ITBA

Extensión de TiX para medición de la calidad de los accesos Internet domiciliarios

por

Jose Ignacio S. Galindo, Cristian Adrián Pereyra

Tutor: Dr. José Ignacio Alvarez-Hamelin

Tésis de Ingeniería Informática

25 de Febrero de 2015

Índice general

Agradecimientos

En primer lugar se agradece al tutor de proyecto, que con su guía y ayuda, hizo posibile la finalización exitosa del mismo.

En segundo lugar se agradece al equipo de desarrollo anterior, que se mostró sumamente disponible a responder dudas que surgían a lo largo del desarrollo y a dar soporte cuando se los necesitaba.

También se agradece a LACNIC (http://www.lacnic.net) por el financiamiento del proyecto y a las demás personas que contribuyeron desde su lugar para hacer el proyecto posible.

Y finalmente se agradece a los usuarios, sino los cuales no se podría haber contado con la retroalimentación adecuada como para hacer la corriente versión de TiX usable, escalable y mantenible.

Capítulo 1

Introducción

La medición de la calidad de internet es una temática sumamente relevante en muchos ámbitos, tanto desde un punto de vista contractual (por ejemplo, para controlar el acuerdo de calidad de servicio prometido por alguna compañía proveedora de internet), hasta un punto de vista científico, como puede ser la evaluación del comportamiento de las personas de carga y descarga de contenido desde sus enlaces hogareños. Hoy en día el uso de internet impacta en prácticamente todas las actividades de la vida cotidiana, ya sea en el ámbito laboral, personal o educativo, la calidad de la conexión de internet puede ser un diferencial muy importante en la calidad de vida, en la productividad y en la educación de las personas.

De esta forma, la principal motivación del proyecto es la confección de un análisis estadístico de la realidad de la situación de internet en la Argentina, pudiendo así realizar distintos estudios sobre la calidad brindada por cada proveedor de internet (comúnmente llamados *ISP* o *Internet Service Provider*).

Para llevar a cabo este análisis, se continua con el desarrollo de TiX (Traffic information eXchange), el cual es un software capaz de realizar los análisis necesarios para conocer en profundidad la calidad de la conexión a internet en una oficina, un hogar o una universidad.

El presente estudio se enfoca en investigar nuevos requerimientos, mejorar y agregar diversas funcionalidades del software, y finalmente en llevar a cabo pruebas con un número significativo de usuarios para analizar los resultados obtenidos mediante el uso del mismo.

1.1. Motivación Inicial

La importancia de la calidad de conexión a internet en el mundo actual es sin duda indiscutible. Internet revolucionó la forma en la que hacemos absolutamente cualquier actividad, todas las empresas, universidades e instituciones en general dependen hoy en día de una conexión de calidad a internet.

Es aquí donde la distribución de los enlaces de red toma una relevancia muy alta. Y esta es una distribución que comúnmente, por motivos financieros, hace que todos los usuarios compartan de cierta forma esa capacidad en conjuntos de conexiones. Es decir, si todos los usuarios utilizaran el 100 % de su conexión probablemente no obtendrían la misma velocidad ya que el enlace por el cual se conectan a internet se vería saturado.

A su vez, saber cuantitativamente cuándo un enlace se encuentra saturado, sin ser administradores de esa red, resulta un problema no trivial de resolver, ya que sólo puede analizarse desde el *host* (dispositivo conectado, ej: PC, Smartphone, etc) que se conecta a internet en ese momento, ya sea por un dispositivo de hardware, o por una herramienta de software. Y ya que cada *host* puede tener un sistema operativo distinto, soportar estos sistemas también se torna un requerimiento para dicha herramienta.

Con este objetivo en mente, anteriormente al presente trabajo se desarrolla TiX como una solución a estos problemas: el mismo es un software que brinda mediante un análisis estadístico de diversos envíos de paquetes hacia un servidor, diversas variables que permiten cuantificar la calidad de un enlace o de un proveedor de internet.

1.2. Problemática

Durante el desarrollo y lanzamiento de TiX surgen nuevas problemáticas que abren puertas a distintas mejoras y cambios, que sumadas a las extensiones que ya fueron planteadas en trabajos anteriores, son parte del foco del presente trabajo.

El primero de los problemas se presenta en la práctica, donde, se notaron ciertas variaciones en las mediciones debido a diversos filtros que proveen los *ISPs* para

priorizar ciertos *paquetes* (unidad de datos con un formato dado) que viajan en internet. En particular, algunos *ISPs* priorizan los paquetes del tipo *TCP* (*Transmission Control Protocol*) por sobre los paquetes del tipo *UDP* (*User Datagram Protocol*). Ambos *TCP* y *UDP* son dos de los tipos de paquetes más comunes utilizados en las conexiones a internet, y TiX depende de la utilización de paquetes *UDP* para garantizar mediciones cualitativas.

Por otro lado, en la práctica también surgen inconvenientes con la distribución del ejecutable a usuarios finales en varias plataformas. Se encuentra en este punto una pieza fundamental para poder tener un alto nivel de adopción por parte de los usuarios, con lo cual su resolución resulta de suma importancia.

Finalmente, se encuentran varios inconvenientes técnicos dentro del servidor de medición que limitan la posibilidad de medir con una cantidad grande de usuarios en intervalos de tiempos grandes, con lo cual se decide mejorar el monitoreo del servidor y los algoritmos utilizados para realizar los cálculos con la finalidad de tener una mayor estabilidad y soportar una mayor carga de trabajo.

Una limitación encontrada en TiX es la forma en la que se visualizan los datos obtenidos de los usuarios, y más aún, la imposibilidad de visualizar los datos de distintos proveedores mas allá de ver los datos de un cliente en particular. Esto impide la posibilidad de tener comparativas de calidad entre proveedores. A esto se suma la falta de un resumen de resultados mensual, el cual permita al usuario final estar al tanto del resultado de sus mediciones, sin la necesidad de ingresar al sistema periódicamente para revisarlas.

Otra limitación, previamente mencionada en el trabajo anterior sobre TiX es la imposibilidad de realizar actualizaciones del sistema sin que el usuario tenga que reinstalar el ejecutable, y precisamente esta es otra de las mejoras sobre las cuales se decide hacer hincapié.

1.3. Objetivo y soluciones propuestas

Tomando en consideración el punto de partida actual, es decir, el software TiX en su versión inicial, se propone un conjunto de objetivos a cumplir y una propuesta de implementación para solucionar las limitaciones y problemáticas previamente mencionadas.

Como objetivo principal se pretende realizar pruebas con un conjunto de usuarios reales. Estos usuarios permitirán obtener resultados relevantes relacionados con la calidad de servicio de distintos proveedores de internet.

Para alcanzar el objetivo mencionado, se proponen las siguientes soluciones:

- La investigación sobre la posibilidad de implementar las mediciones de TiX mediante conexiones TCP, sin afectar la calidad de las mismas y la implementación de la solución en caso de ser posible.
- La creación de un empaquetado que permita instalar el ejecutable de TiX en diversas versiones de Linux, y un ejecutable para las versiones más modernas de *Mac OSX*.
- La creación de una extensión al ejecutable que permita que el mismo sea actualizado periódicamente y de forma automática, sin necesidad de intervención del usuario final.
- La extensión de las visualizaciones de los resultados en un módulo de reportes, disponible tanto para los usuarios como para los clientes, tanto en el sistema *online*, como en un reporte mensual enviado vía correo electrónico.
- La configuración y optimización del servidor de mediciones para poder soportar una carga mayor de usuarios y a fin de poder realizar pruebas con usuarios finales.

Capítulo 2

Arquitectura Básica

La aplicación consiste de **múltiples módulos** que interactúan entre sí, como puede verse en la figura 2.1. En las secciones subsiguientes se va a explicar el funcionamiento de cada módulo y su interconexión con los demás componentes del sistema.

2.1. Ejecutable para el usuario final

Su rol principal es el de medir la calidad del enlace mediante el envío de diversos paquetes de medición al servidor.

Uno de sus requisitos más importantes es el de ser multiplataforma, para permitir así llegar a la mayor cantidad posible de usuarios. Para conseguir una portabilidad en diversos sistemas operativos, se utiliza la biblioteca *Kivy*, la cual corre en el lenguaje de programación *Python*. Para la instalación de las dependencias se utiliza la biblioteca *PyInstaller*, la cual extrae las bibliotecas básicas del proyecto y permite hacerlo portable tanto para Linux, Mac, Windows y Android.

El principal inconveniente de la utilización de *PyInstaller* es que se requiere tener un sistema configurado y preparado con el sistema operativo para el cual se quiera compilar el ejecutable, es decir, se debe tener un sistema operativo *OSX* totalmente configurado para hacer un ejecutable de *Mac*. Teniendo esto en cuenta se decide crear diversas máquinas virtuales para poder compilar y ejecutar los diversos ejecutables.

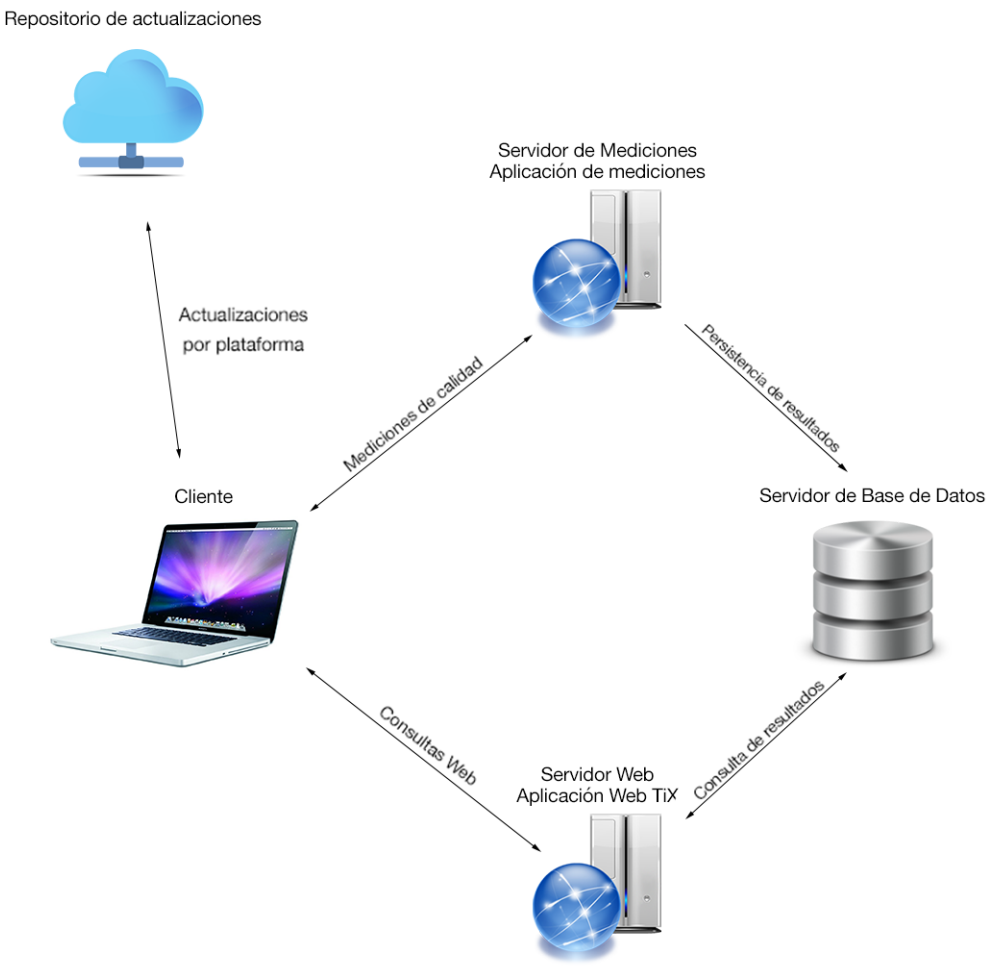

FIGURA 2.1: Arquitectura Básica de TiX

A su vez, el cliente almacena temporalmente y luego envía las mediciones al servidor de mediciones. El servidor posteriormente guarda esa información, y cuando tiene datos de a última hora, los procesa y crea una nueva entrada en la base de datos.

Como mejora adicional ahora también tiene la capacidad de actualizarse interactuando con el repositorio de actualizaciones. Esto permite introducir cambios en el sistema sin la necesidad de que los usuarios descarguen el software nuevamente. La implementación del sistema de actualizaciones es totalmente *ad-hoc*, ya que se tiene en cuenta la condición de que el proyecto sea multiplataforma.

2.2. Servidor de mediciones

Su rol es el de procesar las mediciones enviadas por los clientes, almacenando previamente un histórico de las mismas. Tras ser procesados, estos datos son almacenados en el servidor de base de datos para su uso posterior desde el servidor web, ya sea en reportes o en consultas del usuario.

2.3. Servidor web

Su principal rol es el de permitir el registro de los usuarios, la descarga de los clientes ejecutables y la visualización de los datos, ya sea en forma individual, o, como nueva funcionalidad, en forma de reportes agregados por proveedor, es decir, permitiendo comparar calidades de diversos proveedores de forma global utilizando todas las mediciones disponibles. Finalmente, es el servidor desde donde se envían dichos reportes vía correo electrónico de forma mensual.

2.4. Repositorio de actualizaciones

Su rol es el de proveer la información necesaria a los clientes sobre el último paquete de actualización disponible, los cuales consultan si hay alguna actualización una vez por día. El mismo se encuentra constantemente *online* utilizando los servicios en la nube de *GitHub* para proyectos *open source*.

2.5. Servidor de base de datos

Su rol es el de persistir y proveer la información relacionada con los resultados de las mediciones y las cuentas de usuario, asimismo, el mismo tiene configurados *backups* automáticos con una herramienta externa, mediante la cual se tiene *backups* de la base de datos con diversos intervalos de tiempo, para cubrir cualquier tipo de eventualidad que puedan ocurrir con los datos.

Capítulo 3

Requerimientos y alcance

De todo lo discutido en la sección anterior de desprenden los siguientes requerimientos y casos de uso 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 - Reportes

RF1 - Reporte del administrador

Requerimiento funcional	
Relevancia	ALTA
Especificación	El administrador debe poder ingresar desde su panel a una pantalla que visualice un reporte con información agregada de todos los usuarios del sistema.

RF2 - Histogramas en reporte del administrador

Requerimiento funcional	
Relevancia	ALTA
Especificación	Dentro de su reporte, el administrador debe poder ver histogramas para cada proveedor de internet con la información agregada para todos los usuarios del sistema. Para cada proveedor de internet se mostrarán 4 histogramas cada uno correspondiendo a los 4 parámetros relevantes (calidad subida, calidad bajada, utilización subida, utilización bajada).

RF3 - Diagramas de bloques en reporte del administrador

Requerimiento funcional	
Relevancia	ALTA
Especificación	Dentro de su reporte, el administrador debe poder ver diagramas de bloques para cada proveedor de internet con la información agregada para todos los usuarios del sistema. Para cada proveedor de internet se mostrará 4 diagramas de bloques en un sólo gráfico cada uno correspondiendo a los 4 parámetros relevantes (calidad subida, calidad bajada, utilización subida, utilización bajada). A su vez, a un lado del título del proveedor de internet se mostrará la cantidad de ocurrencias los diagramas correspondientes.

RF4 - Filtros en reportes del administrador

Requerimiento funcional	
Relevancia	ALTA
Especificación	Dentro de su reporte, el administrador debe ver por defecto toda la información agregada de los usuarios del último mes. Además, el administrador debe poder filtrar la información de los usuarios por fecha de comienzo y fin, día de la semana y por hora de comienzo y fin.

RF5 - Reporte del usuario

Requerimiento funcional	
Relevancia	ALTA
Especificación	El usuario debe poder ingresar desde su panel a una pantalla que visualice un reporte con información del último mes acerca de todos los proveedores a los que alguna vez se conectó.

`RF6 - Tabla con medianas mensuales en el reporte del usuario`

Requerimiento funcional	
Relevancia	ALTA
Especificación	Dentro de su reporte, el usuario debe poder ver una tabla con las medianas mensuales con información del último mes para los 4 parámetros estudiados para cada proveedor de internet al que alguna vez se conectó.

`RF7 - Gráficos del último mes por instalación para reporte del usuario`

Requerimiento funcional	
Relevancia	ALTA
Especificación	Dentro de su reporte, el usuario debe poder ver gráficos con la información agregada del último mes para los 4 parámetros estudiados para cada una de sus instalaciones.

`RF8 - Envío de reportes de usuario mensualmente`

Requerimiento funcional	
Relevancia	ALTA
Especificación	Mensualmente el sistema debe enviar por correo electrónico una versión en formato *PDF* del su reporte a cada uno de los usuarios utilizando el sistema.

`RF9 - Obtención de las mediciones del último mes por parte del administrador`

Requerimiento funcional	
Relevancia	ALTA
Especificación	El administrador debe poder descargar un archivo *CSV* con los datos el último mes del sistema.

Servidor

`RF10 - Garantizar una alta disponibilidad del servidor de mediciones`

Requerimiento funcional	
Relevancia	ALTA
Especificación	Es crítico que el servidor de mediciones se encuentre disponible la mayor cantidad de tiempo posible, garantizando así la mayor cantidad posible de mediciones.

Ejecutables

`RF11 - Crear un ejecutable que corra en Linux y Mac`

Requerimiento funcional	
Relevancia	ALTA
Especificación	Es necesario que el proceso de instalación en estas plataformas sea sencillo y directo, sin necesidad de instalar otros paquetes previamente de forma manual.

`RF12 - Permitir actualizaciones automáticas sin intervención del usuario`

Requerimiento funcional	
Relevancia	ALTA
Especificación	El sistema debe soportar actualizaciones automáticas sin intervenciones del usuario, para poder mejorar el sistema con la base de usuarios obtenida en ese momento.

Requerimientos no funcionales

Base de datos

RNF1 - Persistencia de resultados de mediciones

Requerimiento no funcional	
Relevancia	ALTA
Especificación	Se debe poder resguardar la información de la base de datos para evitar que la misma se pierda.

Pruebas de usuario

RNF2 - Realizar primeras pruebas con usuarios finales

Requerimiento no funcional	
Relevancia	ALTA
Especificación	Se deben realizar pruebas con una cantidad relevante de usuarios reales para analizar el correcto funcionamiento de todos los eslabones del sistema en un ambiente de producción.

Dubugging y logs

RNF3 - Ambientes de producción y staging

Requerimiento no funcional	
Relevancia	MEDIA
Especificación	El sistema debe funcionar en dos ambientes, uno de producción y uno de pruebas.

RNF4 - Logs de información relevante

Requerimiento no funcional	
Relevancia	ALTA
Especificación	Deben poder activarse logs para guardar toda la información relevante que pueda contribuir a renonocer el estado del sistema en todo momento de tiempo y posibles fallas.

RNF5 - Servicios para monitorear estado del servidor

Requerimiento no funcional	
Relevancia	ALTA
Especificación	Debe integrarse un servicio que permita controlar cuanta memoria, CPU y otros datos importantes se están utilizando en todo momento de tiempo.

`RNF6 - Servicios para monitorear fallas en el servidor`

Requerimiento no funcional	
Relevancia	ALTA
Especificación	Debe integrarse un servicio que notifique cuando el servidor tenga una falla en su funcionamiento.

3.1.2. Definición de actores

A continuación de describen los actores que entran en juego al interactuar con la aplicación:

Usuario: Es el actor que utiliza el sistema para validad la calidad y la congestión de su enlace de internet.

Administrador: Es el actor que monitorea la información de los distintos usuarios del sistema y el correcto funcionamiento del mismo.

3.1.3. Casos de uso

Reportes

`Acceso y filtrado de la información del reporte de administrador`

Caso de uso	
Trazabilidad	RF1, RF2, RF3 y RF4
Complejidad	MEDIA
Descripción	1. El administrador accede a su panel ingresando su usuario y contraseña. 2. En el panel, el usuario hace *click* en en botón "Ver gráficos de utilización y calidad". 3. El administrador será conducido a la pantalla donde podrá ver su reporte con los histogramas y diagramas de bloque para cada uno de los proveedores utilizados y con la posibilidad de filtrar la información. 4. El administrador filtra la información por fecha seleccionando una fecha de comienzo y una de fin. 5. El administrador filtra la información por día de la semana haciendo *click* en el *checkbox* y seleccionando el día por el que quiere filtrar. 6. El administrador filtra la información por hora del día, seleccionando una hora de comienzo y una hora de fin. 7. El administrador hace *click* en el botón "Filtrar"para concretar el filtrado. 8. La vista de reporte se vuelve a cargar con la información solicitada.

`Acceso a reporte de usuario`

Caso de uso	
Trazabilidad	RF5, RF6 y RF7
Complejidad	MEDIA
Descripción	1. El usuario accede a su panel ingresando su usuario y contraseña. 2. En el panel, el usuario hace *click* en en botón "Reporte de usuario" en la barra lateral izquierda. 3. El usuario es conducido a la pantalla donde puede ver su reporte con la tabla de medianas mensuales para todos los proveedores de internet que ha usado y con gráficos con información agregada del último mes para cada una de sus instalaciones.

`Descarga de datos del sistema`

Caso de uso	
Trazabilidad	RF9
Complejidad	MEDIA
Descripción	1. El administrador accede a su panel ingresando su usuario y contraseña. 2. En el panel, el usuario hace *click* en en botón "Generar csv de congestión y calidad". 3. El administrador puede descargar a través del navegador un archivo *CSV* con la información solicitada.

Distribución de clientes multiplataforma

Distribución de clientes multiplataforma	
Trazabilidad	RF11, RF12
Complejidad	ALTA
Descripción	1. El usuario descarga el instalador para su plataforma desde el sitio web de TiX una vez creada su cuenta. 2. Presionando doble click, el usuario obtiene instrucciones para instalar el aplicativo en su plataforma. Si es Linux, el gestor de paquetes lo ayuda a instalar el paquete, si es Mac, el ejecutable le solicita que lo copie a la carpeta de Aplicaciones. 3. El usuario puede utilizar TiX sin necesidad de instalar otro programa antes o después de descargar TiX. 4. El usuario recibe actualizaciones automáticas sin la necesidad de cualquier tipo de intervención manual.

Capítulo 4

Detalle de implementación

En este capítulo se explicará en detalle la implementación de cada una de las mejoras al sistema incluidas dentro del alcance del proyecto.

4.1. Mediciones de *TIX* con *TCP*

Al comenzar con la extensión del sistema, la comunicación entre el cliente y el servidor, se realiza en la capa de transporte por medio del protocolo *UDP* a través del puerto 80.

El problema pronosticado es que algunos proveedores de internet limitan el paso de paquetes *UDP* priorizando determinados tipos de tráfico. Para evitar este problema se comienza a evaluar la viabilidad de implementar *sockets raw TCP* (es como se le denomina a las implementaciones de sockets *ad-hoc* por así decirlo, donde uno puede modificar los campos del paquete antes de enviarlos) haciendo que se comporten como *UDP*, de manera de poder evitar ser detectados por los reguladores de tráfico de muchos proveedores de internet.

Se evalúan varias posibilidades para su implementación, utilizando directamente los *sockets* provistos por *Python* y utilizando diversas bibliotecas para la manipulación de *sockets*.

La idea es poder detectar cuando se pierde un paquete, detectar cuál es el que se pierde y en vez de pedirlo de nuevo en este caso, seguir pidiendo los siguientes,

y enviar el paquete de confirmación (conocido como *ACK*) del paquete que se perdió, de manera de lograr un comportamiento similar a *UDP*.

La biblioteca que se encuentra para realizar lo anterior se llama *Scapy*. *Scapy* es una poderosa biblioteca para la manipulación de paquetes. Con ella es posible crear y decodificar paquetes dentro de una amplia selección de protocolos, enviarlos dentro del enlace, capturarlos, encontrar la correlación entre un pedido y una respuesta, entre muchas otras cosas. Debido a estas características se considera a *Scapy* como una herramienta fundamental para solucionar la tarea en cuestión.

Finalmente y luego de realizar una amplia variedad de pruebas, se encuentran determinadas limitaciones que, dada la arquitectura y naturaleza del sistema, imposibilitan la realización de las implementaciones.

La limitación fundamental que presenta *Scapy*, que lo hace inviable para el presente trabajo, es que la magnitud y naturaleza de las dependencias que tiene lo hacen inviable para funcionar en algunas plataformas móviles, requisito fundamental para futuras extensiones del sistema. Sumado a lo anterior, *Scapy* necesita funcionar con el usuario *root* (se denomina *root* al usuario privilegiado del sistema que tiene permisos que otros usuarios no poseen), lo que en plataformas de escritorio no es un problema grave, pero sí lo es para algunas plataformas móviles, como puede ser algunas versiones que *Android* que no permiten acceso a *root*.

De esta forma, la portabilidad (condición necesaria para el proyecto) queda imposibilitada.

Sumado a lo anterior se continúan haciendo pruebas y se detecta que finalmente no hay proveedor de internet dentro de los analizados que limite los paquetes *UDP*, por lo que se decide para poder cumplir con el alcance del proyecto, dejar la implementación con *sockets UDP* como inicialmente se pretendía.

4.2. Módulo de Reportes

Tal como se indica en la sección de alcance, una de las finalidades principales del proyecto es realizar un sistema de reportes tanto para el usuario como para el administrador.

Tanto los reportes del usuario como los del administrador pueden ser accedidos desde la aplicación web y pueden ser vistos *online*.

En la secciones posteriores que describe en detalle la naturaleza y el funcionamiento de ambos tipos de reportes.

4.2.1. Introducción

Los reportes fueron realizados en función a las necesidades del usuario y del administrador. Al dar comienzo a la extensión del sistema tanto el usuario como el administrador no cuentan con una forma de acceder a la información presente en el sistema de una forma fácilmente entendible y ordenada.

4.2.2. Reportes del Usuario

El usuario puede acceder a los reportes desde su panel de control (*dashboard*), desde el enlace denominado "Reporte del usuario".

Además de la vista web de los reportes del usuario, los mismos son generados en *PDF* y enviados a los distintos usuarios una vez por mes (con información del mes) por medio de una tarea *cron* en el servidor.

4.2.2.1. Conformación del reporte del usuario

Los reportes del usuario están confirmados por una tabla de las medianas mensuales, y por una serie de gráficos para cada una de las instalaciones que las cuales el usuario es propietario.

Los reportes permiten observar la medianas para cada uno de los cuatro parámetros estudiados (con información del último mes) y para cada uno de los proveedores de internet que el usuario ha utilizado. De esta forma, el usuario puede comparar rápidamente entre los distintos proveedores de internet en función a cada uno de los parámetros estudiados.

Por otro lado, los demás gráficos muestran información del último mes de cada uno de los 4 parámetros medidos en cada momento del tiempo. Los datos del último mes se encuentran totalmente visualizados en el gráfico, lo que permite al

usuario ver el comportamiento de cada uno de los 4 parámetros estudiados para cada proveedor, sin tener que filtrar la información ni ir cambiando de fecha. Si el usuario quisiese analizar la información más profundamente podría ir al su panel de control y hacerlo con mayor detalle.

4.2.2.2. Generación y envío de reportes en *PDF*

Además de la posibilidad que tienen los usuarios de acceder a sus reportes por medio de la aplicación web, el sistema se encarga de generar estos reportes una vez por mes en formato *PDF* y de enviarlos a las casillas de correo de los usuarios.

Para poder realizar lo anterior, el sistema se ocupa de llamar una vez por mes, por medio de un servicio *cron* que está activo en el servidor, a un *script* de *bash* que básicamente obtiene la información de todos los usuarios, genera los reportes con la información del último mes para cada uno, convierte esos reportes a formato *PDF*, y los envía por medio de un servicio a sus casillas.

4.2.3. Reportes del Administrador

Se plantea la realización de reportes del administrador para permitir al mismo acceder de una forma ordenada a la información recolectada de todos los usuarios.

También se le da al mismo la posibilidad de filtrar esa información en base a distintos criterios y de visualizarla discriminando por proveedor de internet en base a dos tipos de gráficos: histogramas y diagramas de cajas (*boxplots*).

El administrador puede acceder a sus reportes, habiendo accedido, por medio de dos botones desde su panel de administración.

Mediante el botón “Ver gráficos de utilización y calidad” puede acceder a histogramas y *boxplots* por proveedor de internet organizados con la información agregada de todos los usuarios.

Al acceder a la pantalla de gráficos de utilización y calidad el administrador puede seleccionar las fecha de comienzo y la fecha de fin en la cual los datos van a ser obtenidos.

Por otro lado, La información en bruto puede ser descargada en forma de archivo *CSV* por medio del botón “Generar csv de utilización y calidad”.

Adicionalmente, el administrador puede filtrar la información por día de semana y seleccionando un rango de horas, lo que le da más versatilidad al momento de analizar la información.

La finalidad de los gráficos es mostrar información agregada de todos los usuarios del sistema de una forma simple y organizada por proveedor de internet. De esta forma, los administradores podrán determinar para cada proveedor de internet cual es su nivel de servicio, tanto de bajada como de subida.

Se presentan 4 histogramas para cada uno de los proveedores de internet que fueron utilizados por los usuarios del sistema. Cada uno de estos histogramas está dividido en 10 rangos e indica para cada rango la cantidad de ocurrencias de todos los usuarios del sistema. De esta forma, se pueden apreciar para cada proveedor de internet cual es el nivel de servicio que efectivamente ofrecen a sus usuarios para cada uno de los 4 parámetros estudiados.

Se presenta un *boxplot* por proveedor de internet que para cada uno de los 4 parámetros medidos (calidad de subida, calidad de bajada, congestión de subida y congestión de bajada), muestra la mediana, máxima, mínimo, primer y tercer cuartil.

Complementariamente a los histogramas, los *boxplot* sirven para determinar cual es la calidad de servicio de los distintos proveedores de internet que los usuarios del sistema están utilizando.

La importancia de este reporte es central en el trabajo en cuestión, ya que mediante los mismos, se puede detectar la calidad de internet de muchos usuarios del sistema y determinar cuantitativamente la realidad del servicio de internet en la Argentina.

4.2.4. Conclusión

Como se puede ver anteriormente, se provee a los usuarios y al administrador de una forma fácil y comprensiva de acceder a la información presente en el sistema.

El anterior es uno de los objetivos centrales, y una de las motivaciones fundamentales del trabajo en cuestión.

Los reportes del usuario, harán que los mismos puedan conocer su situación y puedan comparar la calidad de la conexión entre distintos proveedores, dándole la posibilidad de elegir con más cuidado y con información clara y precisa.

Por otro lado, el administrador podrá analizar la situación argentina y podrá comparar y determinar cual es el proveedor de internet con mejor servicio.

4.3. Distribución de clientes multiplataforma

Otro de los objetivos del proyecto es extender el proceso de *empaquetado*, es decir, el proceso de creación de ejecutables del proyecto para múltiples plataformas, de forma tal que sea más simple su distribución en distintos sistemas operativos.

Adicionalmente, se busca es que el ejecutable tenga un sistema de actualizaciones automáticas, donde no sea necesaria la intervención del usuario.

4.3.1. Distribución multiplataforma

Durante el trabajo anterior se consigue tener un ejecutable funcional en OSX y Linux, pero no se llega a realizar completamente la distribución o *empaquetado* del ejecutable para que el cliente no tenga que instalar ninguna dependencia extra al realizar la instalación.

Esta es una tarea que puede resultar sumamente complicada, ya que en TiX cada sistema operativo puede poseer distintas versiones de la misma biblioteca y las dependencias pueden ser incompatibles. Se tuvo que recurrir al uso de máquinas virtuales con diversas versiones de los sistemas operativos en cuestión para probar si los ejecutables funcionaban correctamente.

El primer enfoque del problema es el de copiar absolutamente todas las dependencias del ejecutable en la carpeta interna mediante el uso de *PyInstaller*, lo cual funciona pero presenta varios problemas.

En Linux, prácticamente es necesario realizar un empaquetado por cada distribución, ya que las bibliotecas instaladas pueden no ser compatibles con las que se agregan al empaquetado. Esta alternativa no es lo suficientemente buena y en base a eso se reciben varios correos electrónicos de usuarios con versiones incompatibles.

En cuanto a *Mac OSX*, las diferencias entre las versiones 10.8, 10.9 y 10.10 hacen que sólo creando un ejecutable para la versión inferior (10.8) se obtenga un ejecutable compatible con todas las versiones.

En Windows no se encuentran diferencias ni problemas, al menos entre Windows XP, Windows 7 y Windows 8.1. Empaquetando en Windows 8.1, los ejecutables funcionan sin problemas ya sea en Windows XP o Windows 7 y visceversa.

Como el primer enfoque resulta ser efectivo para dos de las tres plataformas, y tras el *feedback* de los primeros usuarios de Linux, se logra armar un empaquetado más simple para Linux, el cual, utilizando el sistema de dependencias *aptitude* de *Debian*, marca como dependencias a las bibliotecas que varían según el sistema operativo, dejando sólo las dependencias fundamentales dentro del empaquetado. De esta forma se obtiene un empaquetado mucho más completo para una de las distribuciones más Linux.

4.3.2. Sistema de updates

Durante el desarrollo de TiX aparecen diversos inconvenientes con las diversas versiones internas del ejecutable que van surgiendo a medida que se implementan las diversas funcionalidades. Por ejemplo, algunas versiones traen datos erróneos por el hecho de no ser compatibles y estar desactualizadas.

Por lo anterior, surge la necesidad de incorporar un sistema de actualizaciones automáticas que permitan tener cierto nivel de coordinación entre las versiones de TiX.

Después de una búsqueda e investigación, no se encuentra ningún sistema de código abierto que permita realizar actualizaciones automáticas, multiplataforma, sin intervención del usuario y con la capacidad de ser lo suficientemente extensible como para correr en diversas versiones de Linux, Windows y Mac. Por eso mismo, se decide realizar una implementación propia *ad-hoc* para resolver este problema.

Desde el lado del servidor de actualizaciones, sobre el cual se utilizó el *API* de *releases* de *GitHub*, se suben para cada plataforma los archivos binarios (sin empaquetar) como archivo *zip*, con su correspondiente versión y plataforma claramente marcados.

Desde el cliente, cada día, se compara el *shasum* (un identificador único armado en base a todos los archivos del sistema, compuesto por la suma de los contenidos del paquete) actual en el cliente con el del servidor. Si el mismo no cambia, no se hace nada. En cambio, si el mismo es diferente, significa que hubo una actualización y se descarga el paquete, se instala y se guarda nuevamente el *shasum* del paquete para así esperar a la próxima versión.

Este *script* se encuentra desarrollado para Linux y Mac, y es fácilmente portable para Windows.

4.4. Debugging, Logs y Monitoreo

Uno de los objetivos que se presenta en la etapa previa a las pruebas de usuario, es la de poder realizar un *debugging* (corrección de *bugs* o errores del programa), del código *Python*, ya que se encuentran diversos errores en la etapa de procesamiento.

Ya que los errores suceden de forma muy esporádica y son difíciles de repetir, se toman varias acciones hasta resolver correctamente el problema.

Inicialmente sólo se sabe que, de forma aleatoria, el código del procesamiento de las mediciones falla, y el proceso de cálculo deja de funcionar. Con lo cual la primera acción a realizar es la de garantizar una alta disponibilidad del servidor de mediciones.

A continuación se aumenta la cantidad de logs, agregando logs en las distintas etapas del cálculo de variables, y revisando cada ocurrencia de errores.

Por otro lado, se agregan dos servicios comerciales de monitoreo de servidores, que en sus planes básicos proveen suficiente información para resolver la mayoría de los errores que se presentan.

Rollbar notifica por la ocurrencia de errores inesperados en el servidor, permitiendo resolver *bugs* de forma rápida, viendo directamente la línea de código y las variables que intervienen en el error.

New Relic alerta si el servidor tiene algún inconveniente, ya sea por uso de *CPU*, memoria, disco o capacidad excesivos, directamente al celular desde su aplicación móvil o vía correo electrónico.

El error más grave que se encuentra es uno que causa un *segmentation fault* (error de acceso de memoria de bajo nivel, es decir, accesos a sectores de memoria no permitidos por el sistema operativo, algo muy raro en un entorno *Python*, donde, en teoría, ese tipo de error no debería presentarse) en la etapa de procesamiento de las mediciones y requiere el uso de *gdb* (una herramienta de *debugging* histórica de linux) mediante un paquete especial de *Python* (*python-dbg* una versión de *debugging* del lenguaje compatible con *gdb*) para obtener la línea donde se produce el error. Tras encontrar la línea, se encuentra que la biblioteca *numpy* no provee una función de cálculo de cuadrados mínimos que sea *threadsafe* (compatible con un procesamiento de múltiples *threads*, es decir, que pueda ejecutar código en paralelo), con lo cual, se deshabilita el uso de multiples *threads* en la etapa de procesamiento. Esto no causa un impacto en la performance por la existencia del *Global Interpreter Lock* en *Python*, que es parte del mecanismo mediante el cual se interpreta el código *Python*, el cual soporta intrínsicamente sólo un *thread*, por lo que no es posible obtener mayor performance en la parte de procesamientos utilizando múltiples *threads*, y este cambio no afecta a la performance de las mediciones en ese aspecto.

Durante esta etapa de *debugging* y puesta a punto del servidor, también se decidió aplicar varias buenas prácticas para el manejo de servicios con una alta utilización como es TiX. Una de estas prácticas es la utilización de un entorno de *staging*, para realizar pruebas de nuevas versiones antes de subirlas definitivamente al entorno denominado *producción*, que es el cual ven los usuarios finales. Otra práctica que se introduce es la de tener un proceso de *deploy*, es decir, de subida del proyecto, unificado y automatizado mediante el uso de la herramienta *capistrano*, con la cual se sube el código ya sea al entorno de producción o staging, tanto del servidor web como el de procesamiento, y reinicia los servicios correspondientes para que se actualice la versión que se encuentra corriendo en cada entorno. Estos dos cambios permiten subir rápido los cambios al servidor, de forma ordenada y automáticamente documentada por nuestra herramienta de control de versiones, *git*.

4.5. Base de datos

Siendo la base de datos del sistema uno de los pilares fundamentales de su funcionamiento, se considera necesario explicar algunos detalles en cuanto a su estructura y resguardo de información.

4.5.1. Estructura

La base de datos de proyecto se encuentra compuesta por la siguiente estructura:

Tabla ISP	
Nombre de Columna	**Tipo**
id	integer
name	character varying(255) NOT NULL

Esta tabla muestra como es persistida la información para los distintos proveedores de internet. Básicamente guarda un *id* único y el nombre del proveedor.

Tabla INSTALLATION	
Nombre de Columna	**Tipo**
id	integer
encryptionkey	text
location	integer
name	character varying(255) NOT NULL
owner_id	integer

Esta tabla corresponde a cada una de las instalaciones del sistema. Para ello se guarda un *id* único de la instalación, una clave foránea el usuario al que corresponde la instalación, un nombre, una ubicación y una clave de encriptado.

Tabla USERS	
Nombre de Columna	**Tipo**
id	integer NOT NULL
birthyear	integer
nickname	character varying(255) NOT NULL
password	character varying(255) NOT NULL
passwordrecoveryrequestcode	character varying(255)
type	character varying(255) NOT NULL

Como su nombre lo indica, la tabla USERS persiste los datos de los usuarios del sistema. Básicamente se cuenta con un *id* único, una fecha de nacimiento, un nombre de usuario, una contraseña, una clave para la recuperación de contraseña y un tipo de usuario, que puede ser administrador o usuario regular.

Tabla USERS_INSTALLATION	
Nombre de Columna	**Tipo**
users_id	integer NOT NULL
installations_id	integer NOT NULL

Tabla RECORDS	
Nombre de Columna	**Tipo**
id	integer NOT NULL DEFAULT
h_rs_down	real NOT NULL
h_rs_up	real NOT NULL
h_wave_down	real NOT NULL
h_wave_up	real NOT NULL
calidad_down	real NOT NULL
calidad_up	real NOT NULL
timestamp	timestamp without time zone NOT NULL
userdowncongestion	boolean NOT NULL DEFAULT FALSE
userupcongestion	boolean NOT NULL DEFAULT FALSE
utiliz_down	real NOT NULL
utiliz_up	real NOT NULL
installation_id	integer
isp_id	integer
user_id	integer

Finamente la tabla RECORDS, guarda la información de cada uno de los registros que se generan con la información obtenida de las distintas insolaciones de los usuarios. De esa forma, se guardan claves foráneas al usuario, el proveedor y la instalación en cuestión, así como de cada uno de los parámetros medidos y un *timestamp* para determinar de forma unívoca el momento de registro del dato.

4.5.2. Backups automáticos

Para la realización de backups y su posterior almacenamiento, se utiliza una biblioteca de *ruby* llamada *backup* que se encarga de resolver el problema mediante un archivo de configuración muy simple, el cual utilizando también *cron*, realiza un backup completo de la base de datos *PostgreSQL*. Los mismos son almacenados en un archivo comprimido dentro del servidor de mediciones.

4.6. Pruebas de usuario

Realizar correctamente pruebas con usuarios reales para todos los requerimientos del sistema en un ambiente de producción es uno de los requerimientos no funcionales primordiales de presente estudio. A continuación se describen algunas de las incumbencias principales con las que se encuentran en esta etapa del estudio.

4.6.1. Obtención de usuarios

En el momento dónde se valida empíricamente que el sistema ya funciona correctamente y está en una versión estable, se decide dar comienzo a las pruebas de usuario.

Para ello, uno de los puntos centrales es obtener usuarios interesados en probar el sistema y en el problema que el proyecto soluciona.

Con el objetivo de garantizar la anterior, se realiza una serie de envíos de correo electrónico a personas involucradas en el ámbito académico (profesores y estudiantes) instando a que prueben el sistema y provean su retroalimentación.

4.6.2. Cuenta de soporte y principales problemas

Para el envío de las distintas cadenas de correo electrónico y para dar soporte a los usuarios se crea una cuenta de correo electrónico exclusiva de TiX.

A través de la misma se mantiene el contacto directo con los distintos usuarios y se los insta a que instalen y prueben la instalación.

A su vez, a lo largo el proceso de prueba se recibe una gran cantidad de retroalimentación por parte de los usuarios.

Esa retroalimentación concierne tanto temas de experiencia del usuario, como temas de diseño y funcionalidad. Por ejemplo, fue muy corriente que en ciertas distribuciones y versiones de Linux el empaquetado no funcionase correctamente, lo que hizo dar cuenta de la dificultad yacente en hacer un sistema cien por ciento multiplataforma.

Otros problemas que suceden, están relacionados a requerimientos implementados anteriormente que en un ambiente de producción no funcionan correctamente y fueron solucionados, como fue el caso de la recuperación de contraseña por parte de los usuarios.

4.6.3. Conclusión

El desarrollo de las pruebas de usuarios anteriormente descripto significa un proceso en el que se descubren y se dan solución a gran cantidad de problemas yacentes en el sistema.

Además de los problemas mencionados anteriormente, se encuentran problemas de infraestructura; tanto desde el punto de vista de realizar el control de la memoria y capacidad de procesamiento ocupada del servidor, hasta darse cuenta que por la cantidad de usuarios usando el sistema y por la longitud de los logs que se guardan hay que liberar espacio en el disco con mayor frecuencia.

Otro punto sumamente interesante es que una vez validado que las mediciones eran acordes con la realidad y a medida que el tiempo fue pasando la cantidad de datos disponible de cada proveedor de internet fue creciendo y esto permite ver aproximadamente (y cada vez con menos error) cuales son los valores de los 4

parámetros estudiados para cada uno de los proveedores de internet usados por los usuarios del sistema.

Así se puede ver, por ejemplo, la calidad de bajada de determinado proveedor está entre 90 y 100 porciento en la gran mayoría de los casos, mientras que el mismo parámetro para otro proveedor presentar una distribución mucho menos favorable para la calidad de servicio que promete.

Capítulo 5

Resultados

Sin duda el presente trabajo trajo consigo una gran cantidad de significativos resultados y mejoras. Sin embargo, se ha decidido hacer foco en los resultados centrales, que debido a su magnitud engloban a muchos de los resultados más particulares.

Se considera importante empezar por el resultado que se considera central, el de poder haber reunido a una considerable cantidad de usuarios y haber probado el sistema en ambiente en producción obteniendo mediciones acordes con la realidad y de gran valor tanto académico como comercial.

A su vez, se reconoce que el valor de esos resultados va a ir aumentando a medida de que avance el tiempo, ya que mientras más información de más usuarios se cuente en el sistema, tanto mejor será la precisión y la amplitud de los mismos.

Para poder evidenciar el resultado anterior, es central contar con el sistema de reportes de administrador que es producto del presente trabajo.

El sistema de reportes en cuestión garantiza el análisis de toda la información del sistema de una forma clara y precisa, dando gran facilidad de filtrado para un análisis más exhaustivo. También permite comparar rápidamente la calidad de servicio de diferentes proveedores de internet para cada uno de los parámetros estudiados, lo que sin duda también constituye un resultado de gran utilidad para el análisis del problema en cuestión.

Dejando de lado momentáneamente al resultado anterior, otro efecto más que interesante es la posibilidad que se le da a los usuarios de poder acceder rápidamente a la información del sistema (a través de la tabla de medianas) y a la información agregada del último mes para cada una de sus instalaciones, y proveerla en la comodidad de su casilla de correo en formato *PDF*, sin que siquiera el usuario tenga que acceder a la aplicación web. Esto le da al usuario otra herramienta para controlar a sus proveedores de internet y para entender con datos reales cual es la mejor opción para el servicio que realmente esperan.

Como tercer resultado significativo, se destaca la probada portabilidad que se le dio al sistema al haber creado un sistema de distribución eficaz, multiplataforma que permite realizar actualizaciones automáticas. Este resultado sin duda contribuye enormemente a la experiencia de usuario del sistema y por ende a su basta adopción y crecimiento, lo que en este momento del ciclo del vida del producto debe posicionarse como un punto más que destacado.

Finalmente, como cuatro resultado, pero no menos importante, se destaca la solidez alcanzada de la corriente versión, que se alcanza no sólo por la continua solución de problemas de la plataforma, ni por el contacto directo con usuarios reales, sino por las facilidades de logueo, resguardo de información y monitoreo alcanzadas en la presente versión y que hacen de TiX un proyecto mucho más mantenible y escalable.

Llegar a esta solidez sin duda no fue un camino sencillo, ya que la complejidad del sistema, de naturaleza distribuido y multiplataforma, hace que sea difícil introducir cambios o modificaciones con velocidad, y requiere tener en cuenta múltiples aspectos antes de dar cualquier paso o agregar una nueva funcionalidad. El hecho de que los clientes tengan que instalar en su computadora el software, la necesidad de actualizar ese software, la portabilidad del código y el bajo uso de recursos son todos aspectos que siempre hay que tener en cuenta.

5.1. Mediciones de distintos proveedores

En las figuras presentadas a continuación, se muestran los resultados de las mediciones agregadas de los distintos usuarios para cada proveedor.

Se puede ver claramente que *Cablevisión S.A.* es quien brinda en términos relativos una calidad mayor con respecto a los otros proveedores, ya que tanto en términos de subida como bajada su calidad se orienta más al 100 %. Por el lado de *Telecom S.A.* puede verse que se encuentra en la posición más desfavorable con respecto a la calidad de su subida. Muy probablemente esto se deba a que la gran mayoría de su infraestructura es telefónica, por lo que está atada a la tecnología *ADSL*, la cual tiene limitaciones de capacidad. Sin embargo Telefónica de Argentina S.A. tiene ese mismo tipo de infraestructura, con lo cual se cree que evidentemente la infraestructura de *Telecom* no se encuentra propiamente actualizada, o se utiliza en un porcentaje más bajo. Por otro lado, se ve que *Cablevisión S.A.* gracias a su implementación basada en cablemódem (*DOCSIS 2.0 y 3.0* la cual es más moderna que el *ADSL* de *Telefónica* y *Telecom*), tiene una más que excelente calidad de bajada, especialmente considerando que brinda velocidades superiores, de hasta 30mbps en sus planes de*Fibertel*, a comparación de los 10mbps que brindan los competidores con *ADSL*.

Histogramas

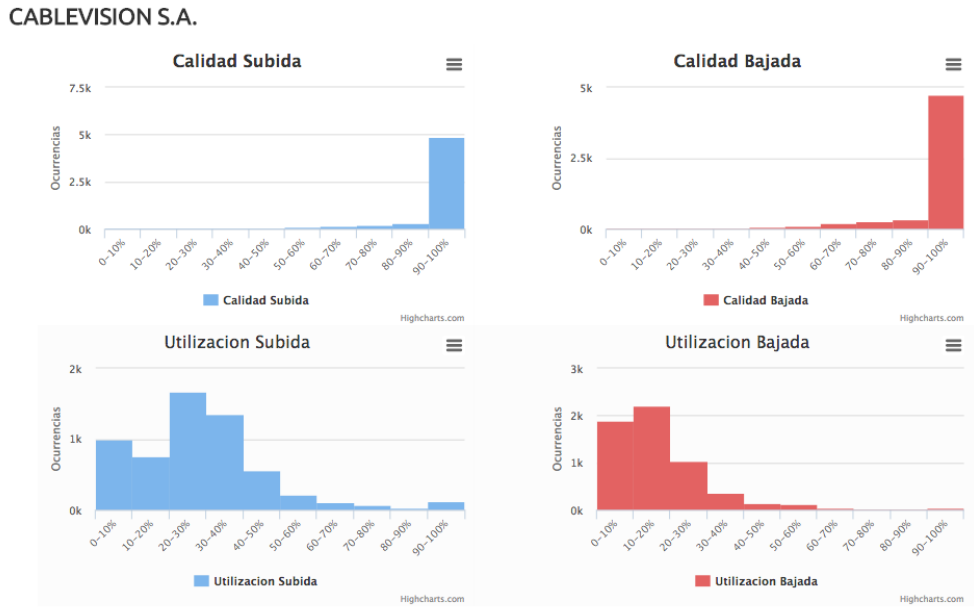

FIGURA 5.1: Mediciones de Cablevisión S.A

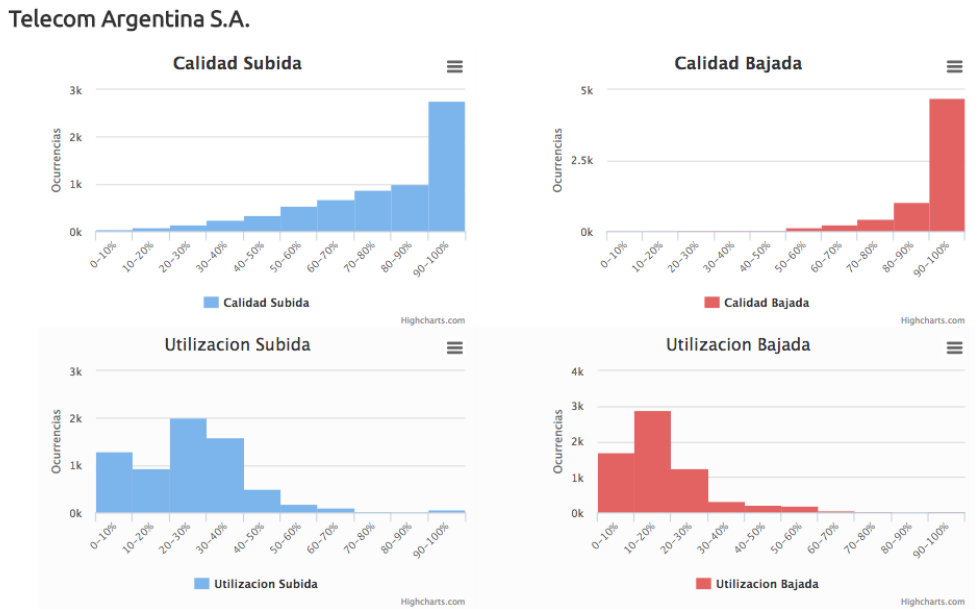

FIGURA 5.2: Mediciones de Telecom Argentina S.A

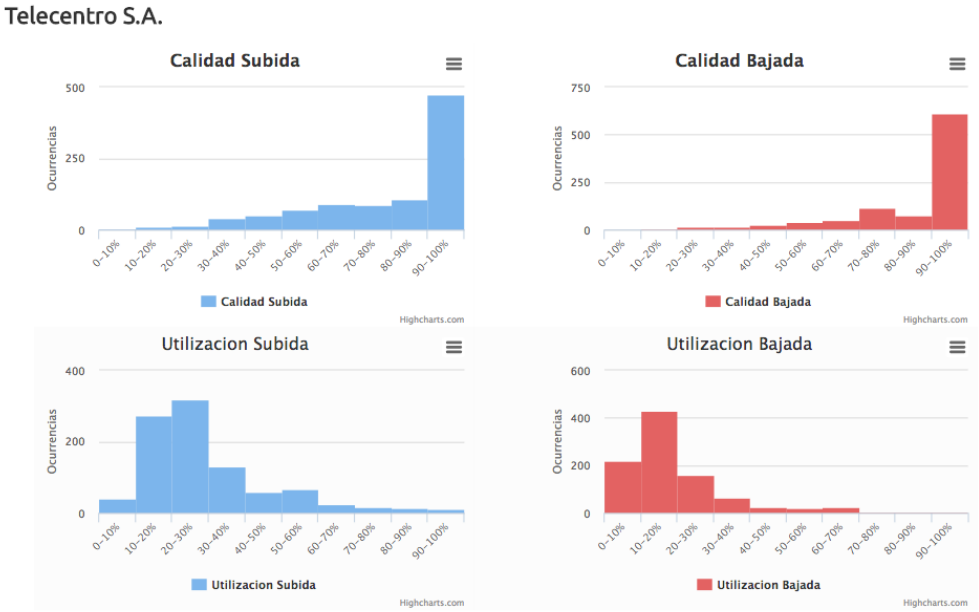

FIGURA 5.3: Mediciones de Telecentro S.A

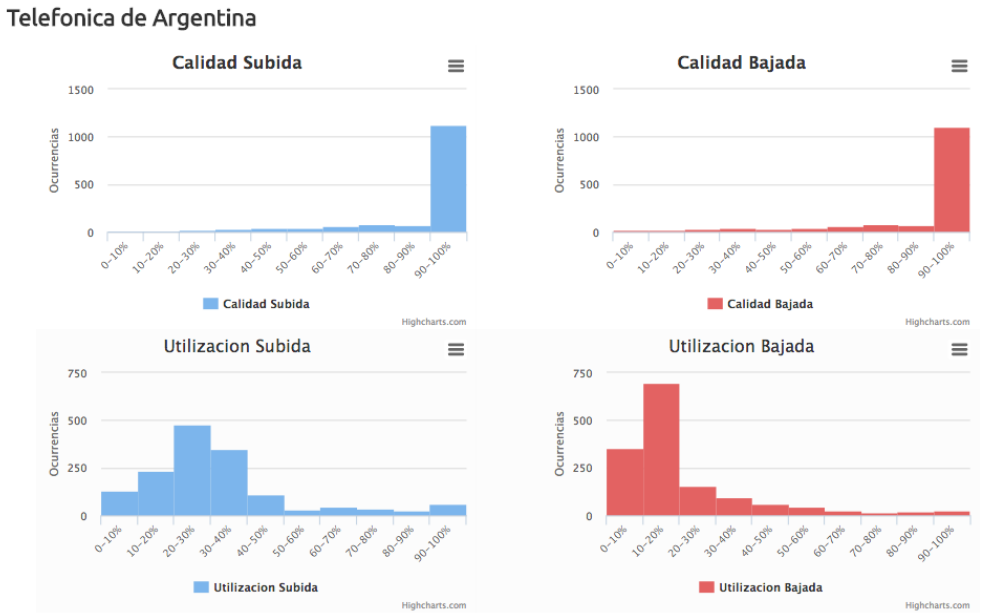

FIGURA 5.4: Mediciones de Telefónica de Argentina S.A

Diagramas de bloques

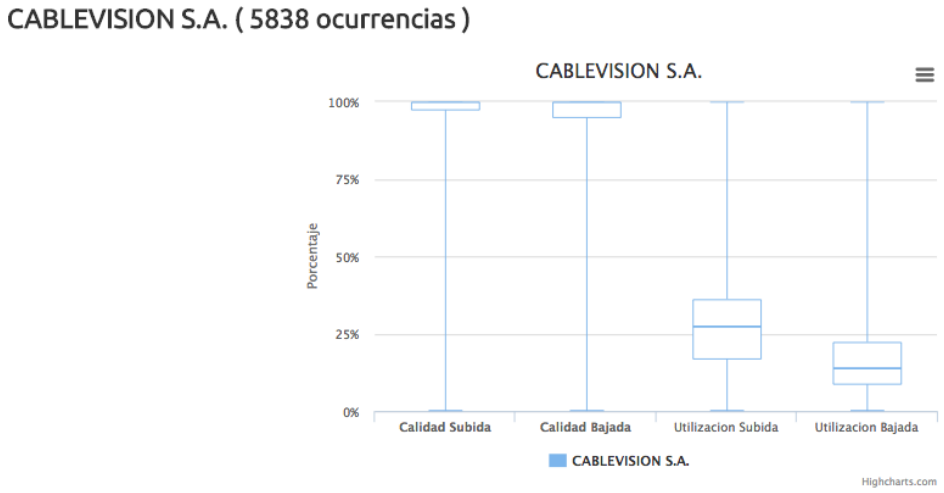

FIGURA 5.5: Diagrama de bloques de Cablevisión S.A

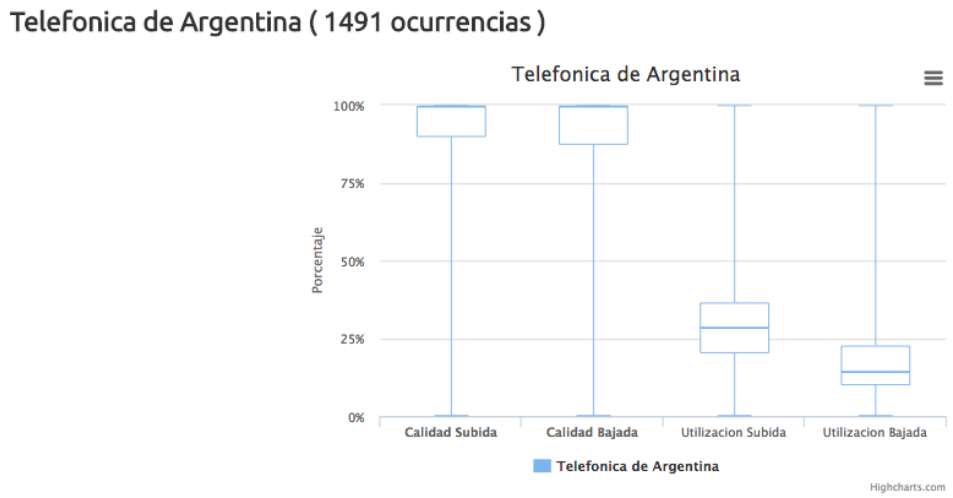

FIGURA 5.6: Diagrama de bloques de Telefónica S.A

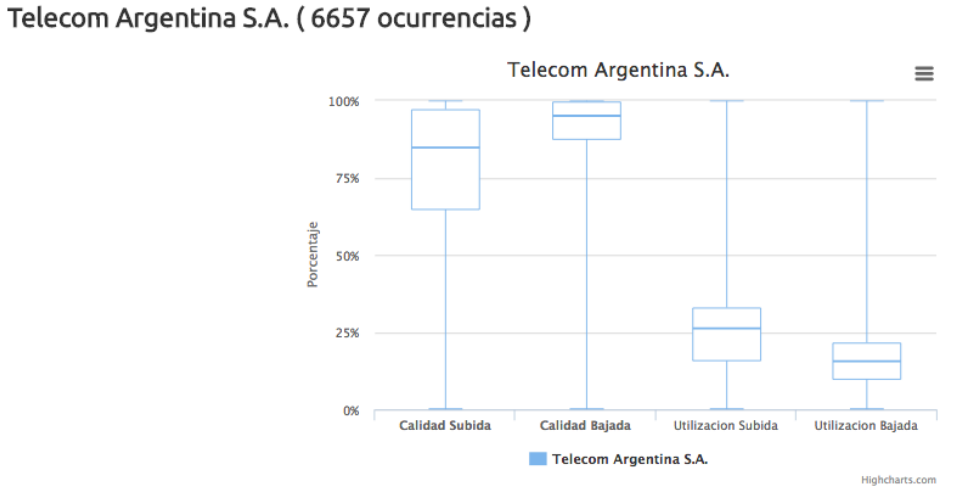

FIGURA 5.7: Diagrama de bloques de Telecom S.A

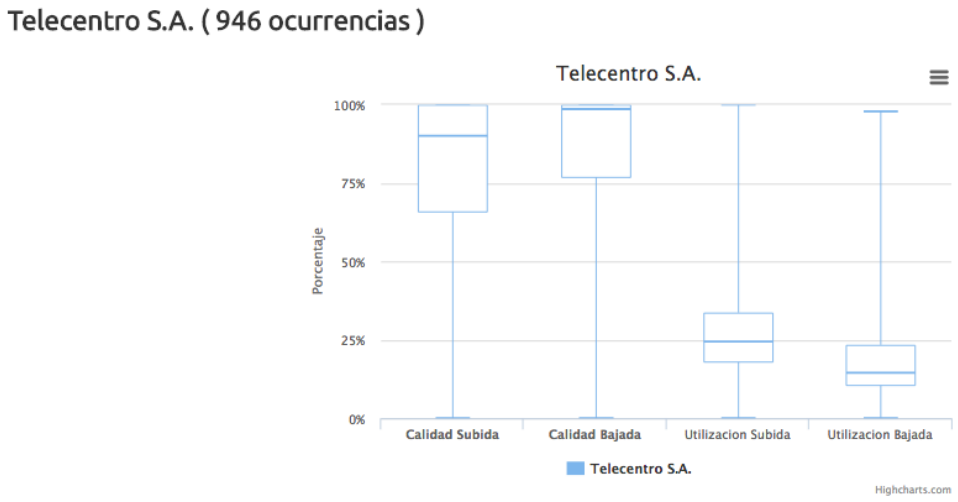

FIGURA 5.8: Diagrama de bloques de Telecentro S.A

Capítulo 6

Documentación Técnica

6.1. Plan de avance

- Etapa preliminar
  - Introducción a la plataforma ya desarrollada.
  - Definición de objetivos iniciales.
- Etapa 1
  - Investigación de *sockets UDP* y *TCP*.
- Etapa 2
  - Implementación de consultas para agrupar ISPs.
  - Empaquetado en Mac.
- Etapa 3
  - Implementación de reportes Web.
  - Empaquetado en Linux.
- Etapa 4
  - Implementación de reportes vía email.
  - Instalación del empaquetado en Linux.
- Etapa 5

  - Puesta a punto del servidor.
  - Pruebas con usuarios finales.

- Documentación y presentación
  - Elaboración de este informe y documentación.
  - Presentación.

6.2. Herramientas y plataformas utilizadas

Linux (www.linux.org) es el sistema operativo donde corre el servidor web, el de base de datos y el servidor de mediciones. Es el proyecto de código abierto más importante de mundo y corre en prácticamente cualquier plataforma, tiene variedad de distribuciones, es decir, variantes, entre las cuales se encuentra *Ubuntu*. Esta es la distribución que se utiliza en su variante de *servidor*, ya que tiene una amplia variedad de paquetes disponibles con fácil instalación y un bajo uso de recursos.

MacOSX (www.apple.com/es/osx) es el nombre del sistema operativo utilizado en las computadoras *Macintosh* de *Apple*. Son computadoras ampliamente utilizadas en el ámbito de IT y académico al cual apuntamos, por lo cual se decide realizar un ejecutable que funcione en las últimas versiones de este sistema.

Java (www.java.com) es un lenguaje de programación fuertemente tipado y compilado, que brinda una amplia portabilidad y muy buena performance, y es utilizado junto con *Spring* y otras bibliotecas para mostrar el sitio web de TiX y toda la plataforma web interna en el servidor web.

Python (www.python.org) es un lenguaje de programación interpretado y de tipado dinámico, también de una muy buena portabilidad pero muchas veces inferior a la de java. Es utilizado para los cálculos hechos en el servidor de mediciones por sus excelentes bibliotecas de análisis numérico como *scypy* y *numpy*. Python a diferencia de Java, tiene un uso de memoria bastante bajo.

PostgreSQL (www.postgresql.org.es) es probablemente el motor de bases de datos relacional más completo que hay. A diferencia de otros proyectos similares como *MySQL*, es mantenido totalmente por la comunidad abierta *PostgreSQL*

Global Development Group. Por defecto viene configurado para realizar escrituras directamente a disco, soportando fallas mediante un *log* de escrituras, pero muchísimos parámetros pueden ser optimizados de esta base de datos pueden ser optimizados para tolerar una amplia cantidad de usuarios con muchas escrituras y pocas escrituras como es el caso de TiX.

Spring (www.spring.io) es uno de los frameworks MVC más populares en el la comunidad del lenguaje java. Es ámpliamente modular y en conjunto con otras herramientas como *Hibernate* permite un desarrollo ágil y rápido.

Hibernate (www.hibernate.org) es un *ORM* (Object Relational Mapper), es decir, una biblioteca que permite relacionar objetos de dominio con tablas de bases de datos relacionales, brindando transacciones, consultas parametrizadas y muchas otras ventajas, utilizando el lenguaje de programación java en vez de comandos SQL, permitiendo así tener un código más robusto y menos dependiente de la base de datos.

Tomcat (tomcat.apache.org) es un contenedor de aplicaciones web java que utiliza el concepto de *contenedor*, es decir, un sólo archivo que contiene toda la aplicación web, esto facilita mucho el proceso de deployment, ya que sólo es necesario cambiar el archivo contenedor, que tiene la extensión *.war*

Nginx (nginx.org) es un servidor de aplicaciones web/proxy reverso de alta performance que se utiliza para reducir la carga del servidor tomcat, brindandole mejoras de performance mediante el cacheo de algunos pedidos al servidor, y redirijiendo los puertos según sea necesario.

Highcharts (http://www.highcharts.com/) es la biblioteca que se utiliza para realizar las gráficas de las mediciones del usuario y los reportes con velocidades por ISP. Si bien es una biblioteca comercial, su licencia permite su uso para proyectos de código abierto.

Bootstrap (getbootstrap.com) es un framework de css creado por *Twitter* para facilitar el desarrollo de aplicaciones web con html5/css3. Permite a los desarrolladores realizar el diseño básico de formularios, menúes y diversos componentes de la aplicación, para que luego un diseñador pueda adaptar ese diseño básico al diseño final de la plataforma.

wkhtmltopdf (wkhtmltopdf.org) es, como su nombre lo dice, un motor de *Webkit* (El motor detrás de *Safari*, *Konqueror*, *Opera*, y muchos otros navegadores web, es el antecesor del motor *Blink* de *Google Chrome*, que es una versión más avanzada de *Webkit*), que muestra contenido html, junto con su css y html, y lo convierte finalmente a *PDF*, se utiliza esta herramienta para desarrollar todo el sistema de reportes que se envían por correo electrónico.

Capítulo 7

Conclusiones

En esta capítulo se pretende explicar los conclusiones centrales que se obtuvieron en el presente trabajo, así como las principales dificultades encontradas, los aprendizajes más provechosos y las posibles extensiones del sistema.

7.1. Aprendizajes

El presente estudio trajo consigo una gran cantidad de aprendizajes, fundamentalmente por que claramente solicitó la aplicación de todos los conocimientos adquiridos durante la carrera.

Desde la programación de una aplicación web en *JAVA* hasta el análisis de errores de concurrencia, desde la implementación de buenas prácticas de *HCI* (interacción hombre computadora), hasta el análisis y manipulación de protocolos de capa de transporte, desde la el manejo de dependencias y la creación de empaquetados, hasta el análisis y corrección de errores en *GDB*; realmente la realización del trabajo constituyo una recorrida por los contenidos por toda la carrera y una integración completa de todos los conocimientos adquiridos a lo largo de la misma.

Y sin lugar a duda, en esta integración subyace el más grande aprendizaje, ya que allí mismo es donde uno entiende la complejidad y la conjunción de habilidades que un proyecto de importante magnitud requiere.

Se termina este trabajo, con una gran satisfacción de saber que como futuro ingeniero informático se puede resolver integralmente problemas de gran complejidad y resolver cualquier tipo de desafío técnico.

7.2. Extensiones al Sistema

Son muchas las posibles extensiones del sistema. Entre ellas se destacan:

La total conversión de TiX en un proyecto *open source*, para permitir de la comunidad libre pueda contribuir a su crecimiento y adopción.

La adopción de un *pool* de procesos en vez de un *pool* de *threads* en el servidor de mediciones, para poder aprovechar los cuatro núcleos de procesamiento que posee el servidor en vez de utilizar uno sólo.

La realización de TiX *Mobile*, idealmente para *iOS*, *Android* y *Windows Phone*.

La realización de los cálculos estadísticos dentro del cliente y no centralizado en un servidor, reduciendo así los costos del mismo.

Permitir al cliente observar sus propios datos de forma local, incluso si no se encuentra conectado a internet (caso de uso muy común en plataformas móviles).

Apéndice A

Problemas encontrados y soluciones

A lo largo de la realización del presente trabajo, no todo fue resultados y soluciones, sino que hubo gran cantidad de dificultades y problemas que complejizaron la correcta concreción del mismo. Se detallarán ellos en las siguientes secciones.

A.1. Concurrencia en el servidor de mediciones

Uno los los errores más complicados que surgieron fue un problema de concurrencia en la función que realizaba el cómputo de la información. Entre problema impactó negativamente la primera parte de los pruebas con usuarios finales, hasta su solución.

El problema generaba caídas en el servidor de medición durante el cómputo más importante de la información obtenida, y resultó difícil de resolver por la naturaleza del error.

Tras realizar diversos cambios y pruebas se terminó encontrando que el problema radicaba en una biblioteca de *Python*, precisamente *Scipy*, la cual no permite realizar cálculos de forma concurrente, ya que su implementación nativa en *C* no es *thread safe*(es decir que no está pendasa para correr automácamente).

Encontrar y reproducir inequívocamente esta condición de carrera dentro del código en producción resultó ser imposible, por lo cual tras plantear la hipótesis

de que esta biblioteca podría ser la responsable del error *segmentation fault*, se decidió hacer una prueba aislando los cálculos, concluyendo que no fallaban al correr individualmente.

Al tener un *thead pool*(conjunto de threads fijos que se van asignando a medida que se necesitan) *threads* con 10 threads, e intentar correr 10.000 tareas iguales, rápidamente este programa falla en cualquier plataforma con la última versión de *Scipy*.

Para resolver el inconveniente, se cambió la cantidad de *threads* del *pool* a 1, garantizando así que todos los pedidos se ejecuten en serie, sin ningún tipo de ejecución en paralelo.

Este error tomó mucho tiempo de detección de errores y análisis de código, ya que los errores de este tipo son realmente poco comunes en *Python*, y se tuvo que recompilar todas las bibliotecas para poder analizar con *Gdb* (programa para detección de errores de *C*) la ejecución paso por paso. Esto último, sumado al uso de *dumps*(logs) del kernel, ayudó a suponer que algo estaba pasando con la biblioteca *Scipy*.

A.2. Manejo de threads y uso de memoria

Otro problema importante encontrado tras realizare las pruebas de usuario fue el del uso de memoria de forma excesiva. Esto se ve causado por la creación de un *thread* por cada paquete *UDP* recibido, incluso cuando no era necesario crear ese thread (ya que era descartado por una comparación posterior). Esto, combinado con algunos cálculos que requerían más tiempo que otros, daban como resultado un incremento en el uso de la memoria del servidor.

Inicialmente este problema resultaba ser muy esporádico, pero al incrementar la cantidad de usuarios, inmediatamente se comenzó a tener alertas por parte de *New Relic* (la herramienta de monitoreo), indicando que el uso de memoria subía, al punto de que el servidor se quedaba sin memoria, y empezaba a usar el disco como *swap*. Esto agudizaba el problema, ya que el swapping reducía la *performance* de los cálculos, y terminaba dejando al servidor con cada vez más threads pendientes, y procesando cada vez más lento.

La solución a este problema fue simplemente no crear los threads hasta que las mediciones estaban totalmente listas para ser calculadas, al principio parecía una optimización innecesaria pero se tornó en algo fundamental, ya que el uso de memoria se tornó mucho menor, y nunca volvimos a tener inconvenientes con la memoria.

A.3. Verbosidad de logs y uso de disco

Al realizar debugging se incrementa el tamaño de los logs varios cientos de veces, con lo cual fue normal tener durante un par de semanas caídas del servidor por falta de disco.

Una vez resueltos los bugs se volvió a reducir la verbosidad de los logs, se configuran las alertas correspondientes en *New Relic* para un uso mayor al 80 % del disco.

Apéndice B

Manual de instalación

B.1. Descarga del código

Para descargar el código es necesario utilizar *git*. El siguiente comando permite descargar el código del proyecto.

```
git clone git@github.com:joseignaciosg/TiX.git
```

B.2. Deployment del proyecto

Para realizar el *deployment* del proyecto, es necesario tener las credenciales propiamente configuradas en el servidor de TiX (clave pública *SSH* y permisos para *pushear* al repositorio *git*).

Una vez obtenido lo anterior, se deben correr los siguientes comandos.

Por única vez se debe instalar *Ruby*, y correr.

```
gem install bundler
bundle install --local
```

Luego, ya es posible ejecutar el siguiente comando para realizar un *deployment*.

```
cap production deploy
```

Todos los cambios que se hayan subido en la rama *master* del proyecto serán subidos al servidor. Se actualizará *Tomcat* en caso de ser necesario, y se reiniciará el servidor de mediciones.

B.3. Realización de empaquetado

Para realizar el empaquetado primero es necesario descargar *Kivy*, ya sea para Linux o Mac del sitio http://kivy.org/

Luego es necesario ejecutar por única vez desde la base del repositorio git:

```
scripts/osx_full_install.sh # para OSX
scripts/linux_full_install.sh # para Linux
```

Nota: todos los *scripts* se encuentran en la sección B.6, anexo.

A continuación se debe correr para realizar un empaquetado desde la base del repositorio git:

```
scripts/package_osx.sh # para OSX
scripts/package_linux.sh # para Linux
```

El resultado del empaquetado se encontrará en la carpeta *dist/* del proyecto, como archivo *.deb* en Linux y *.dmg* en Mac.

B.4. Realización de update

Para realizar una actualización es necesario utilizar el repositorio *Github* original del proyecto como *remote origin* en *git*, ya que esto es una condición que asume el script que genera los updates.

Para generar un update en la plataforma en la que se encuentre se debe correr el siguiente comando desde la base del repositorio git:

```
updater/tix_release_version.sh
```

Luego, para subir el release, debe ejecutarse desde la base del repositorio git:

```
git push origin releases
```

B.5. Realización de backups en la base de datos

Para realizar un backup es necesario acceder por ssh al servidor, con el usuario *pfitba*. Una vez que se tenga el acceso, se debe acceder a la carpeta *Backup/*

Para correr un backup individual se debe correr el siguiente comando desde la base del repositorio git:

```
backup perform -t backup
```

Para revisar el *crontab* con el cual se ejecutan los backups debe editarse el archivo *Backup/config/schedule.rb*, el mismo contiene la configuración de las tareas cron a correr cada día.

El crontab se actualiza corriendo el siguiente comando, el cual ejecuta la configuración del backup, que figura el anexo B.6:

```
whenever --update-crontab
```

B.6. Anexo - Scripts

osx_full_install.sh

```
\#!/bin/bash

# NOTE: Install kivy!

sudo easy_install pycrypto
sudo pip install pyasn1==0.1.7
sudo pip install wsgiref==0.1.2
sudo pip install pyinstaller
sudo pip install requests
sudo pip install rsa
```

linux_full_install.sh

```
./prepare_linux.sh
sudo apt-get install python-pip
sudo apt-get install python-dev
sudo apt-get install build-essential
sudo apt-get install libpython-dev
sudo pip install -r dependencies.txt
sudo pip install pyinstaller
sudo apt-get install rubygems
sudo gem install fpm
./package_linux.sh
sudo dpkg -i tix_0.1_i386.deb
```

package_linux.sh

```
#!/bin/bash
pyinstaller TiX/Tix.linux.spec
cd dist/TixApp
fpm -n tix_project -s dir -t deb -v 0.1 \
        --depends libgcc1 \
        --depends libsdl-ttf2.0-0 \
        --depends libmad0 \
        --depends libjson0 \
        --depends libxrender1 \
        --depends gstreamer0.10-alsa \
        --depends libcdparanoia0 \
        --depends libslang2 \
        --depends libvorbisenc2 \
        --depends libasyncns0 \
        --depends libgupnp-igd-1.0-4  \
        --depends libncursesw5  \
        --depends libshout3  \
        --depends libgupnp-1.0-4  \
        --depends libxau6  \
        --depends gksu \
        --depends libxext6  \
        --depends libsndfile1  \
        --depends libxfixes3  \
        --depends libiec61883-0  \
        --depends libvorbis0a  \
        --depends libdbus-glib-1-2  \
        --depends libncurses5  \
        --depends libcaca0  \
        --depends libgpg-error0  \
        --depends libreadline6  \
        --depends libfontconfig1  \
        --depends libglib2.0-0  \
        --depends libssl1.0.0  \
        --depends libpng12-0  \
        --depends libsmpeg0  \
        --depends libnice10  \
        --depends libxcb-shm0  \
```

```
--depends libvisual-0.4-0  \
--depends libsdl-mixer1.2  \
--depends libgdk-pixbuf2.0-0  \
--depends libselinux1  \
--depends libwrap0  \
--depends libsqlite3-0  \
--depends libgstreamer-plugins-base0.10-0  \
--depends libgnome-keyring0  \
--depends libpcre3  \
--depends libaa1  \
--depends gstreamer0.10-x  \
--depends libbz2-1.0  \
--depends libjack-jackd2-0  \
--depends gstreamer0.10-pulseaudio  \
--depends libv4l-0  \
--depends libavc1394-0  \
--depends libwavpack1  \
--depends libxv1  \
--depends libcairo-gobject2  \
--depends libmtdev1  \
--depends libflac8  \
--depends libxcb1  \
--depends libdbus-1-3  \
--depends libraw1394-11  \
--depends libgcrypt11  \
--depends gstreamer0.10-plugins-good  \
--depends libxdamage1  \
--depends libffi6  \
--depends libquadmath0  \
--depends libgfortran3  \
--depends libcairo2  \
--depends libspeex1  \
--depends libv4lconvert0  \
--depends libfarstream-0.1-0  \
--depends bluez-gstreamer  \
--depends libxcb-render0  \
--depends libdv4  \
--depends libxdmcp6  \
--depends gstreamer0.10-plugins-base  \
--depends libpixman-1-0  \
--depends libxml2  \
--depends libfreetype6  \
--depends libgudev-1.0-0  \
--depends libasound2  \
--depends libjpeg-turbo8  \
--depends liborc-0.4-0  \
--depends libtag1-vanilla  \
--depends libtinfo5  \
--depends libgconf-2-4  \
--depends gstreamer0.10-gconf  \
--depends libgirepository-1.0-1  \
--depends libstdc++6  \
```

```
        --depends libtheora0  \
        --depends libx11-6  \
        --depends libogg0  \
        --depends libgpm2  \
        --depends libgstreamer0.10-0  \
        --depends zlib1g  \
        --depends libc6  \
        --depends libsdl-image1.2  \
        --depends libsdl1.2debian  \
        --depends libpulse0  \
        --depends libvorbisfile3  \
        --depends libgssdp-1.0-3  \
        --depends libsoup-gnome2.4-1  \
        --depends libpango1.0-0  \
        --depends libsoup2.4-1  \
        --depends libexpat1  \
        --depends libuuid1  \
        .=/usr/share/tix TiX.desktop=/usr/share/applications/TiX.desktop
cp *.deb ..
cd ../..
```

package_osx.sh

```
#!/bin/bash
set -e
rm -rf dist/* build/*
# Fix for kivy.app on mac's source code
sed -i '' 's/IOError/Exception/'  \
/Applications/Kivy.app/Contents/Resources/lib/sitepackages/pygame/macosx.py
kivy `which pyinstaller` TiX/Tix.unix.spec
mv dist/TixApp dist/TixApp.app
hdiutil create Tix.dmg -srcfolder dist/TixApp.app -ov
```

tix_release_version.sh

```
#!/usr/bin/env bash

DIR="$( cd "$( dirname "${BASH_SOURCE[0]}" )" && pwd )"
source $DIR/tix_lib.sh

get_os
get_variant

current_branch=$(git rev-parse --abbrev-ref HEAD)

create_new_tag() {
  echo "Commiting the package"
  rm -rvf releases/**/*.deb
  rm -rvf releases/**/*.dmg
  git add -A . > /dev/null
  git commit -m "Release sources for $os/$variant/head" > /dev/null
```

```
  echo "Moving to 'releases' branch"
  git checkout origin/releases > /dev/null
  git checkout releases > /dev/null
  echo "Clearing 'releases' folder"
  rm -rfv *
  echo "Copying files from $current_branch"
  git checkout $current_branch -- releases/ > /dev/null
  rm -rvf releases/**/*.deb
  echo "Commiting release"
  git add -A . > /dev/null
  git commit -m "Release sources for $os/$variant/head" > /dev/null
  echo "Creating tag"
  git tag $os/$variant/head --force > /dev/null
  echo "Created tag $os/$variant/head on branch releases"
  echo "Finishing..."
  git checkout $current_branch > /dev/null
  echo "DONE!"
}

package_os() {
  echo "Packaging TIX..."
  get_os
  case $os in
    linux)
      bash $DIR/../scripts/package_linux.sh 2>&1 | xargs echo > /dev/null
      ;;
    mac)
      bash $DIR/../scripts/package_osx.sh 2>&1 | xargs echo > /dev/null
      ;;
  esac
}

prepare_files() {
  get_os
  get_variant
  rm -rf $DIR/../releases/*
  case $os in
    linux)
      cp -r $DIR/../dist/TixApp $DIR/../releases/
      ;;
    mac)
      cp -r $DIR/../dist/TixApp.app $DIR/../releases/
      ;;
  esac
}

tix_release_version() {
  package_os
  prepare_files
  create_new_tag
}
```

```
tix_release_version "$@"
```

my_backup.rb

```
Model.new(:backup, 'Backup for TiX databases') do

  store_with Local do |local|
    local.path       = "~/backups/"
    local.keep       = 90
  end

  database PostgreSQL do |db|
    # To dump all databases, set `db.name = :all` (or leave blank)
    db.username           = "root"
    #db.name           = "tix_db"
    db.password           = "<PASSWORD>"
    db.host               = "localhost"
    db.port               = 5432
  end

  compress_with Gzip

end
```

Apéndice C

Manual de usuario

En esta sección de describe en detalle la utilización de los requerimientos implementados.

C.1. Instalación del ejecutable

C.1.1. Registro

Para instalar el ejecutable el usuario primero debe registrarse ingresando a http://tix.innova-red.net/ y haciendo click en Instalar TiX.

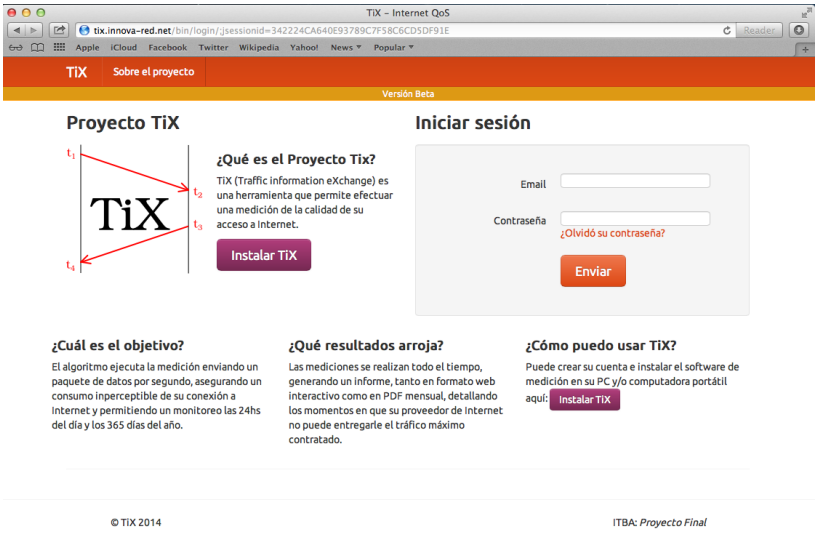

FIGURA C.1: Acceso inicial al sitio http://tix.innova-red.net

Allí debe llenar sus datos personales para poder crear su usuario.

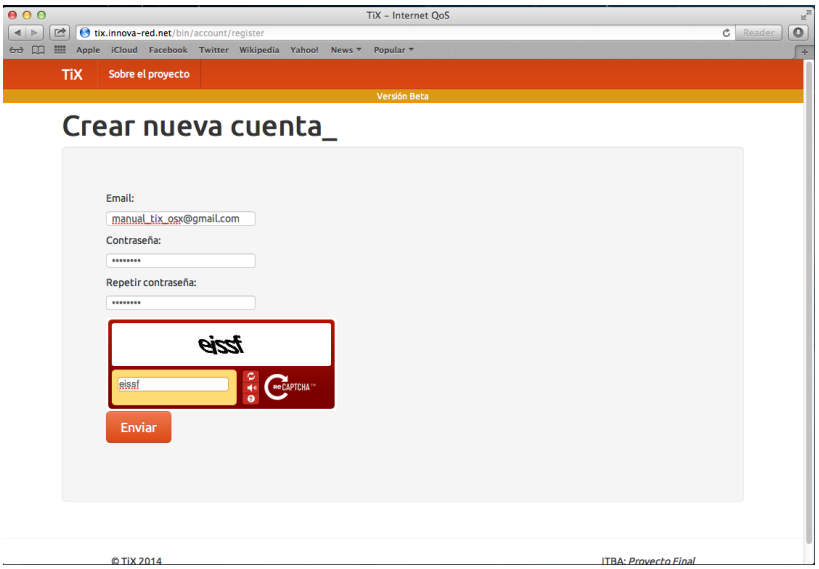

FIGURA C.2: Formulario de Registro

Finalmente debe elegir la plataforma sobre la cual quiere instalar su ejecutable.

FIGURA C.3: Elección de plataforma

C.1.2. Instalación - Linux

En el caso de tener una distribución de Linux como Ubuntu, o cualquier variante de *Debian*, debe hacer doble click en el archivo *.deb* del proyecto una vez descargado. Tras seguir la instalación de su gestor de paquetes, tendrá disponible TixApp en su carpeta de aplicaciones.

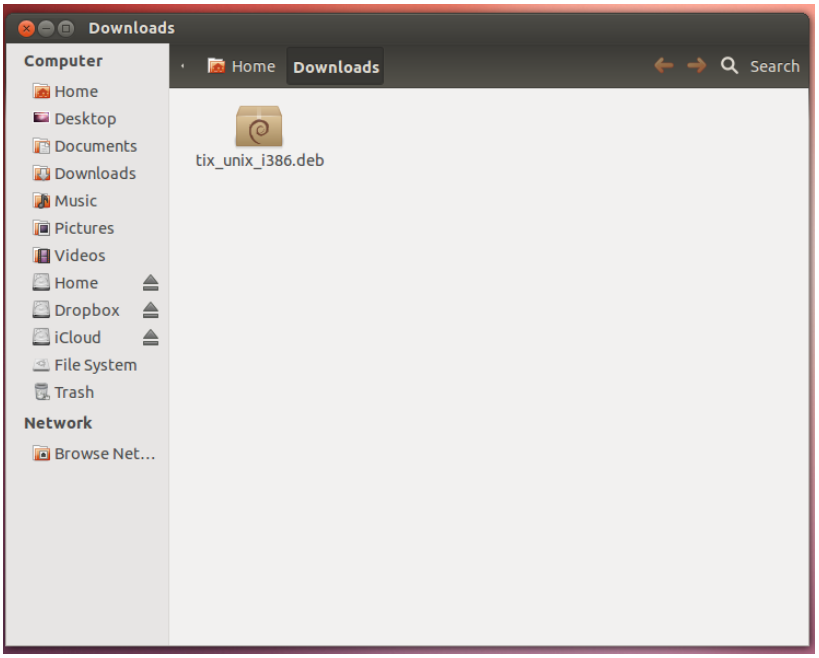

FIGURA C.4: Archivo *.deb* para instalar en *Debian*

C.1.3. Instalación - Mac OSX

En el caso de tener Mac OSX, el usuario debe copiar el archivo Tix.app contenido en Tix.dmg en la carpeta de aplicaciones, al igual que cualquier aplicación diseñada para Mac OSX. El ejecutable se puede abrir haciendo click en Tix.app.

FIGURA C.5: Ubicación de *Tix.app* en la carpeta de Aplicaciones

C.1.4. Creación de la instalación

Al abrir el ejecutable de TiX se solicitan el usuario y la contraseña.

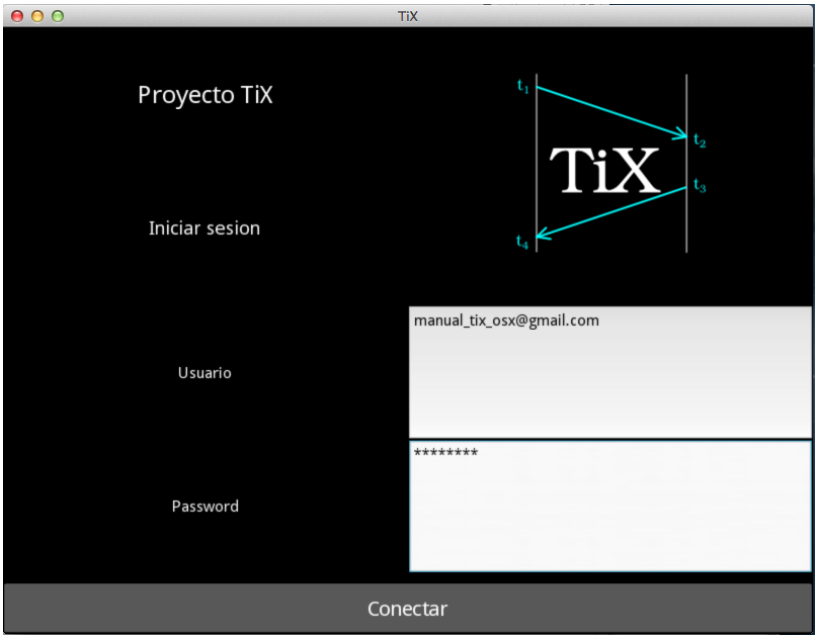

FIGURA C.6: Inicio de sesión

Seguido de esto se solicita que se coloque un nombre a la instalación que se va a crear.

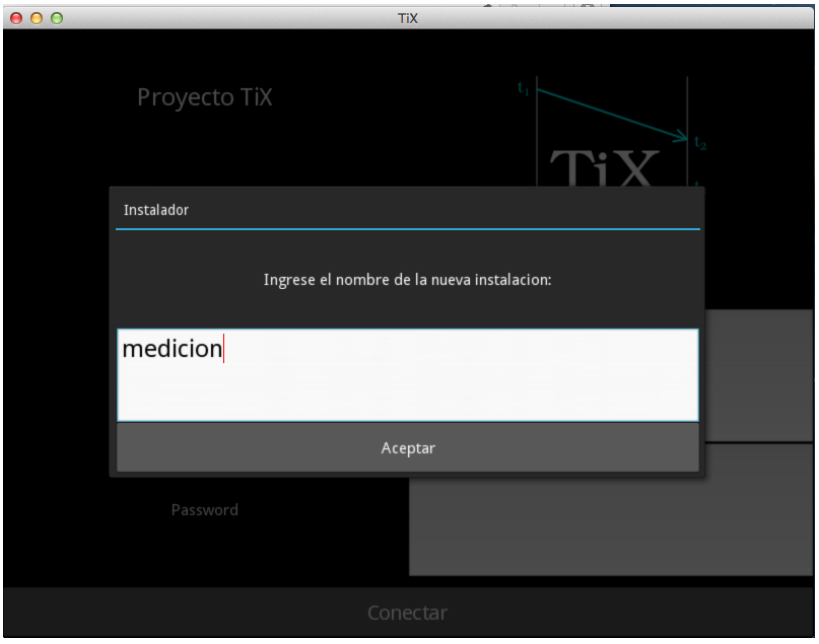

FIGURA C.7: Creación de instalación

Finalmente se notifica que se crea la instalación y el programa termina. A partir de ese momento ya están comenzando las mediciones.

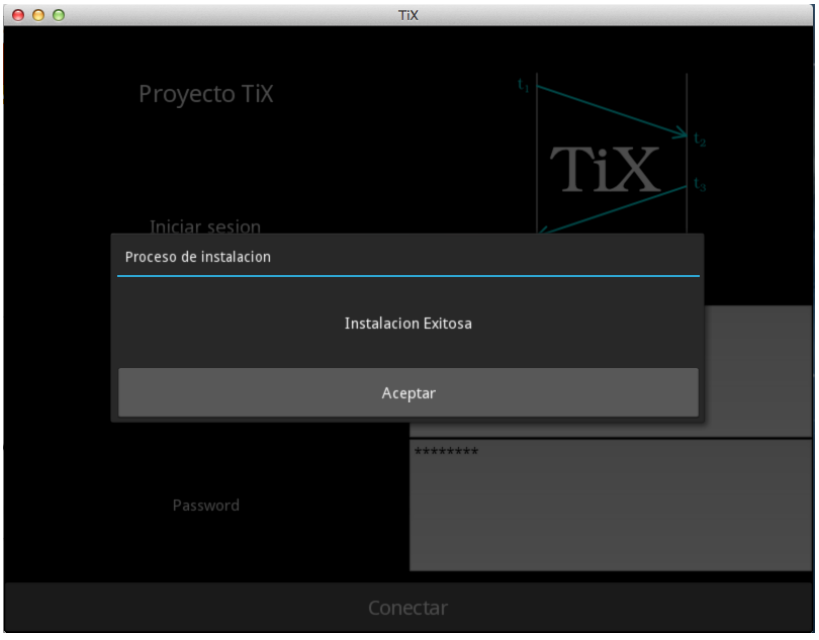

FIGURA C.8: Confirmación de instalación exitosa

C.1.5. Reportes de administrador

C.1.6. Acceso

Para acceder a sus reportes, el administrador debe acceder a la plataforma utilizando sus credenciales desde la página inicial de TiX.

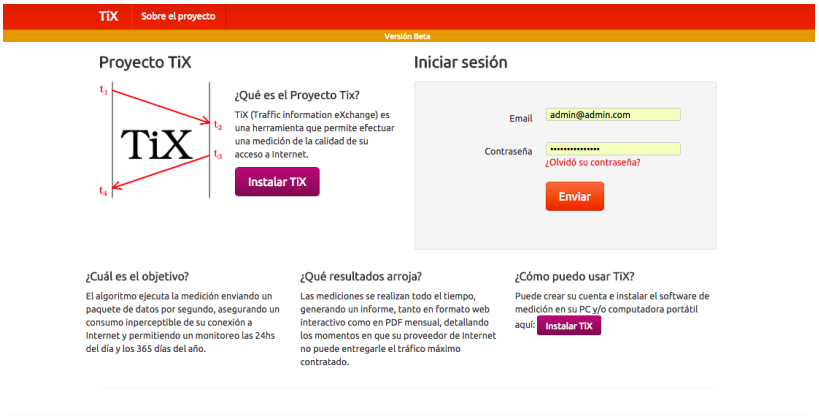

FIGURA C.9: Pantalla inicial con credenciales de administrador.

Habiendo ingresado sus credenciales y accedido a su panel deberá hacer *click* sobre el botón "Ver gráficos de utilización y calidad".

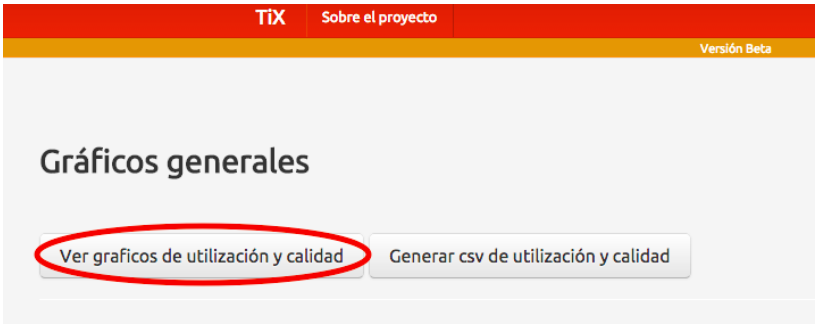

FIGURA C.10: Botón para acceder al reporte del administrador.

Una que que el usuario ha hecho *click* sobre el botón "Ver gráficos de utilización y calidad", será redirigido a la siguiente pantalla donde podrá acceder a la vista de reporte de administrador.

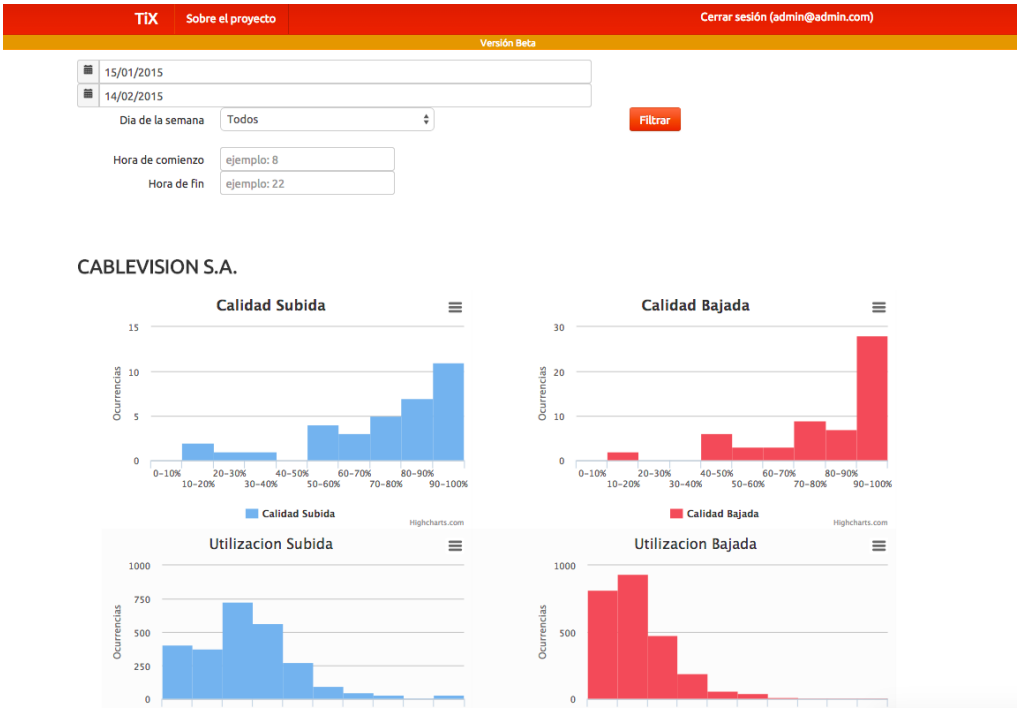

FIGURA C.11: Reporte del administrador.

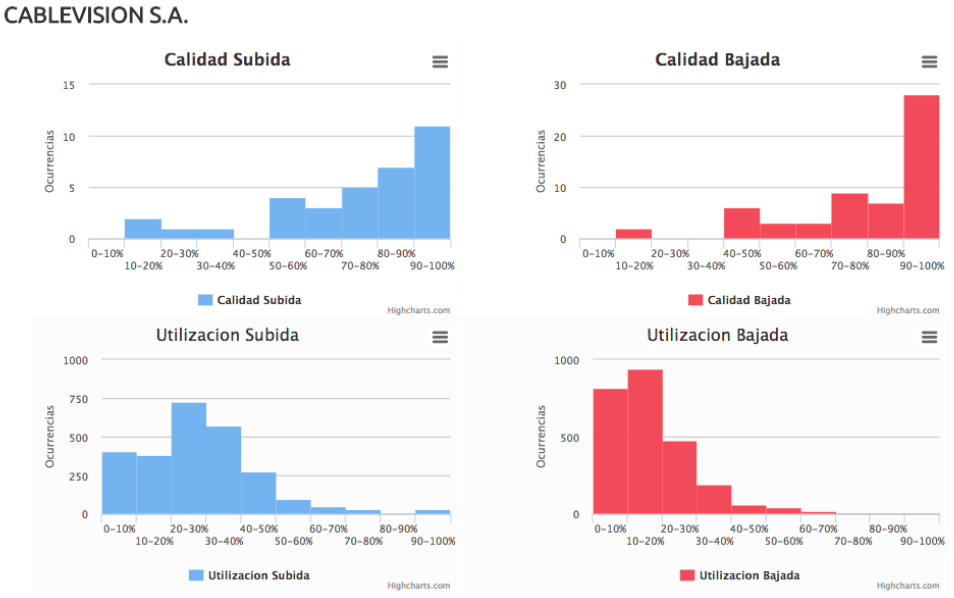

FIGURA C.12: Histograma de reporte de administrador para proveedor.

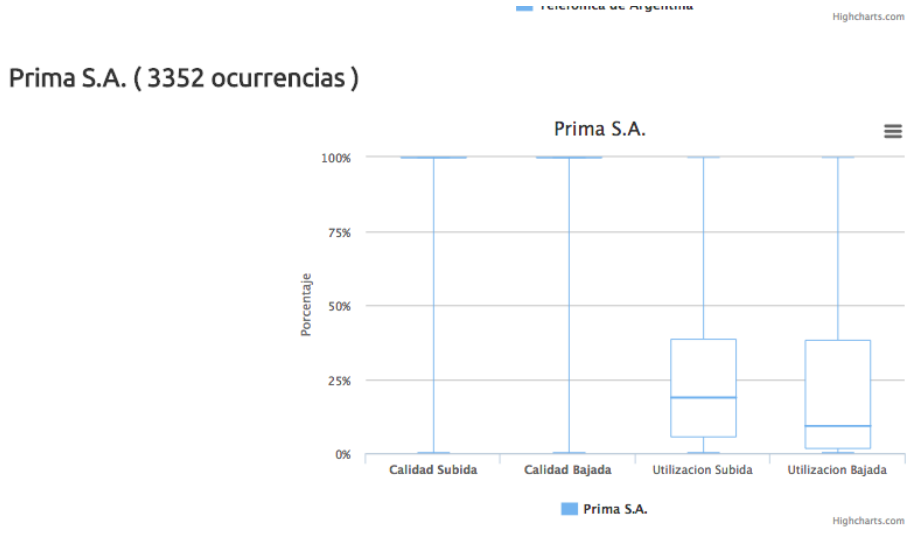

FIGURA C.13: Diagrama de bloques de reporte de administrador para proveedor.

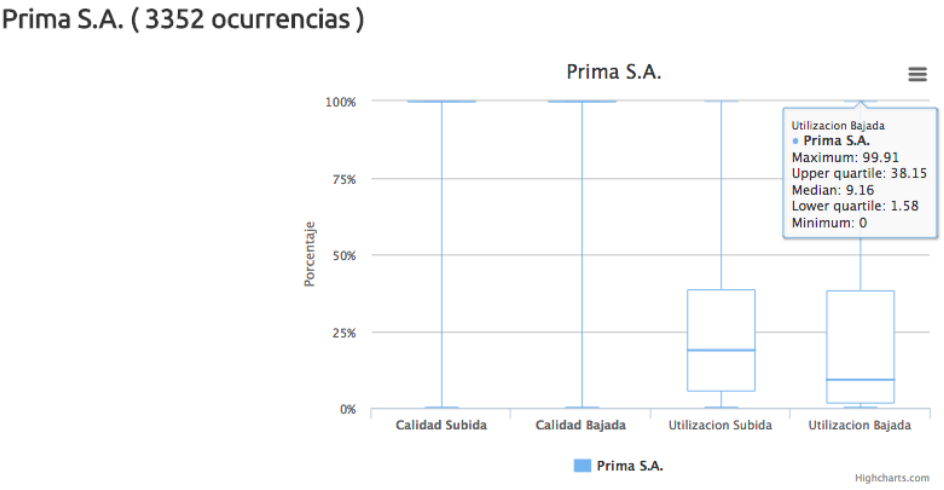

FIGURA C.14: Detalle de diagrama de bloques.

C.1.7. Filtrado por fecha

Una vez que el administrador está dentro de su vista de reporte, podrá filtrar por fecha ingresando una fecha en los dos cuadros en entrada superiores.

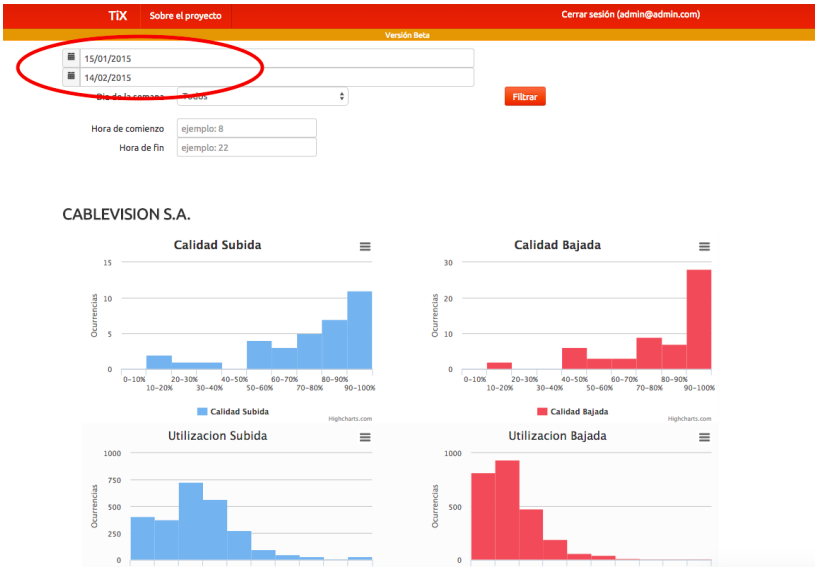

FIGURA C.15: Botón para acceder al reporte del administrador.

Una vez que las nuevas fechas fueron ingresadas, bastará con que el administrador haga *click* sobre el botón filtrar para obtener el reporte con la información filtrada.

C.1.8. Filtrado por día

Una vez que el administrador está dentro de su vista de reporte, podrá filtrar por día seleccionando en el *checkbox* de selección el día que corresponda.

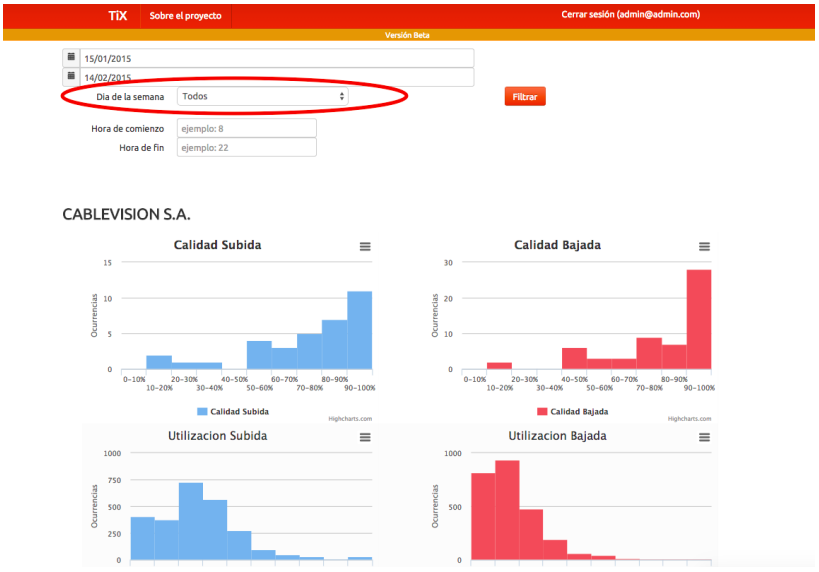

FIGURA C.16: Botón para acceder al reporte del administrador.

Una vez que el nuevo día es ingresado, bastará con que el administrador haga *click* sobre el botón filtrar para obtener el reporte con la información filtrada.

C.1.9. Filtrado por hora

Una vez que el administrador está dentro de su vista de reporte, podrá filtrara por hora del día ingresando una hora en los dos cuadros en entrada inferiores.

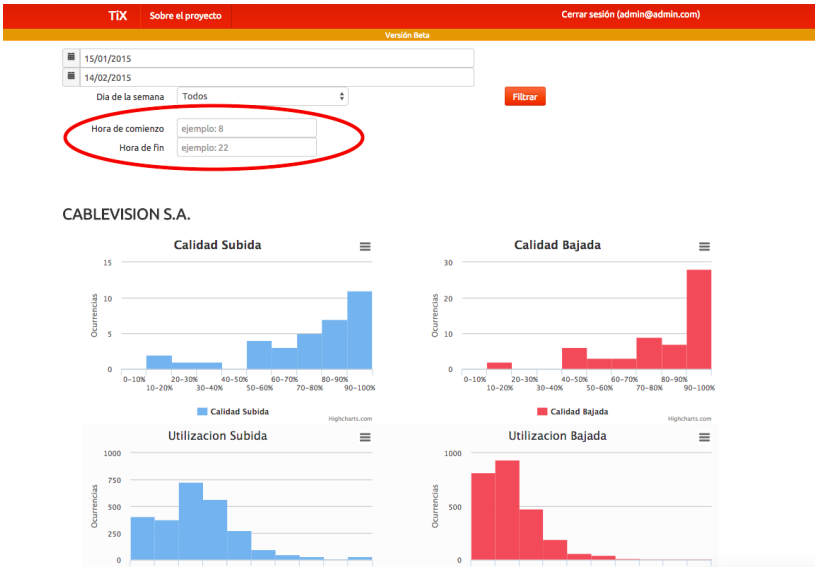

FIGURA C.17: Botón para acceder al reporte del administrador.

Una vez que las nuevas horas fueron ingresadas, bastará con que el administrador haga *click* sobre el botón filtrar para obtener el reporte con la información filtrada.

C.1.10. Descarga de información

Para acceder a sus reportes, el administrador debe acceder a la plataforma utilizando sus credenciales desde la página inicial de TiX.

Habiendo ingresado sus credenciales y accedido a su panel deberá hacer *click* sobre el botón "Generar csv de utilización y calidad".

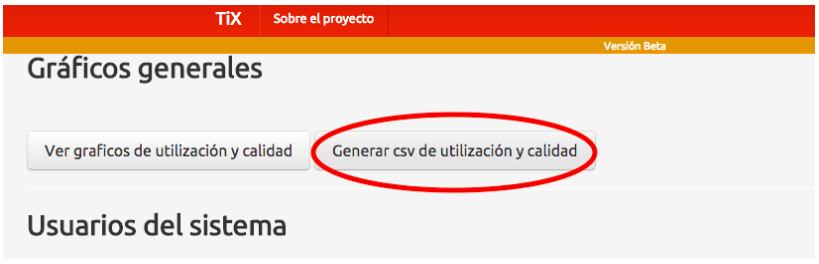

FIGURA C.18: Botón para acceder a la información en formato CSV.

Eso hará que el navegador le pregunte si quiere descargar el archivo con la información correspondiente. El administrador tendría que aceptar para bajar el archivo y acceder a la información de los registros.

C.2. Reportes de usuario

C.2.1. Acceso

Para acceder a sus reportes, el usuario debe acceder a la plataforma utilizando sus credenciales desde la página inicial de TiX.

Habiendo ingresado sus credenciales y accedido a su panel deberá hacer *click* sobre el botón Reporte de usuario".

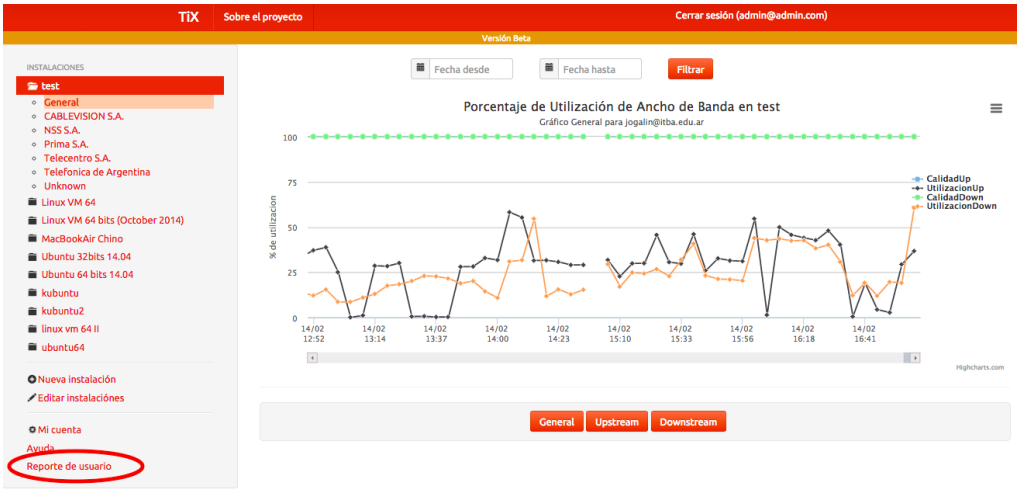

FIGURA C.19: Botón para acceder al reporte del usuario.

Esa acción conducirá al usuario a la vista con su reporte donde podrá ver una tabla con medianas mensuales para cada proveedor de internet que usó y un gráfico con información agregada del último mes por cada instalación.

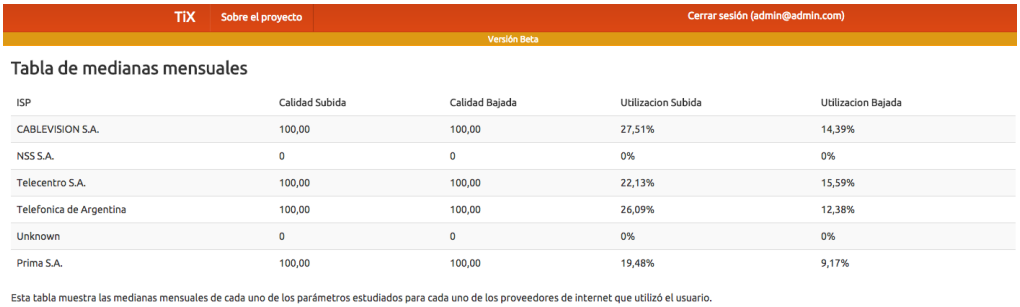

FIGURA C.20: Tabla de medianas en reporte de usuario.

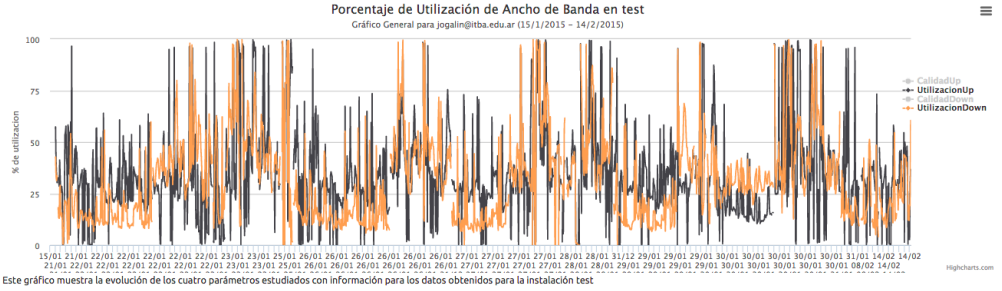

FIGURA C.21: Gráfico con información agregada en reporte de usuario.