: Credenciales de administrador para realizar cambios sobre el sistema.

Por último, si el proceso no devolvió ningún tipo de errores, el usuario visualizará un cartel informando que se ha creado la nueva instalación y el cliente de medición comenzará a correr.

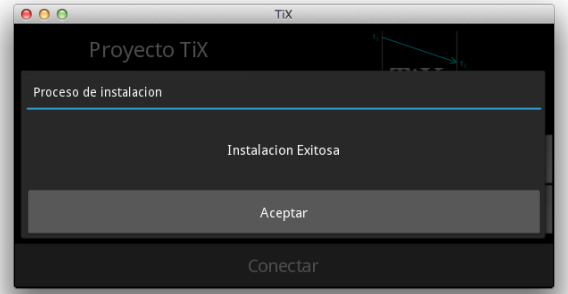

Figura 7.9: Notificación de instalación exitosa.

Finalizado este proceso, el usuario debería poder entrar al sitio web de *TiX* y visualizar la nueva instalación dentro de su *dashboard*.

7.1.3. Descripción de interfaz web

La interfaz de usuario, de ahora en adelante llamado *dashboard*, se divide en dos secciones principales: la columna de administración y navegación situada a la izquierda y el gráfico de mediciones junto a sus controles situado en la parte derecha de la pantalla.

Consulta de mediciones

Como se comentó, la función principal del *dashboard* es la de desplegar los datos recolectados por el cliente de medición.

El gráfico mostrará las mediciones correspondientes a la instalación seleccionada en el panel izquierdo:

Figura 7.10: División de mediciones por ISPs.

Como se puede notar en la figura, el sistema guarda las mediciones separadas por ISP. Si se presiona el botón *General*, el gráfico mostrará los datos agregados de todos los ISPs disponibles para la instalación en cuestión. En cambio, si se presiona el nombre de puntual de algún ISP, el gráfico mostrará los datos sólo para éste.

Si se cuenta con más de una instalación, el sistema permite la elección de una de ellas como predeterminada. Para hacer esto se debe presionar el botón *Set as default* que se encontrará debajo del nombre de la instalación, como muestra la figura a continuación.
Una vez seleccionada como *default*, la instalación será la primera en aparecer dentro del listado y será la que, cuando se abra el *dashboard*, muestre las mediciones dentro del gráfico.

Figura 7.11: Establecer una instalación como predeterminada.

Por otro lado, para un mejor manejo e interpretación de los datos, se dispone de un conjunto de botones accionables:

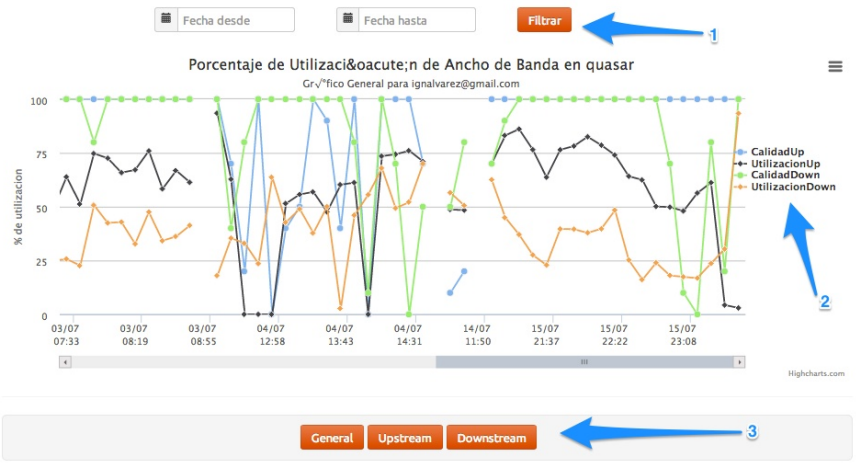

Figura 7.12: Acciones aplicables sobre el gráfico de mediciones.

1. **Filtrado por fechas:** Mediante los campos que se encuentran en la parte superior el usuario tiene la posibilidad de hacer un filtrado de las mediciones en un cierto rango de fechas.

2. **Calidad y utilización:** Estos controles situados a la derecha de las mediciones permiten mostrar u ocultar las curvas de calidad y/o utilización. Asimismo notar que los colores próximos al titulo de cada curva permiten saber fácilmente qué se está graficando.

3. **Bajada y Subida:** Los botones naranja permiten comandar qué datos se mostrarán en el gráfico. En caso de optar por la opción *General*, se mostrará tanto las mediciones de subida como las de bajada. En caso de optar por alguna de las otras, ya sea subida o bajada, se mostrarán sólo los datos correspondientes a ésta elección.

Acciones de usuario

Dentro del panel izquierdo de acciones de usuario podemos encontrar dos secciones bien distinguidas:

Figura 7.13: Formulario para actualización de datos personales.

Edición de perfil de usuario

El sistema permite la modificación de los datos principales del usuario. Estos son el nombre de usuario y la contraseña. En el *dashboard* de usuario presionar el botón *Mi cuenta*.

Figura 7.14: Formulario para actualización de datos personales.

Simplemente rellene los formularios con la información correspondiente y los datos de su cuenta serán actualizados. En caso de que alguno de los campos no cumpla el formato requerido o no valide, una caja de error será desplegada pidiendo revisión.

Agregado y borrado de instalaciones

Dentro del panel de comandos izquierdo encontrará los botones *Nueva instalación* y *Editar instalaciones*.

En caso de querer instalar el cliente de medición en un nuevo dispositivo, deberá presionar el primero de ellos y proceder como se explicó en la sección de instalación.

Asimismo puede visualizar y eliminar las instalaciones correspondientes bajo el botón *Editar instalaciones*.

Figura 7.15: Panel de edición de instalaciones.

Aquí se mostrarán todas las instalaciones que contiene la cuenta y se permitirá editar su nombre como así también ver su clave pública y/o eliminarlas. Notar que al eliminar una instalación se perderán todo el registro de mediciones de la misma.

7.1.4. Panel de administración

Si ud. cuenta con las credenciales del usuario administrador del sistema, en vez de ver un *dashboard* de mediciones al iniciar sesión, verá un panel de administración como el que se ilustra a continuación.

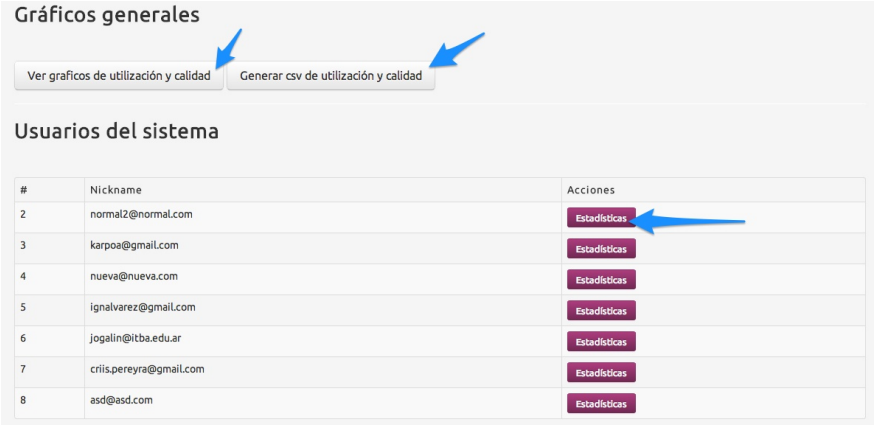

Figura 7.16: Panel del usuario administrador.

Este panel para usuarios privilegiados permite ver las estadísticas de medición para todos los usuarios del sistema.

Asimismo, en la parte superior se permite, mediante los botones de *Ver gráficos de utilización y calidad* y *Generar csv de utilización y calidad*, visualizar información general agregada del sistema con estadísticas basadas en las mediciones de todos los usuarios.

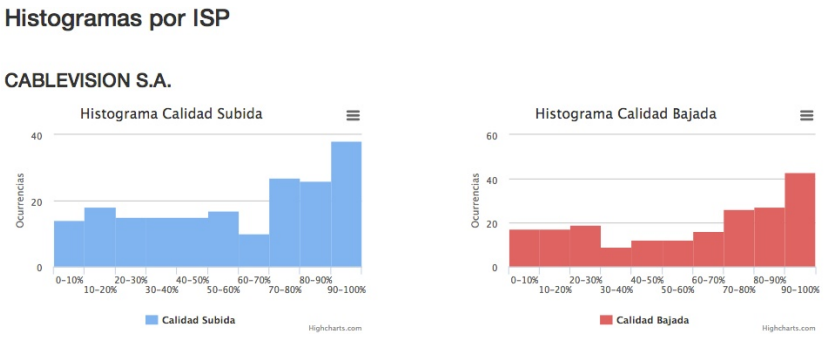

Figura 7.17: Histogramas de calidad.