PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
FACULTAD DE CIENCIAS E INGENIERÍA

SISTEMA ADMINISTRADOR DE RECURSOS PARA UNA *GRID* COMPUTACIONAL BASADA EN EL *MIDDLEWARE* BOINC

Tesis para optar por el Título de Ingeniero Informático, que presenta el bachiller:

Renzo Phellan Aro

ASESOR: Leopoldo Genghis Ríos Kruger

Lima, octubre del 2012

Resumen

Actualmente, la administración de los recursos de la *grid* computacional de la Pontificia Universidad Católica del Perú (PUCP) desarrollada por la Dirección de Informática Académica (DIA) se ha convertido en una tarea compleja. Esto debido al incremento de la cantidad de recursos relacionados a la *grid*: administradores, computadoras de escritorio y proyectos científicos. Es por ello que se ha decidido elaborar un sistema informático que ayude a organizar la labor administrativa. Ello permitirá lograr continuidad en el crecimiento de la *grid*, promoviendo así la aplicación del súper cómputo en el ámbito académico; lo que llevará a que la PUCP se convierta en una organización de vanguardia en el uso de las TIC.

El sistema propuesto como proyecto de fin de carrera permitirá automatizar y organizar las actividades administrativas de los recursos vinculados a la *grid*. Además, se encargará de establecer y mantener la comunicación entre las computadoras y los proyectos científicos de la PUCP que los aprovechan. También, permitirá asociar los recursos al área académica a la que pertenecen. Por otra parte, contará con opciones para agrupar las computadoras según sus características de hardware y configurar sus preferencias de uso. Adicionalmente, ayudará a generar reportes con información relevante sobre los recursos de la *grid*. Por último, brindará opciones para distribuir la carga administrativa entre varios administradores, con diferentes roles, según el área académica a la que pertenecen.

Tabla de Contenidos

Índice de ilustraciones

Índice de tablas

Introducción

La Dirección de Informática Académica (DIA) de la Pontificia Universidad Católica del Perú (PUCP) tiene la misión de planificar y promover la aplicación de las TIC en el ámbito académico. Esto con el objetivo de convertir a la universidad en una organización de vanguardia en el uso de las mismas (DIA 2006).

A consecuencia de lo indicado, desde fines del año 2008, la DIA se encuentra involucrada en el proyecto Legión (Quintana 2009). Éste se enfoca en aplicar la tecnología de computación *grid* para poner a disposición de los proyectos científicos de la PUCP el poder de cómputo reunido de las aproximadamente 500 computadoras a cargo de la DIA (Rios 2009).

Como se puede notar, la *grid* comprende una cantidad considerable de computadoras. Esto hace que su administración represente una tarea compleja para el personal actual. En particular, al momento de planificar cuáles computadoras apoyarán a qué proyectos científicos; organizar los recursos según su ubicación geográfica, área académica o características de *hardware;* o de controlar el estado de los mismos.

Frente al problema mencionado, se propone como solución elaborar un sistema de información que permita automatizar y organizar las actividades administrativas de los recursos vinculados a la *grid*.

1 Generalidades

En este capítulo se ofrece una visión general del proyecto de fin de carrera, enmarcándolo en el contexto en el que se desarrolla. Para ello, se inicia con una descripción del problema a tratar. Luego, se presenta un conjunto de conceptos necesarios para una mejor comprensión del proyecto. A continuación, se lista un grupo de soluciones que tratan el problema planteado, analizando sus principales características. Después, se aborda la metodología a utilizar para el desarrollo del proyecto, junto con el plan que se deriva de ésta. Para terminar, se brinda una explicación detallada de la solución a desarrollar, justificando el porqué de su elección.

1.1 Definición del Problema

El problema que busca solucionar el presente proyecto es la dificultad que representa las labores de administración de la *grid* computacional de la PUCP. En los siguientes párrafos se presenta el contexto particular del problema y su detalle.

La Dirección de Informática Académica es una unidad de la PUCP. Tiene como objetivo central planificar y promover la aplicación de las TIC en el ámbito académico. Con ello se busca convertir a la universidad en una organización de vanguardia en el uso de las mismas (DIA 2006).

Para cumplir el objetivo planteado la DIA desarrolla diversos proyectos tecnológicos que dan soporte a las actividades académicas. Así se tiene, por ejemplo, Blog PUCP, que permite a los miembros de la comunidad universitaria publicar periódicamente artículos relacionados al quehacer académico; Videos PUCP, un espacio que permite a los miembros de la comunidad universitaria publicar videos de interés académico; Paideia PUCP, una plataforma en línea que permite la interacción virtual entre alumnos y profesores, permitiendo la publicación de material de clases y tareas (DIA 2006).

En consonancia con lo indicado, desde fines del año 2008, la DIA se encuentra involucrada en el proyecto Legión (Quintana 2009). Su objetivo es conformar una *grid* computacional que aproveche la infraestructura informática de la universidad, en particular, las computadoras de los laboratorios. Para ello, se recurre al *middleware* BOINC, que permite hacer uso de la capacidad computacional ociosa de estos últimos. Cada computadora integrada a la *grid* dona ciclos de CPU que son aprovechados por científicos de diversas áreas de la PUCP para realizar cálculos complejos como parte de sus investigaciones. Por ejemplo, en los campos de bioinformática, minería de datos, química cuántica y física de altas energías (DIA 2011; Ríos 2009).

Inicialmente, la *grid* de la PUCP comprendía alrededor de 120 computadoras, distribuidas en 3 laboratorios. Para su administración se recurría a un método no automatizado. Este consistía en llevar un registro en una hoja de cálculo con la dirección IP de cada computadora, sus características, su ubicación y los proyectos científicos a los que se encontraba asignado. Entonces, cuando se solicitaba apoyo para un nuevo proyecto, se consultaba el registro y se decidía qué computadoras asignarle. Para ello, se hacía uso de la interfaz de línea de comandos que ofrece el *middleware* BOINC para realizar tareas administrativas.

Actualmente, el proyecto Legión comprende, aproximadamente, 500 computadoras distribuidas en 15 laboratorios, por lo que su administración representa una tarea ardua. Primero, porque se requiere de tiempo y esfuerzo para mantener actualizado el registro de computadoras y, dado que se modifica manualmente, es posible cometer errores. Además, eventualmente, se requiere de reportes sobre el desenvolvimiento y estado de la *grid.* Por ejemplo, reportes sobre cuántos ciclos de CPU fueron utilizados en cada proyecto científico o cuáles computadoras se encuentran actualmente funcionando y listas para ejecutar tareas. Su elaboración es una tarea tediosa que comprende muchas consultas manuales a las bases de datos de los proyectos científicos, donde el *middleware* BOINC almacena las transacciones realizadas.

Frente a los problemas mencionados se ha adoptado como solución provisional el sistema Jarifa, pero este no los soluciona satisfactoriamente. En primer lugar, Jarifa mantiene un registro de computadoras, pero éste presenta solo un subconjunto de las características de *hardware* que transmite el *middleware* BOINC. Esto es un problema, puesto que dificulta la identificación de las computadoras idóneas para un determinado proyecto. En segundo lugar, Jarifa permite la asignación y retiro de computadoras a los proyectos, mas lo hace de manera global; es decir, asume que todas las computadoras registradas deben apoyar a todos los proyectos registrados. Esto es un problema, ya que en la DIA se necesita poder agrupar las computadoras y asignarlas, de acuerdo a sus características, a diferentes proyectos.

En conclusión, la *grid* computacional de la DIA forma parte de un proyecto que busca ayudar a la PUCP a convertirse en una institución que aprovecha las TIC. No obstante, dado su tamaño actual, las labores de administración de la misma son complejas. Inicialmente, era posible realizar estas tareas de manera no automatizada. Sin embargo, actualmente, esto trae problemas. Es por ello que se decidió utilizar el sistema Jarifa. Empero, este último no logra solucionar satisfactoriamente el problema.

1.2 Marco conceptual

En esta sección se presenta un conjunto de conceptos necesarios para una mejor comprensión del proyecto de fin de carrera. El enfoque que se ha seguido para ordenarlos describe, primero, los conceptos más básicos, para luego poder definir los siguientes en base a los primeros.

1.2.1 Capacidad computacional ociosa

A. Tanembaum (Tanembaum 2008) clasifica los procesos que realizan las computadoras en dos: *CPU-bound* e *I/O bound*. Como se puede ver en la ilustración 1.1, la diferencia entre ambos radica en el tiempo que la unidad central de procesamiento de la computadora (CPU) se encuentra trabajando.

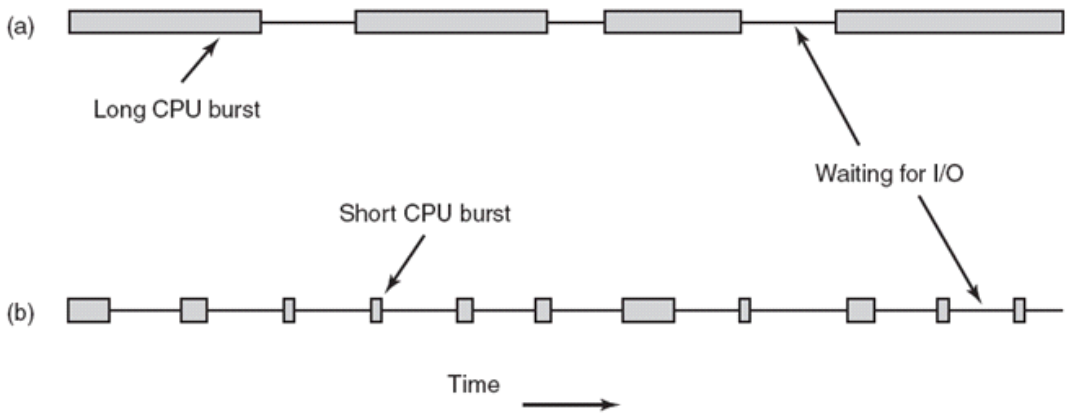

Figure 2-38. Bursts of CPU usage alternate with periods of waiting for I/O. (a) A CPU-bound process. (b) An I/O-bound process.

Ilustración 1.1: CPU-bound process and I/O-bound process (Tanembaum 2008)

Las CPUs tienden a volverse más rápidas con el tiempo, por lo que los procesos tienden a estar dedicados a operaciones de entrada y salida *(I/O bound*) (Tanembaum 2008). Esto trae como consecuencia que exista un mayor tiempo durante el cual la CPU se encuentra sin realizar ninguna tarea. Es a este tiempo, al que en adelante se referirá como capacidad computacional ociosa.

1.2.2 Computación en *grid*

La Universidad de California (Universidad de California 2012a) define a la computación en *grid* como "a form of distributed computing in which an organization (business, university, etc.) uses its existing computers (desktop and/or cluster nodes) to handle its own long-running computational tasks". A partir de la definición anterior se puede realizar las siguientes precisiones.

Primero, la computación en *grid* permite utilizar de manera conjunta la capacidad computacional ociosa de cada nodo o computadora individual, logrando así capacidades iguales a las de una supercomputadora. Todo ese poder reunido se encuentra disponible para realizar tareas exigentes en términos de procesamiento.

Segundo, las computadoras no necesariamente tienen que estar reunidas físicamente en un determinado lugar de la organización. Se puede utilizar las computadoras personales distribuidas en diferentes áreas.

1.2.3 BOINC

BOINC es un proyecto de *software* libre, actualmente vigente, respaldado por la Universidad de California. Se define como: "A software platform for volunteer computing and desktop Grid computing" (Universidad de California 2012a). Entonces, BOINC se constituye como un *software* intermedio o *middleware*, que facilita la creación de otras aplicaciones que permiten establecer, utilizar y administrar una *grid* computacional.

Con respecto a la arquitectura que utiliza BOINC. Esta consta de dos componentes principales: el servidor BOINC y los clientes BOINC (Universidad de California 2012a). El servidor BOINC se encarga de almacenar las tareas que se deben realizar para poder completar un proyecto que requiere de gran capacidad de cómputo. También, tiene como función distribuir esas tareas para que se realicen en los nodos que conforman la *grid* y, finalmente, reunir los resultados de cada tarea para dar respuesta al proyecto (Universidad de California 2012a; Díaz 2008). Los clientes BOINC son cada uno de los nodos que forman parte de la *grid*. Su función es la de realizar una tarea de cómputo que le asigne el servidor BOINC y devolver un resultado (Universidad de California 2012a; Díaz 2008).

Por otra parte, BOINC comprende un conjunto de conceptos básicos a los que se hace referencia a lo largo de este proyecto. Estos se presentan a continuación:

a) Tarea

Una tarea es una entidad compuesta por un conjunto de *work units* y *results.* Un *work unit* representa una computación que será ejecutada por un cliente BOINC e incluye un conjunto de datos de entrada. Un *result* es la instancia de un *work unit* en un determinado cliente BOINC y contiene el resultado de la computación (Universidad de California 2012a).

Lo dicho anteriormente permite que se pueda obtener varios *results,* calculados por diferentes clientes, para un mismo *work unit.* Luego, es posible comparar varios *results* y verificar así que el cálculo es correcto (Universidad de California 2012a). Una vez que todos los *work units* de una tarea han sido ejecutados satisfactoriamente, se da por concluida la tarea.

b) Proyecto BOINC

Un proyecto BOINC es una aplicación científica que comprende un conjunto de tareas programadas de acuerdo a un determinado protocolo computacional, diseñadas para ser ejecutadas por un cliente BOINC. Para su realización requiere, entre otros recursos, de un servidor central, un sitio *web* y participantes que brinden su capacidad computacional ociosa (Malton 2010).

De la experiencia que se tiene, se sabe que un proyecto BOINC no necesariamente concluye cuando se han ejecutado todas las tareas programadas, ya que es posible que se generen nuevas tareas en el tiempo.

Como ejemplo emblemático de proyecto BOINC se tiene a SETI@home. “Este proyecto es un experimento científico que utiliza ordenadores conectados a Internet para la búsqueda de inteligencia extraterrestre” (Universidad de California 2012b). Fue diseñado por el Doctor en Ciencias de la Computación David P. Anderson, actual Director a cargo del sistema BOINC.

c) Crédito

Es una unidad de medida de la cantidad de esfuerzo, en términos de capacidad computacional, contribuida por un colaborador a las tareas de un proyecto BOINC (Malton 2010).

Para ganar créditos, un usuario debe retornar los *results* que le fueron asignados. Estos son verificados de acuerdo al proceso descrito en el punto a). Si son correctos, obtiene una determinada cantidad de créditos. Cada proyecto BOINC calcula cuántos créditos asignará al usuario.

1.2.4 Centro de supercómputo virtual para campus (VCSC)

La Universidad de Westminster afirma haber realizado un ahorro de aproximadamente 125 000 libras esterlinas cada año gracias a su sistema de computación en *grid* basado en BOINC. La *grid* consiste de un total de 1500 computadoras personales distribuidos en los laboratorios de la universidad. Juntos, estos proporcionan el poder de cómputo necesario para evitar la necesidad de comprar una supercomputadora cada cuatro años, valorizada esta última en aproximadamente 500 000 libras esterlinas (Evans-Toyne 2011).

El párrafo anterior presenta un caso de aplicación de la computación en *grid* basada en BOINC a una organización universitaria. Como se puede ver, los ahorros en costos son considerables. Es por ello que la Universidad de California define el concepto de VCSC para identificar este caso particular.

> "A VCSC can provide researchers with the computational power of a large cluster or supercomputer, for a small fraction of the cost. A VCSC is a BOINC project whose applications are supplied by campus researchers. The computing power is supplied by campus PCs." (Universidad de California 2007)

En resumen, un VCSC es una aplicación de la computación en *grid* basada en BOINC a un campus universitario. Éste permite ahorros significativos y un mejor aprovechamiento de la infraestructura tecnológica. La ilustración 1.2 muestra el esquema general de un VCSC.

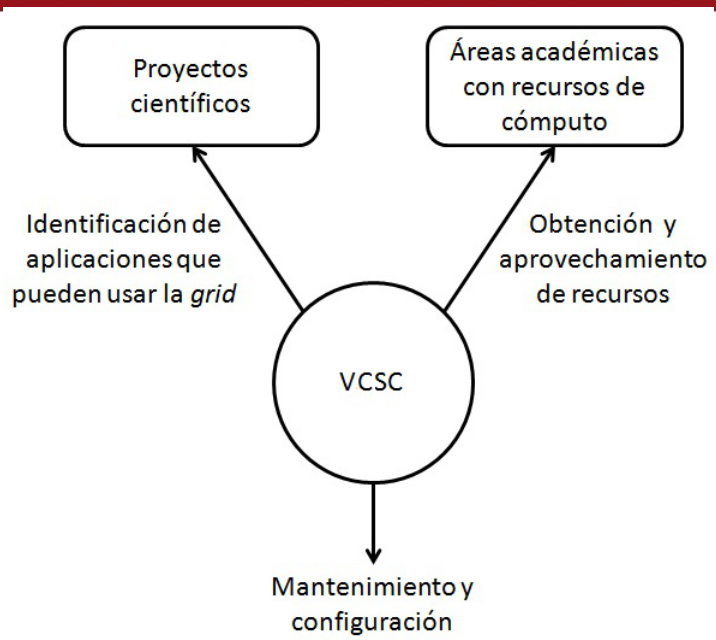

Ilustración 1.2: Esquema de un VCSC (Adaptado de Universidad de California 2012a)

1.2.5 Sistema Legión

Legión es un sistema de computación en *grid* que hace uso de BOINC, desarrollado por la DIA (DIA 2011; Fonseca 2011; Quintana 2009). Permite establecer un VCSC en la PUCP que provee a los investigadores con la infraestructura necesaria para realizar cálculos que requieren un alto poder de procesamiento (Fonseca 2011; Iberico 2009; Quintana 2009; Ríos 2009).

Concordando con su definición, Legión utiliza la capacidad computacional ociosa de las computadoras instaladas en los laboratorios de la PUCP a cargo de la DIA (Ríos 2009). Esto trae dos consecuencias importantes. Primero, el sistema ofrece una potencia de cálculo equivalente a la de 250 procesadores Intel Core 2 Duo (Fonseca 2011). Segundo, dado que solo se aprovechan los recursos ociosos, los usuarios de los laboratorios no deben notar una diferencia significativa en el rendimiento debido a la presencia de Legión (Ríos 2009).

1.2.6 Sistema de administración de cuentas

Como ya se mencionó, BOINC permite dar soporte a proyectos científicos, conformados por varias tareas, que van a ser realizadas por cada uno de los nodos

que conforman la *grid*. Luego, cada uno de estos últimos debe ser asignado individualmente a cada proyecto al que su responsable decida apoyar para poder comenzar a recibir tareas. Esta labor puede resultar tediosa para el administrador de la *grid* en el caso de un VCSC, donde se tiene muchos equipos y varios proyectos que surgen a lo largo del tiempo (Universidad de California 2012a).

Entonces, para solucionar este problema, BOINC sugiere el uso de un sistema de administración de cuentas. Este consiste, típicamente, en un sitio *web* donde, primero, se registran los proyectos. Después, el responsable de un grupo de computadoras solicita una cuenta con sus credenciales al sistema. Estas credenciales se ingresan por única vez en cada cliente BOINC para poder asociarlo a la cuenta del responsable. Una vez realizado esto, se puede acceder a la interfaz *web* que ofrece el sistema de administración de cuentas para realizar la gestión de los recursos, lo que incluye su asociación a los proyectos. Finalmente, BOINC se encarga de que cada nodo se conecte periódicamente y de manera automática a los proyectos para solicitar que se le asigne tareas (Universidad de California 2012a). En resumen, el sistema de administración de cuentas es un intermediario entre los servidores de proyectos BOINC y los clientes BOINC que facilita las labores administrativas de la *grid*. Un esquema de lo antes descrito se presenta en la ilustración 1.3.

Por otra parte, a un nivel técnico, el sistema de administración de cuentas se basa en el uso de llamadas a procedimientos remotos por vía *web*. Los datos viajan a través de mensajes XML, utilizando el protocolo HTTP o HTTPS (Universidad de California 2012a).

A partir de este mecanismo de administración de recursos de la *grid* se ha construido diversas aplicaciones que explotan otras funcionalidades provistas por BOINC y que enriquecen aún más la labor administrativa. En el siguiente capítulo se presentan algunas de estas aplicaciones.

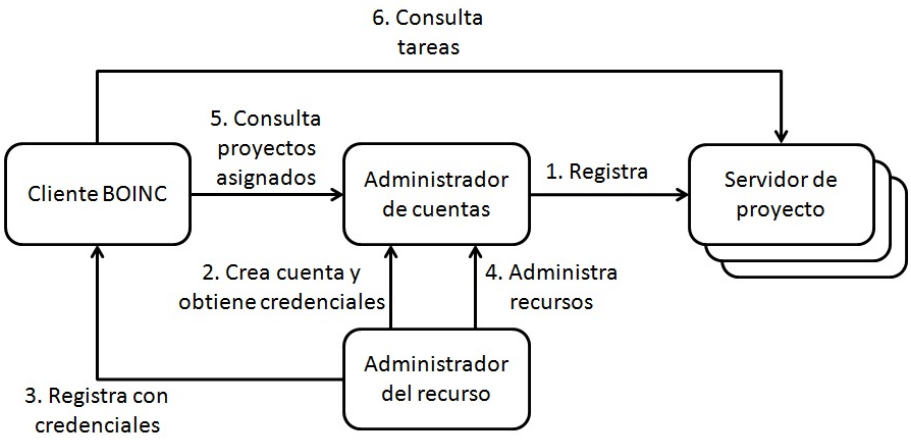

Ilustración 1.3: Sistema de administración de cuentas

1.3 Estado del arte

En esta sección se lista un grupo de soluciones ya existentes, que implementan el sistema de administración de cuentas de BOINC. A manera de resumen, se presenta al final de la sección un cuadro comparativo con las soluciones mencionadas.

1.3.1 BOINC Account Manager

BAM! (BOINC *Account Manager*) es un sistema que permite a un administrador poner sus recursos de cómputo a disposición de un conjunto de proyectos que requieren de la computación en *grid* (Boincstats Team 2011). Los proyectos vienen predefinidos por BAM! y, en su mayoría, son de interés científico.

BAM! es un sistema de administración de cuentas que incluye un conjunto de opciones que facilitan al administrador la gestión de sus recursos de cómputo. Por ejemplo, la capacidad de decidir el nivel y horarios de uso de los recursos; de asignarle un nivel de prioridad a cada proyecto o de saber el estado actual de los proyectos (ver anexos 1, 2 y 3).

La principal ventaja de BAM! es la presentación de reportes estadísticos y gráficos comparativos actualizados sobre el nivel de progreso de los proyectos y el de uso de los recursos (ver anexos 4 y 5). Su desventaja es que ha sido concebido para dar soporte a proyectos a nivel mundial, por lo que una organización particular no tiene control sobre qué proyectos pueden o no ser registrados en el sistema. Es

decir, no es posible limitar BAM! solo a la administración de los proyectos que elija la organización, lo cual es un requerimiento para el caso de la PUCP.

Una característica adicional e importante de BAM! es el manejo de un sistema de créditos (ver anexos 4 y 5). Estos miden la cantidad de ciclos de cómputo entregados por cada administrador de recursos a cada proyecto. Entonces, es posible elaborar una lista con los mejores colaboradores en términos de procesamiento realizado o de los proyectos con mayor cantidad de créditos acumulados. Esto podría aplicarse también, en el caso de una universidad, para medir el nivel de colaboración de cada unidad y promover una mayor participación.

1.3.2 Jarifa

Jarifa es un “system for grid computing on organizational resources, using BOINC.” (Lombraña 2010; Lombraña 2011)

Desde una perspectiva general, Jarifa es un sistema que permite, por un lado, mantener un directorio de proyectos (ver anexo 6) y, por otro, listas de recursos de cómputo (ver anexo 7) que pueden realizar las tareas de cada proyecto. Luego, el sistema se encarga de asignar a cada recurso una tarea correspondiente a alguno de los proyectos que tiene registrados.

Conceptualmente, Jarifa fue diseñado para operar en situaciones en las que un conjunto de proveedores de recursos de cómputo optan por permitir a un comité administrador decidir a qué proyectos BOINC asignar estos recursos, tal y como se puede ver en la ilustración 1.4. Los proveedores pueden configurar las preferencias de uso de los recursos (ver anexo 8), pero no pueden decidir a qué proyectos apoyar (Lombraña 2010; Lombraña 2011).

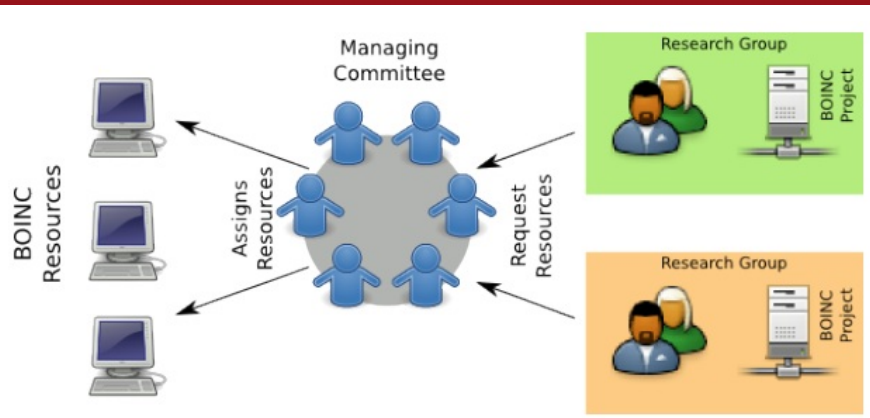

Ilustración 1.4: División conceptual de Jarifa (Lombraña 2010)

De lo dicho en el párrafo anterior, se tiene que un punto a favor de Jarifa es que define e indica claramente un conjunto de roles que desempeñarán los usuarios del sistema. Esto permite organizar y distribuir la carga de trabajo entre varias personas. Como se puede ver en la ilustración 1.5, se tiene: Root, el administrador general del sistema con acceso a todas las funcionalidades; Supplier, el proveedor o dueño de los recursos de cómputo que decide donar su capacidad ociosa; Allocator, el encargado de decidir en qué proyectos se utilizarán los recursos donados y Volunteer, un rol adicional que busca integrar al sistema a individuos, dueños de un solo recurso de cómputo, con deseos de apoyar a los proyectos.

Ilustración 1.5: Roles de Jarifa (Lombraña 2010)

A continuación se explica el funcionamiento de Jarifa como sistema de administración de cuentas.

> "The Allocator runs the Jarifa software on a server. The Suppliers run the BOINC client on their computers, and attach each client to the Allocator's account manager. The BOINC client on the computer periodically communicates with the Jarifa, which instructs it which projects to attach to, and the resource share for each attachment." (Lombraña 2010)

Tomando en cuenta el análisis anterior, se puede realizar una crítica al sistema Jarifa, desde el punto de vista de su adecuación para administrar un VCSC en la PUCP. Ésta es que si bien permite agrupar los recursos por proveedores (ver anexo 8), no permite realizar subdivisiones más finas, las cuales sí se requieren en el contexto universitario. Esto debido a que las áreas de la universidad pueden contener como subconjunto a los laboratorios y dentro de estos, a su vez, se puede agrupar a los nodos según características como tipo de procesador, estado de uso, entre otras.

1.3.3 World community grid

> "World Community Grid reúne personas de todo el mundo que donan el tiempo durante el cual sus computadoras están desocupadas para crear la mayor red de computación voluntaria en beneficio de la humanidad. Nuestro trabajo se basa en la convicción de que la innovación tecnológica combinada con la investigación científica visionaria y el trabajo voluntario a gran escala pueden ayudar a que el planeta sea más inteligente. " (IBM 2012).

World Community Grid es un proyecto de IBM, también basado en la plataforma BOINC, que aprovecha la computación en *grid* para apoyar proyectos científicos. Al igual que BAM! o Jarifa, se encarga de distribuir los recursos de cómputo entre los diversos proyectos. Para ello, dispone de servidores propios que hacen las labores de intermediarios entre ambos elementos, como se puede observar en la ilustración 1.6.

Ilustración 1.6: División conceptual de

World Community Grid (IBM 2012)

World Community Grid obtiene su poder de cómputo gracias a las personas en el mundo que deciden voluntariamente donar sus ciclos de CPU. Para promover su participación, World Community Grid utiliza dos estrategias. Primero, se asocia con negocios, asociaciones, fundaciones, organismos gubernamentales y universidades interesados en el tema. Estos se encargan de motivar a otras personas a colaborar con el proyecto. Segundo, posee un sistema de puntajes y desafíos por equipos. Los colaboradores se agrupan en equipos según países, instituciones o algún otro criterio común y compiten por lograr el mejor puntaje. A mayor cantidad de ciclos de CPU donados, mayor cantidad de puntos, mejor puesto en las estadísticas del desafío y mayor reconocimiento por parte del resto de colaboradores a nivel mundial (IBM 2012) (Ver anexos 9 y 10).

Para favorecer la retroalimentación, World Community Grid ofrece al usuario una variedad de reportes según proyecto, usuario, equipo o región geográfica. Estos contienen un resumen con el puesto, puntaje, cantidad de usuarios y de dispositivos relacionados al criterio elegido (IBM 2012) (Ver anexo 11).

1.3.4 Resumen comparativo de las soluciones existentes

En la tabla 1.1 se presenta un resumen de las tres soluciones descritas. Se puede observar que solo Jarifa permite aislar y registrar de manera directa proyectos propios de la organización. Adicionalmente, Jarifa señala explícitamente los roles que les competen a los usuarios del sistema. En los demás casos, estos deben ser deducidos. Por último, los tipos de reportes y gráficos utilizados se orientan a la cantidad de créditos, puntaje y FLOPS acumulados tanto por los proyectos como por los equipos y grupos de recursos, voluntarios o propios.

	Jarifa	BAM!	World Community Grid
Tipos de proyecto	Proyectos de la organización o externos.	Proyectos científicos predefinidos. Se puede solicitar la publicación de un proyecto propio.	Proyectos científicos predefinidos.
Origen de recursos	Voluntarios o recursos de la institución.	Voluntarios a nivel mundial.	Voluntarios a nivel mundial.
Sistema de calificación	Créditos.	Créditos.	Puntaje.
Roles definidos	Root, Supplier, Allocator y Volunteer.	No se indica de manera explícita.	No se indica de manera explícita.
Tipos de reporte y gráficos	Gráfico histórico de créditos y FLOPS por proyecto.	Reportes estadísticos detallados y gráficos comparativos de créditos entre proyectos y entre proveedores de recursos.	Reportes según proyecto, usuario, equipo o región geográfica

Tabla 1.1: Resumen de características de las soluciones existentes

1.4 Plan del proyecto

En este apartado se describe, primero, la metodología de gestión de proyectos a utilizar, justificando el porqué de su elección. A continuación, se presenta la descomposición de las tareas que comprende y el plan asociado a éstas.

1.4.1 Descripción y justificación de la metodología a utilizar

Para la gestión del presente proyecto de fin de carrera se hace uso de una adaptación del método para desarrollo de productos novedosos Stage-Gate, propuesto por Cooper y Edget (Cooper 2011). Esta adaptación se basa en los conocimientos adquiridos en el curso Temas Avanzados en Tecnología de Información 1. A continuación se detalla las etapas comprendidas y los pasos a seguir, justificando el porqué de su aplicación.

El modelo Stage-Gate para desarrollo de productos novedosos identifica un conjunto de actividades esenciales para el proceso de gestión del proyecto, a las que denomina *stages.* Al final de cada *stage*, se genera uno o más entregables con los resultados obtenidos. Luego, se realiza una evaluación de los mismos de acuerdo a un conjunto de criterios establecidos para decidir si el proyecto debe continuar, ser mejorado, reciclarse o anularse. Estos puntos de decisión son denominados *gates.* Conforme se avanza a través de los *stages* y *gates* el nivel de incertidumbre se reduce, a la vez que el nivel de inversión de tiempo, dinero y esfuerzo se incrementa. De esta manera se asegura un manejo adecuado del riesgo y una adecuada inversión de recursos (Cooper 2011). La ilustración 1.7 presenta un resumen de lo antes indicado.

Ilustración 1.7: Stages y Gates (Cooper 2011)

Se considera la metodología anterior como apropiada para el proyecto actual porque el producto a desarrollar es de carácter innovador. Esto último debido a que se propone tomar el concepto de un producto ya existente, pensado a escala mundial, para poder adaptarlo y hacerlo accesible a un contexto universitario. En este caso, de acuerdo a la clasificación de Cooper y Edget, se tiene una innovación de tipo disruptiva (Cooper 2007).

Como se puede ver en la ilustración 1.8, la innovación disruptiva supone el surgimiento de una nueva trayectoria para el desarrollo tecnológico del producto existente. Esta trayectoria inicialmente lleva a la elaboración de un producto con un desempeño inferior al primero, de acuerdo a los criterios de evaluación tradicionales, pero valioso para la minoría a la que va enfocado. De acuerdo a Cooper y Edget, esta deficiencia inicial se supera con el paso del tiempo (Cooper 2007).

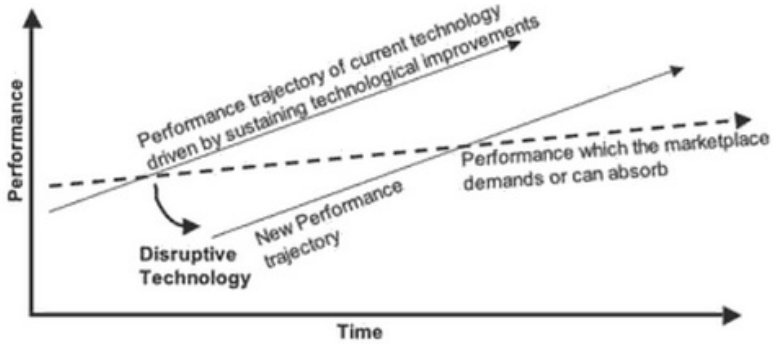

Ilustración 1.8: Innovación disruptiva (Cooper 2007)

Para el caso de este proyecto de fin de carrera, se tiene un primer *stage* de descubrimiento, en el que se elige el producto a realizar. Para esto es necesario, con ayuda del asesor, plantear un conjunto de alternativas basadas en las necesidades de innovación de la DIA. Al final del *stage,* se tendrá una lista de posibles aplicaciones a desarrollar, que luego serán revisadas en el *gate* para decidir cuál de ellas debe ejecutarse (Ver anexo 12).

En el segundo *stage,* alcance, se realiza una rápida evaluación de las soluciones ya existentes en el mercado, con el objetivo de identificar una oportunidad de innovación en los aspectos que éstas no cubren. En el *gate* correspondiente, se

decide si realmente el proyecto genera un valor adicional, teniendo en consideración la experiencia del asesor en este tipo de proyectos y las opiniones del representante del cliente; en este caso, el administrador de la *grid*.

A continuación, en el tercer *stage,* construcción del caso de negocio, se elabora la definición y justificación del producto y el plan de proyecto. Para el caso de la definición del producto, se utiliza el método de prototipos según Cerejo (Cerejo 2010); éste se explica en el siguiente capítulo. El documento resultante es enviado tanto al representante del cliente como al asesor para ser evaluado en el *gate* y decidir si se continúa con la siguiente etapa. En el presente proyecto quien representa al cliente es el administrador actual de la *grid* de la PUCP.

Luego, en el cuarto *stage,* una vez definido y aprobado el proyecto, se desarrolla el producto. En este caso, se ha decidido tomar algunas de las buenas prácticas de la metodología *Extreme Programming*, la cual se detalla y justifica en el siguiente capítulo. En el *gate* de esta etapa se debe probar el producto finalizado.

Como último stage se considera la prueba y validación final de todo el proyecto. Para ello, se analiza los resultados de las pruebas junto con el asesor y el cliente. A partir de lo anterior, en el último gate, se decide si el proyecto de fin de carrera puede ser considerado exitoso. La ilustración 1.9 muestra un esquema con los *stages* y *gates* que se decide aplicar al presente proyecto de fin de carrera.

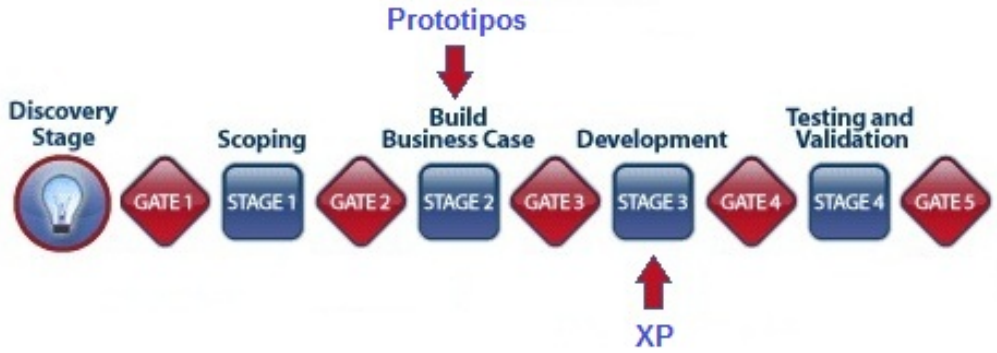

Ilustración 1.9: Stages y gates elegidos de la metodología Stage-Gate (Adaptado de Cooper 2011)

1.4.2 Estructura de descomposición del trabajo y diagrama de Gantt

En este apartado se presenta, primero, el conjunto de productos o entregables que se debe realizar por cada *stage* que abarca el proyecto. Para ello, se utiliza una estructura de descomposición del trabajo (EDT). Luego, se muestra los plazos para cada una de las tareas implicadas en un diagrama de Gantt.

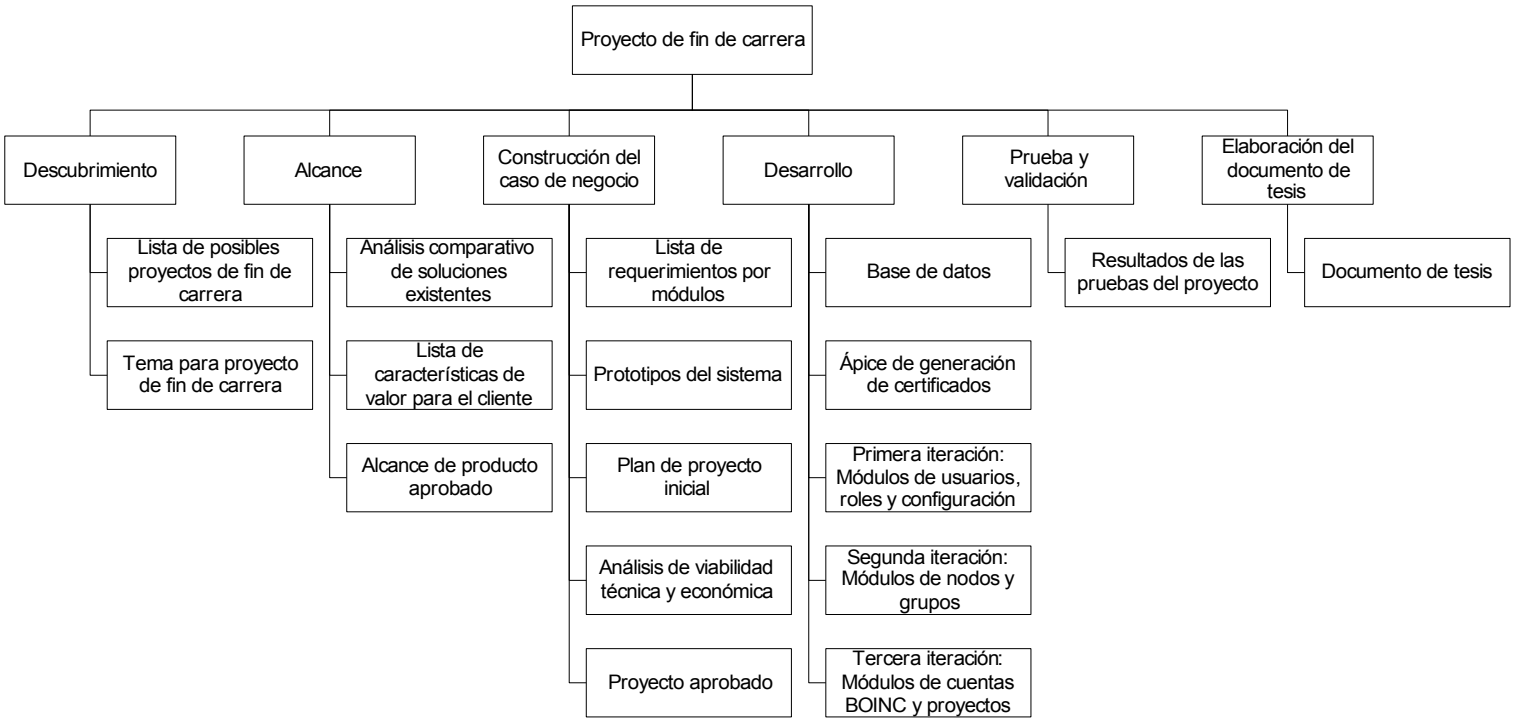

Ilustración 1.10: EDT del proyecto de fin de carrera

En la tabla 1.2 se presenta el diagrama de Gantt del proyecto, con los tiempos y plazos asignados. El diagrama comprende las tareas requeridas para la ejecución del proyecto de fin de carrera. En su estimación se ha tomado en cuenta los tiempos de transporte, concertación de reuniones y esperas, carga académica paralela al desarrollo del proyecto y periodos de vacaciones.

Como se puede notar, el proyecto abarca un total de 1571 horas de trabajo, distribuidas a lo largo de 302 días. La etapa con mayor consumo de tiempo es el desarrollo. Por otra parte, es necesario aclarar que, en la etapa mencionada, la entrega de un módulo incluye tanto las tareas de la iteración como la ejecución de pruebas de integración (PI), revisión y corrección del mismo.

Cabe resaltar que el plan mostrado en la tabla 1.2 difiere del elaborado inicialmente, el cual se puede observar en el anexo 13. Las razones de este hecho se explican en el capítulo 5: Observaciones y Conclusiones.

	Nombre de tarea	Trabajo	Duración	Comienzo	Fin
1	**Proyecto de fin de carrera**	**1,571 horas**	**302 días**	**lun 15/08/11**	**lun 08/10/12**
2	**Descubrimiento**	**19 horas**	**9 días**	**lun 15/08/11**	**jue 25/08/11**
3	Generar propuestas de aplicaciones a desarrollar	15 horas	6 días	lun 15/08/11	lun 22/08/11
4	Evaluar y elegir propuesta	4 horas	3 días	mar 23/08/11	jue 25/08/11
5	**Alcance**	**18 horas**	**11 días**	**vie 26/08/11**	**vie 09/09/11**
6	Investigar soluciones ya existentes	10 horas	10 días	vie 26/08/11	jue 08/09/11
7	Elaborar lista de características valiosas para el cliente	7 horas	10 días	vie 26/08/11	jue 08/09/11
8	Aprobar producto	1 hora	1 día	vie 09/09/11	vie 09/09/11
9	**Construcción del caso de negocio**	**39 horas**	**53 días**	**lun 12/09/11**	**mié 23/11/11**
10	Elaborar lista de requerimientos por módulos	6 horas	45 días	lun 12/09/11	vie 11/11/11
11	Elaborar prototipos del sistema	15 horas	42 días	jue 15/09/11	vie 11/11/11
12	Elaborar plan de proyecto inicial	8 horas	5 días	jue 13/10/11	mié 19/10/11
13	Realizar análisis de viabilidad técnica y económica	8 horas	7 días	jue 20/10/11	vie 28/10/11
14	Aprobar proyecto	2 horas	1 día	mié 23/11/11	mié 23/11/11
15	**Desarrollo**	**1,365 horas**	**213 días**	**lun 12/12/11**	**mar 02/10/12**
16	Definir estándares de base de datos	6 horas	1 día	lun 12/12/11	lun 12/12/11
17	Realizar diseño inicial de base de datos	26 horas	3 días	mar 13/12/11	jue 15/12/11
18	Mapear lista de transacciones a requerimientos	2 horas	1 día	mar 13/12/11	mar 13/12/11
19	Elegir framework de base de datos a utilizar	1 hora	1 día	mié 14/12/11	mié 14/12/11
20	Programar ápice de generación de certificados	40 horas	21 días	vie 16/12/11	vie 13/01/12
21	Programar módulos de usuarios y roles	372 horas	60 días	lun 16/01/12	vie 06/04/12
22	Definir estándares de programación	6 horas	1 día	lun 05/03/12	lun 05/03/12
23	Programar y ejecutar PI de módulos de usuarios y roles	60 horas	26 días	vie 02/03/12	vie 06/04/12
24	Programar módulo de configuración del sistema	24 horas	6 días	lun 09/04/12	lun 16/04/12
25	Programary ejecutar PI de módulo de configuración	12 horas	5 días	mar 10/04/12	lun 16/04/12
26	Programar módulos de nodos y grupos	500 horas	88 días	mar 17/04/12	mié 15/08/12
27	Programar y ejecutar PI de módulos de nodos y grupos	60 horas	14 días	lun 23/07/12	jue 09/08/12
28	Programar módulo de proyectos	176 horas	22 días	jue 16/08/12	vie 14/09/12
29	Programar módulo de cuentas BOINC	48 horas	8 días	lun 17/09/12	mié 26/09/12
30	Programa y ejecutar PI de módulo de proyectos	16 horas	2 días	jue 27/09/12	vie 28/09/12
31	Programar y ejecutar PI de módulo de cuentas BOINC	16 horas	2 días	lun 01/10/12	mar 02/10/12
32	**Prueba y validación**	**20 horas**	**3 días**	**mié 03/10/12**	**vie 05/10/12**
33	Realizar pruebas finales del producto	20 horas	3 días	mié 03/10/12	vie 05/10/12
34	**Elaboración del documento de tesis**	**110 horas**	**267 días**	**lun 03/10/11**	**lun 08/10/12**
35	Elaborar versión inicial de los capítulos 1 y 2	10 horas	30 días	lun 03/10/11	vie 11/11/11
36	Elaborar segunda versión de los capítulos 1 y 2 e inicial del 3	20 horas	15 días	lun 12/03/12	vie 30/03/12
37	Elaborar segunda versión del capítulo 3 e inicial del 4 y 5	30 horas	10 días	lun 02/04/12	vie 13/04/12
38	Elaborar versión de fin de curso del documento de tesis	10 horas	6 días	lun 16/04/12	lun 23/04/12
39	Corregir versión de fin de curso del documento de tesis	10 horas	2 días	dom 17/06/12	lun 18/06/12
40	Elaborar versión final del documento de tesis	30 horas	12 días	vie 21/09/12	lun 08/10/12

Tabla 1.2: Diagrama de Gantt

1.5 Descripción y sustentación de la solución

Recapitulando, hay una dificultad en las labores de administración de la *grid* debido a su tamaño. Es por ello que actualmente se utiliza el sistema Jarifa. Sin embargo, éste no satisface en un nivel suficiente los requerimientos de automatización e información del administrador de la *grid.* A continuación, se presenta una descripción del sistema a desarrollar en este proyecto, en adelante Pretor, enfatizando las mejoras que incluye en comparación con el sistema actual utilizado.

Primero, el sistema Pretor reduce el tiempo y esfuerzo humano que toma asignar los nodos a los proyectos científicos de la PUCP. En el escenario en que no se tiene este tipo de sistema, el administrador de la *grid* debe configurar las preferencias de uso de cada computadora y registrar uno por uno los proyectos a los que brindará su poder de cómputo. Entonces, considerando, como ya se dijo, que la *grid* de la PUCP está conformada por, aproximadamente, 500 nodos, esta tarea se vuelve indeseable. Por lo tanto, el sistema Pretor se implementa como un sistema de administración de cuentas para poder solucionar este problema. Lo que hace es centralizar la información sobre los proyectos y recursos y ofrecer al administrador de la *grid* una interfaz *web* para asociarlos. Entonces, el administrador puede realizar esta labor desde cualquier equipo informático con acceso al sistema, eliminando así la necesidad de desplazarse hacia cada uno de los nodos de la *grid.*

Adicionalmente, como complemento del punto anterior, el sistema Pretor permite, a diferencia de Jarifa, agrupar los recursos de acuerdo a un conjunto de criterios definidos por el administrador de la *grid.* Estos criterios toman como referencia las características de *hardware* de las computadoras, que Pretor muestra con mayor detalle que Jarifa. Esto agiliza las tareas de configuración de preferencias de uso y asignación de proyectos, ya que en vez de realizarlas para cada recurso, se pueden realizar por grupos. Además, los grupos son considerados útiles por el administrador de la *grid,* ya que le permiten situaciones como agrupar un conjunto de recursos con procesadores de alto poder de cómputo y asignarlos a los proyectos más urgentes; agrupar un conjunto de recursos defectuosos y evitar que se les asigne proyectos; agrupar los recursos de acuerdo al laboratorio de la universidad en que se encuentran y configurar sus horas de uso de modo que no

coincidan con las horas de clases, para que no interfieran con los programas de los alumnos, entre otras.

Segundo, una vez registrados y organizados los nodos y proyectos, el sistema Pretor permite mantener una comunicación periódica con los mismos. Esta característica es útil para monitorizar el estado de actividad de cada nodo y proyecto, de modo que, en caso de un comportamiento no esperado, el administrador de la *grid* pueda ejecutar una respuesta adecuada. Por otra parte, de haber un cambio en la asignación de los nodos o grupos de nodos a los proyectos, el sistema Pretor se encarga de difundir automáticamente los datos modificados a todos los nodos involucrados. También, gracias a la comunicación continua, es posible contar con datos recientes del nivel de apoyo de una cuenta a un determinado proyecto o del nivel de consumo de los proyecto de una cuenta; todo esto medido en unidades de créditos, como lo determina BOINC. Esta función se puede utilizar, para, como en el caso de *World Community Grid,* fomentar la competencia y mayor involucramiento de las áreas académicas y obtener mayor cantidad de recursos.

Tercero, el sistema Pretor proporciona reportes detallados, ausentes en Jarifa, basados en los datos actualizados a los que se hace referencia en el párrafo anterior. El objetivo de los reportes es satisfacer los requerimientos de información del administrador de la *grid*. Estos reportes incluyen datos sobre los nodos y proyectos del sistema. Dado el alcance del proyecto de fin de carrera, el sistema Pretor no comprende reportes sobre la evolución histórica del desempeño de los recursos. Tampoco se tiene gráficos comparativos ni históricos. Sin embargo, su posterior inclusión se discute en el apartado de trabajos futuros.

Cuarto, el sistema Pretor incluye un sistema de roles, basado en el esquema de Jarifa, que permite al administrador principal dividir y compartir su labor con otros administradores. Esto evita la concentración excesiva de tareas en una sola persona, como ocurre actualmente, debido al crecimiento de la *grid*.

Por último, el sistema Pretor permite englobar a todos los miembros de una misma unidad de la PUCP dentro de una entidad denominada cuenta BOINC. Gracias a ello se logra que los usuarios del sistema Pretor tengan mayores privilegios sobre

los proyectos y nodos asociados a su cuenta y puedan colaborar también con proyectos de otras cuentas. Cabe resaltar que, si bien esta idea surge de una de las reuniones con el asesor, se sabe que la Universidad de Westminster, mencionada antes como ejemplo de VCSC, define una entidad similar denominada *Virtual Organization* (P-GRADE Portal 2010).

En conclusión, el sistema Pretor, al igual que la solución actualmente utilizada, permite un ahorro considerable de tiempo y esfuerzo en las labores de administración de la *grid*. Adicionalmente, incluye un conjunto de cambios, de acuerdo a lo requerido por el contexto particular de la PUCP. Estos están referidos a la presentación de mayor cantidad de datos para agrupar a los recursos; a la adición de reportes más detallados; a la distribución de la carga administrativa y a la inclusión del concepto de cuenta BOINC.

2 Análisis

Este capítulo tiene como objetivo realizar el análisis del sistema que se desarrolla como proyecto de fin de carrera. Para ello, se presenta primero la metodología utilizada para el desarrollo del sistema. A continuación, la lista de requerimientos funcionales y no funcionales del sistema, la forma en que fueron obtenidos y la explicación de cómo es que resuelven el problema identificado en el capítulo anterior. Por último, se realiza un análisis de la viabilidad del sistema en términos de los recursos que necesita y se concluye con una breve descripción del mismo.

2.1 Metodología aplicada para el desarrollo de la solución

Para el desarrollo del sistema que se trata en el presente proyecto se utiliza una adaptación de la metodología XP, tomando en cuenta las sugerencias del asesor, los requerimientos del cliente y las propias características del proyecto. A continuación, se justifica el porqué de la elección de XP y se explica detalladamente cómo y cuáles de sus prácticas se utilizan.

2.1.1 Justificación de la elección de XP como metodología base

En el artículo *The New Methodology*, Fowler (Fowler 2005) menciona que, en la construcción de *software* para negocios, es normal que se den cambios en los requerimientos. Según él, para algunos, estos cambios son considerados el resultado de una pobre ingeniería de requerimientos. Sin embargo, después de presentar su razonamiento, llega a la conclusión de que esto último no es así. Por tanto, las personas deben asumir el cambio.

A continuación, Fowler expande la validez del planteamiento de que los requerimientos son cambiantes, más allá del mundo de los negocios, a escenarios donde es difícil predecir con seguridad. Él afirma que, aun bajo estos escenarios, las personas intentan aplicar metodologías que describen un proceso predecible, cuando en realidad esto es incorrecto, ya que cae más allá de sus límites.

Por último, para estos escenarios poco predecibles Fowler recomienda el uso de las denominadas metodologías ágiles. Entre sus ejemplos se tiene a las metodologías XP, Scrum, Crystal, Lean Development, entre otras.

Una vez establecido lo anterior, se ha decidido, en primera instancia, que el presente proyecto debe utilizar una metodología ágil, dado su carácter innovador disruptivo, el cual se justifica en el apartado 1.4.1 de este documento. Según Cooper (Cooper 2007), la innovación disruptiva se caracteriza por no ser predecible: surge en un escenario de constante cambio tecnológico, propone una dirección paralela para el cambio y aprovecha las oportunidades que surjan en este escenario.

En particular, en este proyecto, se utiliza el *middleware* BOINC y, como se indica en el marco conceptual, este último es un proyecto vigente. Ello implica que BOINC cambia a un ritmo periódico conforme se generan nuevas versiones. Este hecho, adicionado al proceso de descubrimiento de las oportunidades que ofrece el mecanismo administrador de cuentas de BOINC, trae como consecuencia el escenario de cambio planteado por Cooper.

Por otra parte, existen actualmente muchas metodologías disponibles, tanto ágiles como no ágiles. Para la elección de una de ellas, se ha tomado en cuenta el

resumen elaborado por Abrahamsson junto con otros autores (Abrahamsson et al. 2002). Éste se puede ver en el anexo 14

Se ha elegido basarse en XP por las siguientes razones:

1. XP es una metodología ágil de desarrollo de *software* (Jeffries 2012).
2. El equipo del proyecto está conformado por tres integrantes: el encargado del proyecto, el asesor y el administrador de la *grid*. XP está orientado a equipos pequeños (Abrahamsson et al. 2002).
3. El administrador de la *grid* se compromete a participar de manera diaria a lo largo de todo el desarrollo, cuidando que el producto realmente satisfaga las necesidades de la PUCP en el tema de administración de la *grid*. XP aprueba el desarrollo orientado hacia el cliente (Abrahamsson et al. 2002).
4. Se cuenta con información disponible de manera gratuita sobre XP y sus casos de aplicación práctica (Abrahamsson et al. 2002).
5. El proyecto debe permitir cambios en los requerimientos, dada su naturaleza de innovación disruptiva. XP permite estos cambios gracias al *refactoring* (Abrahamsson et al. 2002).
6. Si bien XP no presta mucha atención a los temas de gestión del proyecto, estos serán cubiertos por la metodología Stage-Gate.

2.1.2 Prácticas de XP elegidas

A continuación se listan las buenas prácticas de programación que se ha decidido utilizar para la elaboración del *software*. Estas han sido tomadas del artículo *What is Extreme Programming?* (Jeffries 2012), elaborado por uno de los creadores de XP, Ronald Jeffries.

a) Equipo completo

XP recomienda que todo el equipo de proyecto debe trabajar junto, en un solo lugar. Este equipo debe contemplar los siguientes roles y funciones:

- El cliente, que provee los requerimientos, define su prioridad y guía el proyecto. Se recomienda que sea un usuario final del sistema.
- Programadores o desarrolladores, quienes construyen el sistema.
- *Testers*, quienes ayudan al cliente a definir las pruebas de aceptación.
- Analistas, quienes ayudan al cliente a definir sus requerimientos.

- Un *coach*, que ayuda al equipo a mantener el ritmo de desarrollo.
- Opcionalmente, un *manager*, quien provee los recursos necesarios para el desarrollo.

Para el caso de este proyecto, el lugar de trabajo es la oficina del área de infraestructura de la DIA. Aquí se reúnen diariamente el representante del cliente y el encargado del proyecto. El representante asumirá los roles de cliente, *coach* y *manager*, antes mencionados. El encargado del proyecto asumirá los roles de programador, *tester* y analista. Cabe resaltar también que el cliente es el administrador de la *grid,* es decir, un usuario final, tal y como recomienda XP.

Por otro lado, en el caso particular de la DIA, se cuenta con dos asesores ocasionales en temas de desarrollo *web.* Uno de ellos, miembro del proyecto Legión, mencionado en el marco conceptual. Ambos cumplen el rol de contribuir con sugerencias de mejora al desarrollo del sistema y brindar asesoría en temas técnicos.

b) Juego de planificación

XP recomienda realizar la planificación del desarrollo tanto para poder predecir qué será cumplido para la fecha límite, como para determinar qué hacer luego. Sin embargo, el plan no se debe tomar como una predicción exacta del tiempo a utilizar, pues esto es muy difícil de conseguir; sino más bien, como una guía que ayuda a mantener el paso al proyecto. A continuación se presenta los dos pasos que requiere la planificación en XP:

- Planificación del *release*, se da cuando el cliente presenta sus requerimientos y los programadores estiman el esfuerzo requerido. En base a esto, el cliente traza el plan del proyecto. Empero, este plan es, según Jeffries, necesariamente impreciso, por lo que el equipo debe revisarlo y ajustarlo constantemente.

- Planificación de la iteración, se da cuando el cliente presenta el conjunto de requerimientos deseado a desarrollar en las siguientes dos semanas. Los programadores, basados en su experiencia de iteraciones anteriores, deben estimar y ajustar los requerimientos que se contemplarán. Se espera que

ellos entreguen *software* ejecutable y útil al final de este tiempo. Esto permite, por un lado, que el cliente pueda cancelar el proyecto si el progreso no es adecuado y, por otro, que se tienda a realizar el proyecto con menor presión y estrés.

Para el caso de este proyecto, se realiza una primera planificación completa del *release* al inicio del proyecto y, otra, hacia la mitad del mismo. Además, se revisa la planificación al inicio de cada iteración, para ajustarla a las decisiones tanto del cliente como del encargado del proyecto.

Con respecto a la planificación de la iteración, se ha modificado la recomendación de XP, en acuerdo mutuo con el cliente. La longitud de la iteración se estima inicialmente en un mes y puede extenderse tanto como sea necesario para completar todos los requerimientos que comprende. El motivo de esto es que tanto el cliente como el encargado del proyecto están de acuerdo en dar mayor importancia al hecho de que se cubra todos los requerimientos asociados a la iteración que al tiempo que esto implique. Sin embargo, conforme con lo indicado por XP, se espera que el encargado del proyecto sea capaz de estimar mejor sus tiempos conforme avanza en el desarrollo del sistema.

c) Pruebas de cliente

XP recomienda que el cliente defina una o más pruebas de integración automatizadas para probar que el producto funciona. Los programadores son los encargados de codificar las pruebas. Éstas deben ser automatizadas, dado que, con la presión del tiempo, las pruebas manuales se tienden a dejar de lado.

Para el presente proyecto, el cliente define sus pruebas como historias de usuario simples, con ayuda del encargado, indicando los casos que deben ser probados. Adicionalmente se considera, para el tema de pruebas de formularios de ingreso de datos, la estrategia de clases equivalentes propuesta por IBM (Myers en IBM 2009). También, para automatizar las pruebas, se utiliza un conjunto de herramientas *open source*. Esto se detalla en el capítulo 4.

d) Lanzamientos pequeños

XP recomienda que se entregue frecuentemente *software* que pueda ser utilizado por el cliente en su negocio. La frecuencia puede ir desde días hasta trimestres.

Esta práctica no se utiliza en el proyecto, dado que se tiene un escenario distinto al de los negocios. Esto es, se busca desarrollar un sistema innovador enfocado al ámbito universitario y científico. El cliente acepta que, bajo este escenario, es aceptable tener un periodo de espera mayor, no tan exigente como en el caso de los negocios.

e) Diseño simple

XP recomienda revisar el diseño constantemente, a lo largo de todo el desarrollo del proyecto, para poder mantenerlo simple y, a la vez, mejorarlo. En particular, se revisa al inicio de las planificaciones del *release* y de la iteración y en las sesiones diarias con el equipo de desarrollo.

Un diseño simple se caracteriza por: evitar el código duplicado, una alta cohesión y un bajo acoplamiento.

f) Refactorización

XP propone la técnica de refactorización, que consiste en revisar y modificar el producto elaborado con el objetivo de mantener el diseño simple. La refactorización se soporta en las pruebas automatizadas, que los programadores utilizan para asegurarse que el sistema no vea afectado su funcionamiento.

En el proyecto, se aplican las prácticas e) y f) tal y como fueron descritas. Se revisa el diseño constantemente con el cliente, al inicio de cada iteración y en las sesiones diarias. Se refactoriza el código y se ejecutan las pruebas para comprobar su corrección.

g) Programación en parejas

XP recomienda construir todo el código en parejas de programadores. Cada pareja utiliza una sola computadora, de forma que mientras una persona programa, la otra se encarga de revisar el código. Esto asegura un mejor diseño, mejores pruebas y mejor código.

Esta práctica no se utiliza tal cual porque este proyecto de fin de carrera fue concebido desde un inicio para ser manejado por un solo desarrollador. Sin embargo, el propio cliente asume la tarea de revisar ocasionalmente el código desarrollado, con el fin de obtener los beneficios que indica XP.

h) Desarrollo orientado por pruebas

De acuerdo con Jeffries, los mejores equipos XP tienen en cuenta esta práctica. Es, en líneas generales, una forma de trabajo en la cual los programadores desarrollan el sistema en ciclos cortos, que consisten en, primero, añadir una prueba y, segundo, generar código que permita superarla. Aplicada de manera continua, esta práctica genera retroalimentación inmediata sobre su desempeño a los programadores. Además, se convierte en un soporte importante para la refactorización.

En el proyecto esta práctica no se utiliza siempre como se ha descrito. En primer lugar, porque durante la primera iteración se tiene una etapa de aprendizaje del uso de las herramientas de automatización, lo que se hace creando pruebas sobre módulos ya programados. En segundo lugar, durante la segunda iteración, las pruebas sí se construyen y ejecutan de manera paralela a la codificación, lo cual ayuda a garantizar la correción de los mensajes que se envía a los nodos. Para terminar, durante la tercera iteración, se decide construir las pruebas al final, con el objetivo de acelerar el paso del desarrollo.

i) Integración continua

XP recomienda mantener el sistema integrado siempre. Esta práctica busca evitar ciertos problemas usuales en la integración infrecuente: código con errores que solo surgen durante la integración y retrasos por esperas a que todos los módulos de programadores concluyan.

Esta práctica es idónea para el presente proyecto, puesto que se cuenta con un solo programador. Entonces, todo el código que éste desarrolle estará siempre integrado.

j) Posesión colectiva del código

XP recomienda que, en un proyecto, cualquier par de programadores pueda mejorar el código en cualquier momento. Es decir, es posible alterar el código elaborado por otros, con el objetivo de mejorarlo. Esto da como resultado una mejora en la calidad del código y reducción de los defectos.

No se utiliza esta práctica dado que este proyecto solo cuenta con un programador.

k) Estándares de programación

XP recomienda definir estándares de programación que permitan hacer que todo el código luzca como si hubiera sido desarrollado por una sola persona. Esto da soporte a la práctica anterior, ya que facilita que todos los programadores entiendan el código del resto.

Este proyecto se ciñe a esta práctica e incluye un documento de estándares de programación, basado tanto en las recomendaciones de instituciones reconocidas en la comunidad de programadores Java como en prácticas adquiridas gracias a la experiencia en proyectos anteriores. Éste se puede ver en el anexo 15.

l) Metáfora

XP recomienda que el equipo conciba una visión general de cómo el sistema trabaja, mediante una metáfora. En algunos casos, esta puede consistir de un enunciado simple o de una imagen.

Como parte de este proyecto se ha elaborado un diagrama simple, presentando a alto nivel las entidades involucradas y las relaciones entre ellas. Este se puede ver al final del capítulo y se considera la metáfora general del sistema.

m) Paso sostenible

XP reconoce que un proyecto de desarrollo de sistemas toma un tiempo considerable. Durante este lapso, los equipos trabajan de manera ardua y sostenida. Por lo tanto, se debe mantener un ritmo que garantice, por un lado, la continuidad del proyecto y, por otro, el bienestar de las personas.

En este proyecto, el desarrollador dedica 8 horas diarias a la elaboración del producto, durante 5 días a la semana, en la época de vacaciones. Durante la época de clases, dedica 6 horas diarias a la elaboración del producto, 5 días a la semana, y 4 horas a la elaboración del documento de proyecto de fin de carrera, 3 días a la semana.

La ilustración 2.1 muestra el resumen de las 13 prácticas de XP antes mencionadas:

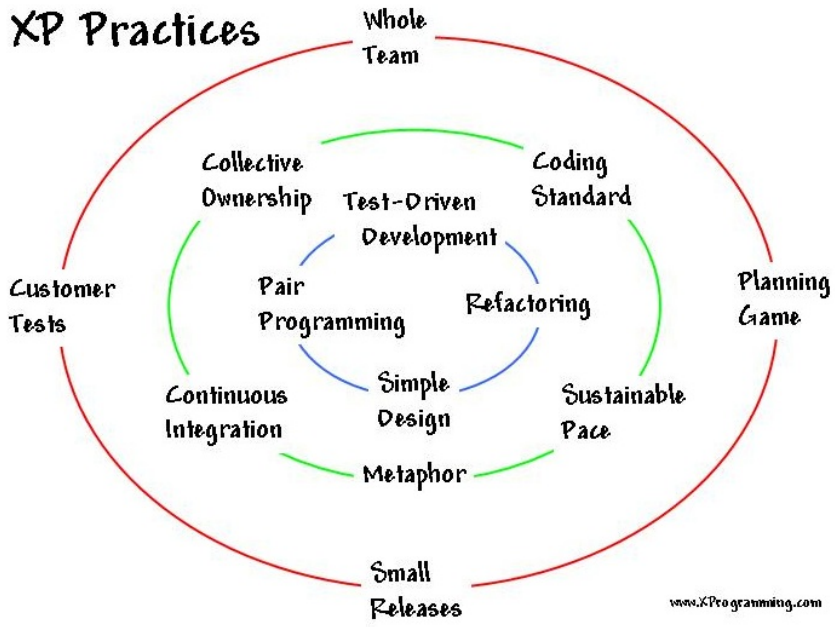

Ilustración 2.1: Prácticas de XP (Jeffries 2012)

2.2 Identificación de requerimientos

En esta sección se indica la forma en que se obtuvo los requerimientos para el producto a desarrollar. A continuación, se lista los requerimientos obtenidos, tanto funcionales como no funcionales.

2.2.1 Toma de requerimientos

La toma de requerimientos inicial para el sistema Pretor se realiza concertando un conjunto de reuniones con el representante del cliente, en las que se utiliza la técnica de “lluvia de ideas”. Además, de manera paralela, se desarrolla un prototipo que es refinado al finalizar cada reunión, con el objetivo de validar los

requerimientos obtenidos y listados, siguiendo la metodología propuesta por Cerejo (Cerejo 2010) (ver ilustración 2.2).

Son varias las razones por la que se eligió esta forma de toma de requerimientos. Primero, debido a que se conciertan reuniones registradas en actas simples (ver anexo 15), permite a los involucrados organizar adecuadamente su tiempo, pensar previamente los temas que se desean tratar y llevar un registro de lo ya tratado. Segundo, dado el carácter innovador del producto, la técnica de "lluvia de ideas" permite identificar aquellas que le agreguen mayor valor. Tercero, los prototipos facilitan la discusión con material visual, en vez de palabras; aseguran que los involucrados tengan un entendimiento común del producto, reducen el riesgo y ayudan a capturar mejor los requerimientos (Cerejo 2010). Por último, la lista de requerimientos permite mantener un registro actualizado de las características del sistema a desarrollar.

Por otra parte, cabe resaltar que se decide no utilizar las historias de usuario usualmente asociadas a XP para la toma de requerimientos dado que se comprueba empíricamente que los prototipos permiten un mejor entendimiento con el cliente. La lista de requerimientos se convierte, entonces, en una herramienta auxiliar que permite al programador organizar su trabajo.

Además, si bien durante esta fase no se utiliza historias de usuario, se ve por conveniente utilizarlas en la fase de pruebas, puesto que facilitan la gestión de las pruebas de integración. Esto se detalla en el capítulo 4: Construcción y Pruebas. A continuación se detalla los pasos del proceso de toma de requerimientos.

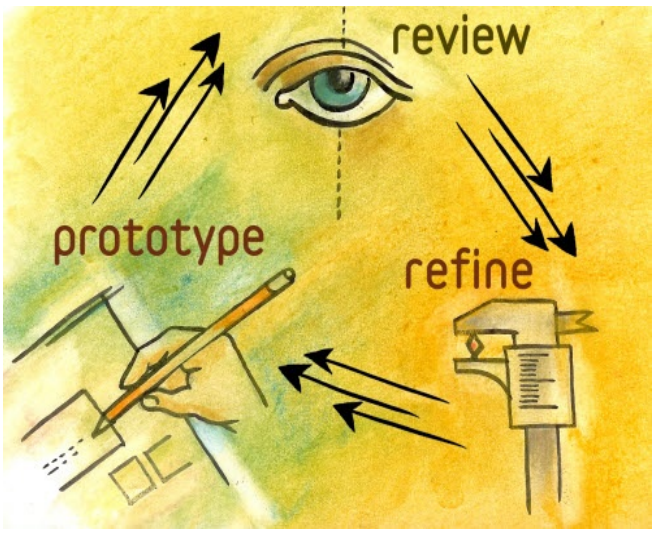

Ilustración 2.2: Metodología de prototipos

(Cerejo 2010)

El primer paso es la preparación previa a la reunión. Primero, se acuerda una fecha y hora para la reunión a través de correo electrónico. Considerando la disponibilidad del representante del cliente y con el objetivo de no interrumpir sus tareas, esta puede ser postergada hasta con un día de anticipación, por el mismo medio. Segundo, es necesario realizar la confirmación de la reunión el día anterior, en caso el representante del cliente olvide agregarla a su calendario o surja algún suceso imprevisto. También, se elabora un acta de reunión con los temas a tratar y la lista de asistentes, que se envía previamente por correo electrónico.

El segundo paso es realizar la reunión. El acta de reunión antes indicada contiene una lista base de temas a tratar. Sin embargo, dado que se utiliza la técnica de "lluvia de ideas" para obtener los requerimientos, es posible incluir nuevos temas que puedan surgir durante la reunión. La información obtenida es anotada por el entrevistador manualmente en un soporte adecuado. Además, se evalúa el prototipo preparado con los requerimientos obtenidos en la reunión anterior y se apuntan, también, las correcciones que sugiera el representante del cliente.

Es de notar que una forma paralela de obtener requerimientos consiste en realizar la evaluación de las soluciones ya existentes indicadas en la sección de estado del arte. A partir de ellas se obtiene una lista de funcionalidades y características que

también son propuestas al representante del cliente y al asesor durante las reuniones.

Por otra parte, si bien se acuerda la hora de inicio de la reunión, la hora de fin dependerá de la disponibilidad de horarios de los involucrados y de su nivel de compromiso con el proyecto. En general, en este caso particular, las reuniones se dan por terminadas, en la mayoría de veces, cuando ninguno de los presentes tiene más ideas por presentar.

El tercer paso es la preparación posterior a la reunión. En esta etapa se transcribe a un formato digital los apuntes manuales realizados y se almacenan en un lugar que garantice su persistencia y facilite su recuperación. Adicionalmente, se corrige el prototipo con los nuevos cambios que solicite el representante del cliente para poder presentarlo en la siguiente reunión. Las actas de reunión y los prototipos elaborados se pueden ver en los anexos 16 y 17.

Cabe resaltar también que los requerimientos que se obtiene inicialmente y son aprobados en el *stage* de construcción del caso de negocio pueden ser modificados, de acuerdo a lo propuesto por XP. Es por ello que, en las actas de reunión del anexo 16, se puede notar la modificación de requerimientos inclusive durante el *stage* de desarrollo. Para las reuniones que se realiza durante este último *stage*, considerando el contacto diario que se tiene con el cliente, no se toma en cuenta las formalidades mencionadas en párrafos anteriores.

2.2.2 Objetivo general, objetivos específicos y resultados esperados

En la tabla 2.1 se presenta el objetivo general del proyecto de fin de carrera, junto con los objetivos específicos (OE) en que se soporta. Adicionalmente, se incluye los resultados esperados (RE) para cada objetivo.

Objetivo general: Realizar el análisis, diseño e implementación de un sistema administrador de recursos para la *grid* computacional de la PUCP, basado en el middleware BOINC.	
OE1: Identificar tres sistemas que hagan uso del mecanismo administrador de cuentas del *middleware* BOINC, utilizarlos y determinar las principales funcionalidades que ofrecen para tomarlas como referencia en el sistema a desarrollar.	RE1: Descripción de las funcionalidades de los tres sistemas identificados.
OE2: Elaborar los prototipos de pantallas que muestren las funcionalidades que ofrecerá el sistema a implementar.	RE2: Prototipos de pantallas del sistema.
OE3: Reutilizar y adaptar la arquitectura del sistema de código abierto Legión de modo que soporte la interacción entre los recursos relacionados a la *grid*.	RE3: Arquitectura del sistema Pretor.
OE4: Estudiar y utilizar un conjunto de herramientas de desarrollo web, junto con el *middleware* BOINC, para realizar la implementación del sistema.	RE4: Sistema Pretor implementado.
OE4: Construir y ejecutar pruebas automatizadas para detectar la mayor cantidad de errores posibles en el sistema.	RE5: Pruebas automatizadas del sistema Pretor.

Tabla 2.1: Objetivo general, objetivos específicos y resultados esperados

2.2.3 Requerimientos funcionales

La tabla 2.2 muestra los requerimientos funcionales que cubre el sistema Pretor. Conforme a la metodología XP, es posible que estos varíen de manera razonable durante el desarrollo del proyecto, de acuerdo a lo que necesite el cliente. En el anexo 20 se puede observar la lista inicial con los requerimientos funcionales del sistema. Por otra parte, la columna de historias asociadas se refiere a las utilizadas para realizar las pruebas de integración del sistema. Esto se detalla en el capítulo 4.

Cód	Requerimiento funcional	Historias asociadas	Actores asociados
Módulo de usuarios			
RF01	El sistema Pretor permitirá el inicio de sesión mediante usuario y contraseña.	1.1, 1.2, 1.3	Administrador, Proveedor y Registrador
RF02	El sistema Pretor permitirá recuperar la contraseña de un usuario, en caso de olvido, por vía correo electrónico.	1.4	Administrador, Proveedor y Registrador
RF03	El sistema Pretor permitirá mantener registros con el identificador, nombres, apellidos, correo electrónico, dirección, teléfono, lenguaje preferido, rol, área académica y estado del permiso de ingreso de los usuarios (activo o suspendido). Se enviará un correo electrónico al usuario notificándole que su registro fue creado o modificado.	1.5, 1.6, 1.7, 1.8	Administrador
RF04	El sistema Pretor permitirá consultar registros de los usuarios según su identificador, nombre o apellidos, mediante un formulario de un solo campo de texto.	1.9	Administrador
RF05	El sistema Pretor permitirá listar por páginas de longitud pre-configurada los registros de los usuarios, mostrando: identificador, nombre y apellidos, área académica, rol y estado.	1.10	Administrador
RF06	El sistema Pretor permitirá ordenar las listas de registros de usuarios, según uno de los siguientes	1.10	Administrador

	campos a la vez: identificador, nombre y apellidos, área académica, rol o estado.		
RF07	El sistema Pretor permitirá que un usuario modifique su perfil, que incluye su nombre, apellidos, correo electrónico, dirección, teléfono e idioma en que desea trabajar. Se enviará un correo automático al usuario notificando la modificación de su perfil, indicando la fecha de modificación	1.11	Administrador, Proveedor y Registrador
RF08	El sistema Pretor permitirá que un usuario modifique su contraseña, indicando previamente la contraseña actual.	1.12	Administrador, Proveedor y Registrador
Módulo de roles			
RF09	El sistema Pretor permitirá mantener registros con el nombre y descripción de los roles permitidos.	2.1, 2.2, 2.3	Administrador
RF10	El sistema Pretor permitirá asignar a los roles privilegios de administrar nodos, administrar grupos, administrar proyectos y asociar grupos a proyectos.	2.1, 2.2	Administrador
RF11	El sistema Pretor permitirá listar los roles registrados, con su nombre y descripción.	2.4	Administrador
RF12	El sistema Pretor permitirá ordenar las listas de registros de roles, según su nombre.	2.4	Administrador
RF13	El sistema Pretor permitirá verificar que un usuario tenga un rol con los privilegios necesarios para realizar una acción en el sistema.	2.5	Administrador, Proveedor y Registrador
Módulo de Cuentas BOINC			
RF14	El sistema Pretor permitirá mantener registros con el nombre, área académica, contraseña y correo electrónico de una cuenta BOINC.	3.1, 3.2, 3.3	Administrador
RF15	El sistema Pretor permitirá registrar y modificar los datos de una cuenta BOINC en cada uno de los proyectos científicos que tiene registrados.	3.1, 3.2	Administrador
RF16	El sistema Pretor permitirá consultar registros de las cuentas BOINC según su nombre, mediante	3.3	Administrador

	un formulario de un solo campo de texto.		
RF17	El sistema Pretor permitirá listar por páginas de longitud pre-configurada los registros de las cuentas BOINC, mostrando: nombre, cantidad de usuarios, cantidad de nodos, cantidad de créditos ejecutados por los nodos, cantidad de proyectos y cantidad de créditos consumidos por los proyectos.	3.4	Administrador
RF18	El sistema Pretor permitirá ordenar las listas de registros de cuentas BOINC, según uno de los siguientes campos a la vez: nombre, cantidad de usuarios, cantidad de nodos, cantidad de créditos ejecutados por los nodos, cantidad de proyectos y cantidad de créditos consumidos por los proyectos.	3.4	Administrador
Módulo de proyectos			
RF19	El sistema Pretor permitirá registrar y modificar proyectos mediante un formulario con los siguientes campos: nombre del proyecto, URL, unidad académica, descripción y firma.	4.1, 4.2, 4.3	Administrador
RF20	El sistema Pretor permitirá comprobar de manera automática que un proyecto existe y cumple con las condiciones de funcionamiento requeridas antes de registrarlo.	4.1, 4.2	Administrador
RF21	El sistema Pretor permitirá bloquear o activar un proyecto.	4.3, 4,4	
RF22	El sistema Pretor permitirá indicar que los nodos deben pedir nuevas tareas a un determinado proyecto de manera inmediata.	4.5	
RF21	El sistema Pretor permitirá consultar y registrar automáticamente y de manera periódica el estado de la comunicación con un proyecto (correcto o incorrecto) y la cantidad de créditos otorgados por el proyecto a cada área académica.	4.6, 4.7, 4.8, 4.9	Tiempo
RF22	El sistema Pretor permitirá consultar registros de	4.6, 4.8,	Administrador

	los proyectos según su nombre mediante un formulario de un solo campo de texto.	4.9	y Proveedor
RF23	El sistema Pretor permitirá presentar una lista general de todos los proyectos registrados en páginas de longitud pre-configurada, mostrando: nombre del proyecto, área académica a la que pertenece, créditos acumulados, cantidad de nodos que lo apoyan, estado de comunicación, estado de bloqueo y estado de actualización.	4.6	Administrador
RF24	El sistema Pretor permitirá ordenar la lista general de proyectos, según su nombre, área académica, créditos acumulados, cantidad de nodos, estado de comunicación, estado de bloqueo y estado de actualización.	4.6	Administrador
RF25	El sistema Pretor permitirá generar un reporte con el resumen de la información de los proyectos, que incluye: nombre, área académica, créditos acumulados, cantidad de nodos, estado de comunicación, estado de bloqueo y estado de actualización.	4.7	Administrador
RF26	El sistema Pretor permitirá presentar una lista de los proyectos propios de un área académica en páginas de longitud pre-configurada mostrando: nombre del proyecto, cantidad de créditos otorgados al área responsable del proyecto, cantidad de créditos otorgados a otras áreas, cantidad total de créditos otorgados a todas las áreas, cantidad de nodos otorgados por el área responsable del proyecto, cantidad de nodos otorgados por otras áreas, cantidad de nodos otorgados por todas las áreas y estado de comunicación.	4.8	Proveedor
RF27	El sistema Pretor permitirá ordenar la lista de proyectos propios de un área académica según su nombre, cantidad de créditos otorgados al área	4.8	Proveedor

	responsable del proyecto, cantidad de créditos otorgados a otras áreas, cantidad total de créditos otorgados a todas las áreas, cantidad de nodos otorgados por el área responsable del proyecto, cantidad de nodos otorgados por otras áreas, cantidad de nodos otorgados por todas las áreas y estado de comunicación.		
RF28	El sistema Pretor permitirá presentar una lista de los proyectos de otras áreas académicas a los que apoya un área determinada en páginas de longitud pre-configurada, mostrando: nombre del proyecto, área académica a la que pertenece, cantidad de créditos otorgados por el proyecto, cantidad de nodos contribuidos al proyecto y estado de comunicación.	4.9	Proveedor
RF29	El sistema Pretor permitirá ordenar la lista de proyectos no propios a los que apoya un área académica, según su nombre, área académica a la que pertenece, cantidad de créditos contribuidos al proyecto, cantidad de nodos contribuidos al proyecto y estado de comunicación.	4.9	Proveedor
RF30	El sistema Pretor permitirá mostrar la lista de todos los proyectos registrados en el sistema, indicando a cuáles de ellos está asociado un determinado grupo. La lista se mostrará en páginas de longitud pre-configurada e indicará: nombre del proyecto, área académica a la que pertenece, estado de asociación al grupo y estado de comunicación del proyecto.	4.10, 4.11	Administrador y Proveedor
RF31	El sistema Pretor permitirá ordenar la lista en que se indica a qué proyectos está asociado un grupo según el nombre del proyecto, el área académica a la que pertenece y el estado de comunicación del proyecto.	4.10, 4.11	Administrador y Proveedor

RF32	El sistema Pretor permitirá asociar y desasociar un grupo de computadoras a los proyectos listados.	4.12, 4.13	Administrador y Proveedor
Módulo de nodos			
RF33	El sistema Pretor permitirá comunicar su nombre y URL a un nodo que lo solicite. Además, le indicará que es un sistema administrador de cuentas y que requiere el ingreso de un usuario y contraseña para registrarlo en el sistema.	5.1	Nodo
RF34	El sistema Pretor permitirá verificar que el mensaje de solicitud de registro que envía un nodo contiene las credenciales correctas de un usuario existente en el sistema, que posea el privilegio de administrar nodos.	5.2	Nodo
RF35	El sistema Pretor permitirá obtener, registrar y actualizar de manera automática los siguientes datos de los nodos de la *grid*: CPID, CPID previo, nombre, dirección ip, versión de cliente BOINC, modo de ejecución, número de núcleos, proveedor, modelo, cantidad de FLOPs, cantidad de IOPs, cantidad de bytes de memoria RAM, cantidad de bytes de disco duro, cantidad de bytes de disco duro no utilizados, nombre de sistema operativo, versión de sistema operativo y versión de aplicación virtual box. Se debe registrar automáticamente la última fecha de actualización de los datos del nodo	5.2	Nodo
RF36	El sistema Pretor permitirá elegir un nodo para que actualice la información de sus coprocesadores, si los tiene. Si tiene coprocesadores ATI, se considera, para cada uno: nombre, memoria RAM disponible, memoria RAM local, frecuencia de memoria. Si tiene coprocesadores NVIDIA, se considera, para cada uno: FLOPs pico, identificador, nombre, versión	5.2	Registrador, Proveedor, Administrador y Nodo

	de *driver*, versión de coprocesador NVIDIA, memoria global total, frecuencia de reloj. Además, para cada coprocesador NVIDIA o ATI, si tuviera soporte para Open Cl, se considera: nombre, proveedor, tamaño de memoria global, máxima frecuencia de reloj, versión de plataforma, versión de dispositivo y versión de *driver*. Se debe registrar automáticamente la última fecha de actualización de los datos de los coprocesadores de un nodo.		
RF37	El sistema Pretor permitirá verificar si un nodo cuenta con la versión mínima requerida de cliente BOINC que configuró un administrador del sistema y enviarle un mensaje indicándole si cuenta con ésta o no.	5.2	Nodo
RF38	El sistema Pretor permitirá identificar si un nodo ha sido previamente registrado, mediante su identificador opaco, identificador ínter-proyectos (CPID) o CPID previo.	5.2	Nodo
RF39	El sistema Pretor permitirá eliminar el registro de un nodo o conjunto de nodos.	5.3	Registrador, Proveedor y Administrador
RF40	El sistema Pretor permitirá consultar registros de los nodos según su nombre, dirección ip, área académica y grupo mediante un formulario de un solo campo de texto.	5.4, 5.5, 5.6	Registrador, Proveedor y Administrador
RF41	El sistema Pretor permitirá listar por páginas de longitud pre-configurada, con fecha de última actualización y cantidad de resultados, los nodos registrados en el sistema, mostrando: nombre, grupo, área académica, modelo de procesador, cantidad de núcleos, cantidad de MFLOPS del procesador, cantidad de GB de disco libre, cantidad de GB de RAM disponibles, sistema operativo, estado de comunicación (activo o	5.7, 5.8, 5.9	Registrador, Proveedor y Administrador

	inactivo) y estado de trabajo (activo o suspendido).		
RF42	El sistema Pretor permitirá ordenar la lista de los nodos registrados según: nombre, grupo, área académica, modelo de procesador, cantidad de núcleos, cantidad de MFLOPS del procesador, cantidad de GB de disco libre, cantidad de GB de RAM disponibles, sistema operativo, estado de comunicación (activo o inactivo) y estado de trabajo (activo o suspendido).	5.7, 5.8, 5.9	Registrador, Proveedor y Administrador
RF43	El sistema Pretor permitirá consultar registros de los nodos según su nombre, cuenta BOINC, grupo, modelo de procesador, cantidad de núcleos, cantidad de MFLOPS del procesador, cantidad de GB de disco libre, cantidad de GB de RAM disponibles, sistema operativo, estado de comunicación y estado de trabajo, mediante un formulario que contemple todos los campos mencionados.	5.10, 5.11, 5.12	Registrador, Proveedor y Administrador
RF44	El sistema Pretor permitirá suspender y reactivar el envío de tareas a un nodo o conjunto de nodos.	5.13, 5.14	Proveedor y Administrador
RF45	El sistema Pretor permitirá generar un reporte de las características de los nodos: identificador, grupo, procesador, sistema operativo, cantidad de núcleos, cantidad de memoria RAM en GB, cantidad de espacio en disco duro disponible en GB, cantidad de GFLOPS, estado y cuenta BOINC.	5.15, 5.16, 5.17	Registrador, Proveedor y Administrador
Módulo de grupos			
RF46	El sistema Pretor permitirá agrupar los nodos en grupos de nodos.	6.1, 6.2	Proveedor y Administrador
RF47	El sistema Pretor permitirá consultar grupos de nodos según su nombre mediante un formulario de un solo campo.	6.3, 6.4	Proveedor y Administrador
RF48	El sistema Pretor permitirá mantener registros de	6.5, 6.6,	Proveedor y

	grupos de nodos, que incluyen su identificador y cuenta BOINC. Además, se incluye otros campos propios de la configuración de la *grid*, que indican las preferencias de uso del procesador, disco, memoria y red de cada nodo del grupo.	6.7, 6.8, 6.9, 6.10	Administrador
RF49	El sistema Pretor permitirá listar por páginas de longitud pre-configurada los grupos registrados en el sistema, mostrando: identificador, cuenta BOINC, cantidad de nodos activos, suspendidos y totales y cantidad de GFLOPS.	6.11, 6.12	Proveedor y Administrador
RF50	El sistema Pretor permitirá ordenar la lista de grupos registrados según: identificador, cuenta BOINC, cantidad de nodos activos, suspendidos y totales y cantidad de GFLOPS.	6.11, 6.12	Registrador, Proveedor y Administrador
Módulo de configuración			
RF51	El sistema Pretor debe permitir bloquear y reactivar el mecanismo de verificación de estado de los nodos registrados.	7.1	Administrador
RF52	El sistema Pretor debe permitir bloquear y reactivar el mecanismo de verificación de estado y recopilación de créditos de los proyectos.	7.2	Administrador
RF53	El sistema Pretor debe permitir modificar su idioma predeterminado, dirección URL, número de filas mostradas por cada tabla, imagen de la de la cabecera de los reportes, imagen de la cabecera de la interfaz web, longitud mínima de contraseña de usuario, máxima cantidad de días para que el usuario pueda modificar su contraseña después de haber solicitado recuperarla, máximo número de intentos errados en el ingreso, frecuencia de verificación del estado de comunicación con los nodos, versión mínima requerida de cliente BOINC, llave pública, nombre y contraseña a utilizar para registrarse en los proyectos, correo electrónico y firma a utilizar	7.3	Administrador

	en los correos electrónicos.		

Tabla 2.2: Requerimientos funcionales

2.2.4 Requerimientos no funcionales

En la tabla 2.3 se muestra los requerimientos no funcionales del sistema Pretor. Estos se encuentran alineados con las políticas informáticas actuales de la DIA.

Código	Requerimientos no funcionales
RN01	El sistema Pretor será desarrollado utilizando como base el mecanismo administrador de cuentas del sistema BOINC. En particular, se aprovechará los mensajes XML-RPC que BOINC genera.
RN02	El sistema Pretor será desarrollado en el lenguaje Java.
RN03	El sistema Pretor será desarrollado utilizando el JDK 1.7.
RN04	El sistema Pretor utilizará un servidor Apache Tomcat, versión 7.
RN05	El sistema Pretor utilizará un servidor central con sistema operativo Scientific Linux, versión 6.
RN06	El sistema Pretor se podrá visualizar en los siguientes navegadores: Internet Explorer 9, Mozilla Firefox 11, Google Chrome 21 y versiones compatibles.
RN07	El sistema Pretor presentará una interfaz basada en los estándares del sistema Legión.
RN08	El sistema Pretor presentará una interfaz bilingüe, en inglés americano y español.
RN09	El sistema Pretor almacenará las contraseñas de usuarios y áreas académicas cifradas con el algoritmo MD5.
RN10	El sistema Pretor utilizará los algoritmos RSA y MD5 para mantener segura la comunicación con los nodos y servidores de proyectos.
RN11	El sistema Pretor cifrará los nombres de los métodos de la capa controladora que sean accedidos por vía web.
RN12	El sistema Pretor registrará en una bitácora las excepciones que surjan durante su ejecución, incluyendo una breve descripción de éstas.
RN13	El sistema Pretor registrará en una bitácora las transacciones realizadas por un usuario, que involucren la consulta, modificación o inserción de datos a la base de datos. El registro debe incluir el identificador del usuario y una breve descripción de la transacción realizada.

RN14	El sistema Pretor incluirá un registro en una bitácora por cada una de las siguientes condiciones que cumpla un nodo: no tiene la versión mínima de cliente BOINC requerida, las preferencias de uso que utiliza no son las que le indicó el sistema Pretor, sus mensajes de comunicación no están bien formados. El registro debe incluir la dirección ip y nombre del nodo.
RN15	El sistema Pretor incluirá un registro en una bitácora cada vez que un nodo se comunique con el sistema. El registro debe incluir el identificador del nodo o su dirección ip y nombre, en caso de no poder obtener el primero.
RN16	El sistema Pretor incluirá un registro en una bitácora cada vez que se realice el contacto con un servidor de proyectos. El registro debe incluir los datos obtenidos del servidor.
RN17	El sistema Pretor incluirá un registro en una bitácora al inicio y fin de un ciclo de actualización del estado y créditos de los proyectos.
RN18	El sistema Pretor incluirá un registro en una bitácora al inicio y fin de un ciclo de actualización del estado de los nodos.
RN19	El sistema Pretor solicitará al usuario que realice una prueba de Turing pública y automática para diferenciar máquinas y humanos (CAPTCHA) si yerra en el ingreso un número de veces previamente configurado.
RN20	El sistema Pretor almacenará las direcciones de los nodos, según corresponda, de acuerdo a uno de los siguientes protocolos: ipv4 ó ipv6.
RN21	El sistema Pretor solo considerará el registro de proyectos con las siguientes características: se comunican utilizando el protocolo http, permiten la creación de cuentas, permiten la creación de equipos, no utilizan autenticadores opacos y no requieren de código de invitación.

Tabla 2.3: Requerimientos no funcionales

2.3 Análisis del sistema

El objetivo de esta sección es, primero, analizar la viabilidad del sistema en términos de los recursos con los que se debe contar: humanos, de *hardware* y de *software* y otros. Luego, se presenta una definición general del sistema, a partir de un diagrama entidad relación.

2.3.1 Recursos humanos

Para el desarrollo del sistema se requiere de la participación de tres actores clave: el administrador de la *grid* de la PUCP, el asesor de proyecto de fin de carrera y el encargado del proyecto. A continuación se presenta el análisis de costo y tiempo estimado para cada uno.

La tabla 2.4 muestra el costo por hora de cada recurso humano, y la 2.5 muestra la cantidad de recursos humanos de cada tipo requeridos, por cada etapa del modelo *stage-gate*. Además, indica la cantidad de horas necesarias y el monto económico asociado a éstas. Los montos se estiman considerando un sueldo promedio mensual de S/. 675 por 120 horas, para un practicante y otro de S/. 2000 por 160 horas, para el administrador de la *grid.* Para el caso del asesor, se considera el costo de la cantidad de créditos de los cursos de Tesis 1 y Tesis 2, de 2 y 4 créditos, respectivamente. Al momento de la elaboración de este documento, el precio de un crédito en la PUCP, en la escala económica tres, es de S/. 398.00 (PUCP 2012). Para el curso de Tesis 1, se considera un total de 28 horas por ciclo y, para el de Tesis 2, de 56 horas por ciclo. Cada ciclo dura 5 meses.

Costo de una hora de administrador (S/.)	13
Costo de una hora de asesor (S/.)	28
Costo de una hora de encargado del proyecto (S/.)	6

Tabla 2.4: Costo por hora de cada recurso humano

Recurso	Cantidad	Horas	Monto (S/.)
Descubrimiento			
Admin. de la *grid*	1	1	13
Asesor de proyecto	1	4	112
Encargado de proyecto	1	19	114
Total descubrimiento		**24**	**239**
Alcance			
Admin. de la *grid*	1	6	78
Asesor de proyecto	1	4	112
Encargado de proyecto	1	18	108
Total alcance		**28**	**298**
Construcción del caso de negocio			
Admin. de la *grid*	1	6	78
Asesor de proyecto	1	4	112
Encargado de proyecto	1	39	234
Total construcción del caso de negocio		**49**	**424**
Desarrollo			
Admin. de la *grid*	1	700	9 100
Asesor de proyecto	1	50	1 400
Encargado de proyecto	1	1365	8 190
Total desarrollo		**2 115**	**18 690**
Prueba y validación			
Admin. de la *grid*	1	10	130
Asesor de proyecto	1	3	84
Encargado de proyecto	1	20	260
Total prueba y validación		**31**	**474**
Total recursos humanos		**2 247**	**20 125**

Tabla 2.5: Costo de recursos humanos por cada etapa del proyecto

2.3.2 Recursos de *hardware* y *software*

Para el desarrollo del sistema, se necesitan dos conjuntos de recursos de *hardware* y *software*, donde cada uno corresponde a un entorno del producto. Los entornos comprendidos son: entorno de desarrollo y de pruebas.

El entorno de desarrollo del producto requiere de una computadora personal. Esta es provista por el encargado del proyecto. Es una computadora portátil, con procesador marca Intel Corei5, capacidad de 4 GB de memoria RAM y 500 GB de disco duro. Posee una tarjeta de red con conexión inalámbrica. Por otra parte, todo el *software* requerido para el desarrollo, instalado en la computadora señalada, es de uso gratuito y se lista a continuación:

- Administrador de máquinas virtuales Oracle VM VirtualBox.
- Aplicación MySQL Workbench.
- Entorno de desarrollo NetBeans.
- Plataforma Java EE.
- Sistema operativo Scientific Linux.
- *Middleware* BOINC.
- Navegador web Mozilla Firefox.
- Navegador web Internet Explorer.
- Navegador web Google Chrome.
- Servidor de base de datos MySQL.
- Servidor web Apache Tomcat.
- *Suite* de aplicaciones de oficina LibreOffice.
- Cliente para control de versiones Tortoise SVN Subversion.

El entorno de pruebas requiere de un conjunto de cuarenta nodos de la *grid* de la PUCP, de un servidor de aplicación web y base de datos y de tres servidores de proyecto científico activos. Tanto los nodos como los servidores y el cableado estructurado que los enlaza son provistos por el cliente. Cada nodo tiene un procesador Intel Corei5 y posee conexión a red. El único *software* exigido en cada nodo es el *middleware* BOINC. Para el caso del servidor de aplicación web y base de datos, se requiere del *software* Apache Tomcat y MySQL.

Como se puede notar, todo el *software* requerido es de uso gratuito, por lo que no supone un costo adicional para el proyecto. Con respecto al *hardware,* se cuenta con todo lo requerido. El costo asociado a éste último se deriva del desgaste que sufre el equipo debido al uso cotidiano. Para asignarle un valor, se utiliza el porcentaje de depreciación para equipos de procesamiento de datos que indica el estado peruano (SUNAT 1994). Entonces, dado que el equipo es utilizado aproximadamente durante doce meses para el proyecto y que su precio de compra es de S/. 2600, el costo asociado es de S/. 650. Además, es el único costo de recursos de *hardware* y *software* significativo en que se incurre para el desarrollo del producto.

2.3.3 Otros recursos

Adicionalmente a los recursos antes indicados, se debe considerar los gastos en servicios de corriente eléctrica, agua y desagüe, teléfono e Internet. Para todos ellos se estima un monto de S/. 200 mensuales, basado en la experiencia del encargado.

La tabla inferior muestra el monto total asociado al proyecto. Esta incluye todos los recursos mencionados anteriormente. Para el caso, de los recursos de tipo otros, que son estimados en montos mensuales, se considera una extensión de doce meses para el proyecto.

Recurso	Monto (S/.)
Recursos humanos	20 125
Recursos de *hardware* y *software*	650
Otros	
Servicios de corriente eléctrica, agua y desagüe, teléfono e Internet	2 400
Total	**23 175**

Tabla 2.6: Monto total del proyecto

2.3.4 Justificación de la viabilidad del sistema

La estructura de recursos antes detallada es presentada tanto al asesor como al representante del cliente, junto con el plan de proyecto, el catálogo de requisitos y los prototipos elaborados. Ambos dan su aprobación para poder continuar con la ejecución del proyecto, teniendo en cuenta el tiempo y costo estimados. También, afirman su compromiso para cumplir con las funciones asignadas y brindar los recursos necesarios.

2.3.5 Descripción del sistema

Para concluir este capítulo, se presenta la metáfora del sistema. Para ello, se recurre a un diagrama entidad relación. Se ha omitido los atributos de cada entidad por motivos de claridad y porque estos se encuentran detallados en la lista de requerimientos.

Como se puede ver en la ilustración 2.3, el sistema asocia a cada unidad de la PUCP una cuenta BOINC. Todos los usuarios, grupos de nodos y proyectos de la unidad son englobados bajo una misma cuenta. Esto con el objetivo de permitir que los usuarios de una misma unidad puedan acceder a la información de sus grupos de nodos y proyectos. Además, el administrador de la *grid* puede generar reportes que permiten obtener y comparar las características de los nodos y proyectos de cada unidad.

Los usuarios del sistema son asignados a los miembros de una unidad que participan de las tareas de administración de la *grid.* Cada usuario goza de un conjunto de privilegios que le permiten realizar determinadas funciones relacionadas con los nodos, grupos de nodos y proyectos.

Los registros de los nodos de la *grid* son agregados automáticamente al sistema mediante el uso del *middleware* BOINC. Este último también permite establecer un mecanismo de comunicación continua para informar a cada nodo las tareas que debe realizar, una vez que se le asignó uno o más proyectos al grupo al que pertenece.

Los nodos de la *grid* son clasificados en grupos, de acuerdo a criterios de ubicación geográfica o características del nodo. Cada grupo tiene un conjunto de preferencias de uso configurables, que es transmitida a los nodos a través del *middleware*

BOINC. Los grupos ofrecen una forma rápida de asignar un conjunto de nodos a un proyecto y administrarlos, en vez de hacerlo uno por uno.

Los proyectos de cada unidad son visibles tanto a nivel interno como también a nivel de otras unidades. Esto permite que los grupos de nodos de una unidad apoyen a los proyectos de otra. Cada proyecto se aloja en un servidor BOINC, con una dirección URL, que debe ser registrada manualmente para poder establecer la comunicación con éste.

El sistema así descrito cuenta con todas las entidades y relaciones necesarias para cubrir la lista de requerimientos identificados. Entonces, cada requerimiento puede ser expresado en términos de la interacción entre entidades relacionadas directa o indirectamente. No se necesita añadir ninguna otra entidad para poder cubrir todos los requerimientos.

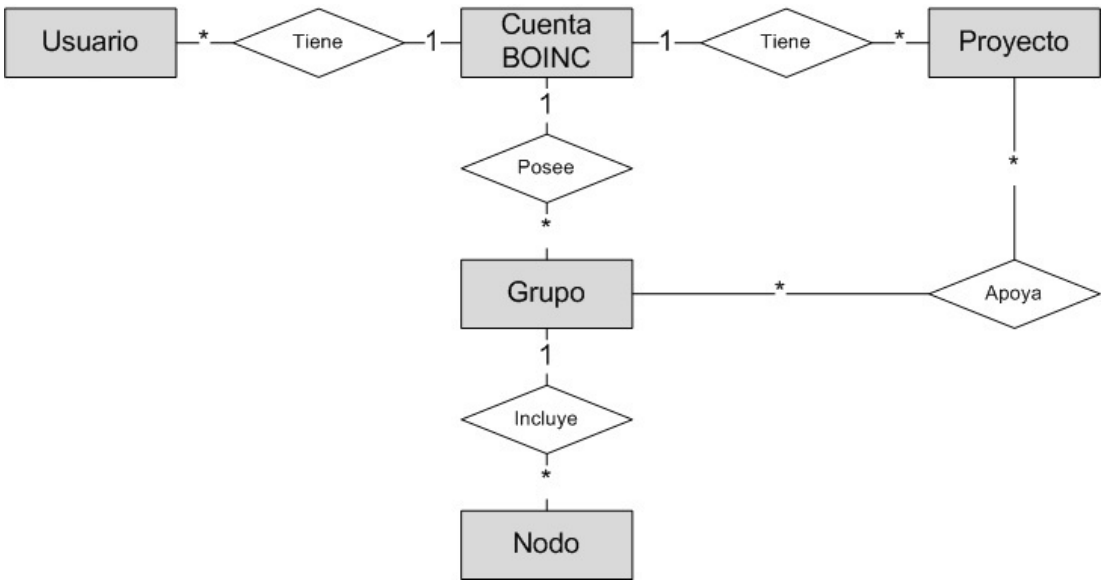

Ilustración 2.3: Diagrama entidad relación

3 Diseño

Este capítulo tiene como objetivo, primero, presentar la arquitectura del sistema que se desarrolla como proyecto de fin de carrera. Para ello, se describen los puntos considerados más importantes, porque facilitaron el entendimiento entre desarrollador y cliente. Luego, se detallan las consideraciones tomadas en cuenta para el diseño de interfaz gráfica.

3.1 Arquitectura de la solución

Tanto Abrahamsson como Fowler indican que XP carece de un enfoque claro para el tema de la arquitectura del sistema (Abrahamsson 2006; Fowler 2004). Es por ello que el primer autor propone el uso de la técnica de historias de desarrollador, mientras que el segundo recomienda plantear una arquitectura inicial general, que se irá modificando a lo largo del desarrollo mediante la técnica de refactorización.

Entonces, dado que XP carece de artefactos que den soporte a la arquitectura, se decide optar por la aproximación propuesta por Fowler. Esto implica el uso de borradores y diagramas simples, que faciliten la comprensión de la arquitectura tanto al cliente como al desarrollador. Por otra parte, estos diagramas no buscan

constituirse como un modelo permanente al que se debe ceñir el sistema (Fowler 2004); esto teniendo en mente las ideas de cambio continuo asociadas a la programación extrema (Jeffries 2012).

En la ilustración 3.1 se presenta el diagrama más representativo del sistema, según el cliente y el desarrollador. Como se puede ver, el sistema permite a los usuarios conectarse a través de un navegador *web*, que se comunica tanto con la vista como con el controlador. Este último también se encuentra a la escucha de las comunicaciones de los clientes BOINC, que se dan mediante llamadas a procedimientos remotos que hacen uso de mensajes XML para transmitir los datos, como se puede ver en el anexo 18. Por otro lado, en el caso de los servidores de proyecto BOINC, la aplicación inicia la comunicación para obtener sus datos y de ello se encarga la capa de servicios.

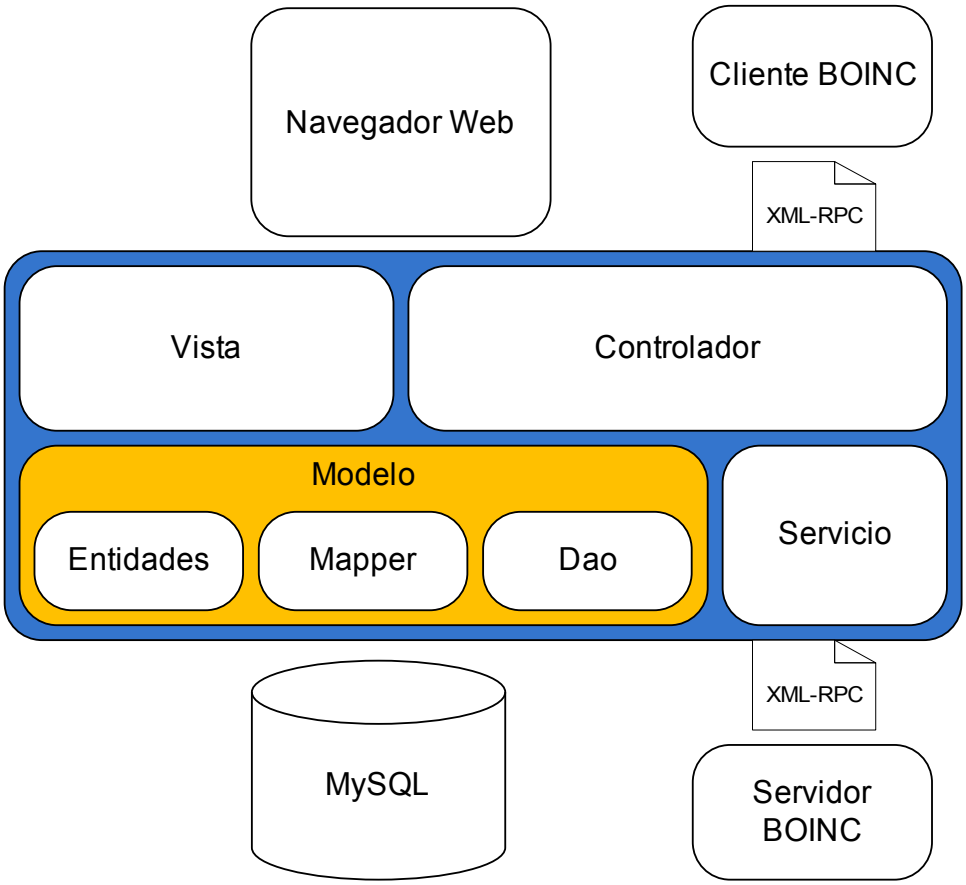

Ilustración 3.1: Arquitectura del sistema Pretor

3.1.1 Descripción de los componentes de la solución

En este apartado se describe brevemente las funciones de cada uno de los componentes del sistema, justificando el porqué de su existencia.

a) Vista

Este componente se encarga de ofrecer al usuario una interfaz gráfica que le permita utilizar las funcionalidades del sistema. Interacciona directamente con el navegador *web*, con las entidades del modelo y con el componente controlador. Se justifica su existencia porque permite reunir en un solo lugar los aspectos relacionados con la interfaz que se presenta al usuario, facilitando su modificación y reemplazo.

b) Controlador

Este componente se encarga de validar los datos que se obtienen tanto de la vista como de los mensajes xml de un cliente BOINC y entregarlos al componente de servicios, para que sean procesados y almacenados. A la inversa, también se encarga de recuperar los datos almacenados en el modelo, recurriendo al componente de servicios, para poder retransmitirlos a la vista o a los clientes BOINC. Además, dentro de este componente se controla la seguridad del sistema y la navegación.

c) Servicio

Éste es un componente intermedio entre el controlador y el modelo. Se justifica su existencia porque permite reunir en un solo lugar las funciones encargadas de obtener y modificar los datos de las entidades del sistema según la lógica de negocio. Esto hace transparente al componente controlador el origen de los datos y la forma en que son obtenidos: mensajes RPC con datos de las bases de los servidores de proyectos o consultas a la propia base de datos del sistema.

d) Modelo

Este componente se encarga de centralizar el proceso de obtención y modificación de datos de la base de la aplicación. Además, en este componente se definen las entidades que utiliza el sistema, por lo que interacciona directamente con el resto de componentes. Se justifica su existencia porque permite reunir en un solo lugar tanto las entidades que usa el sistema, como las clases que contienen el detalle

técnico necesario para acceder a la bases de datos, facilitando así su modificación y reemplazo.

3.1.2 Diseño de base de datos

Como tarea inicial de la fase de desarrollo indicada por la metodología Stage-Gate se elabora el documento de estándares de base de datos, cuyo objetivo es ayudar a un fácil mantenimiento y comprensión de la nomenclatura utilizada. Este se puede ver en el anexo 19.

A continuación, con los estándares ya definidos, se elabora el diseño inicial de base de datos. Es un diseño inicial, ya que puede cambiar según lo que se acuerde con el cliente, en concordancia con la metodología XP. En el anexo 20 se puede observar los diseños inicial y final de la base.

Finalmente, para asegurar que el diseño de base de datos satisface los requerimientos del cliente, se realiza un mapeo de estos a las transacciones necesarias para satisfacerlos, como se puede ver en el anexo 21.

3.1.3 Patrones utilizados

En este apartado se describen los patrones de diseño aplicados al desarrollo del sistema Pretor.

a) Modelo Vista Controlador – Modelo 2

La triada Modelo Vista Controlador (MVC) es en realidad la aplicación de tres patrones de diseño: *Observer*, *Composite* y *Strategy* (Gamma et al. 1998). Considerando lo anterior, se decide explicar brevemente la triada mencionada, como un todo, sin entrar al detalle de cada uno de los patrones que comprende, para fines prácticos de aplicación al sistema Pretor.

Según Chopra, inicialmente el diseño de aplicaciones web se realizó de acuerdo al modelo 1 de la triada MVC. Por las diversas desventajas que presentaba este modelo, se optó por elaborar un segundo (Chopra et al. 2005). No es objetivo de este documento el detallar las características del modelo 2, por lo que se adjunta un gráfico, una breve descripción y sus principales ventajas.

Como se puede ver en el gráfico 3.2, el usuario hace un *request* al controlador (o *servlet*) utilizando su navegador. El controlador recibe el *request* y realiza la tarea que se indica. Para esto, necesita de la lógica de negocio, que se encuentra en el modelo. Terminada la tarea, el controlador determina la vista adecuada para presentar los resultados y la envía de regreso al usuario.

Así planteado, el modelo 2 ofrece cuatro ventajas principales identificadas por Chopra (Chopra 2005 et al.). Evita la repetición de código, facilita el mantenimiento de la aplicación, la hace escalable y fácil de probar. Por todo ello, se decide utilizar este patrón.

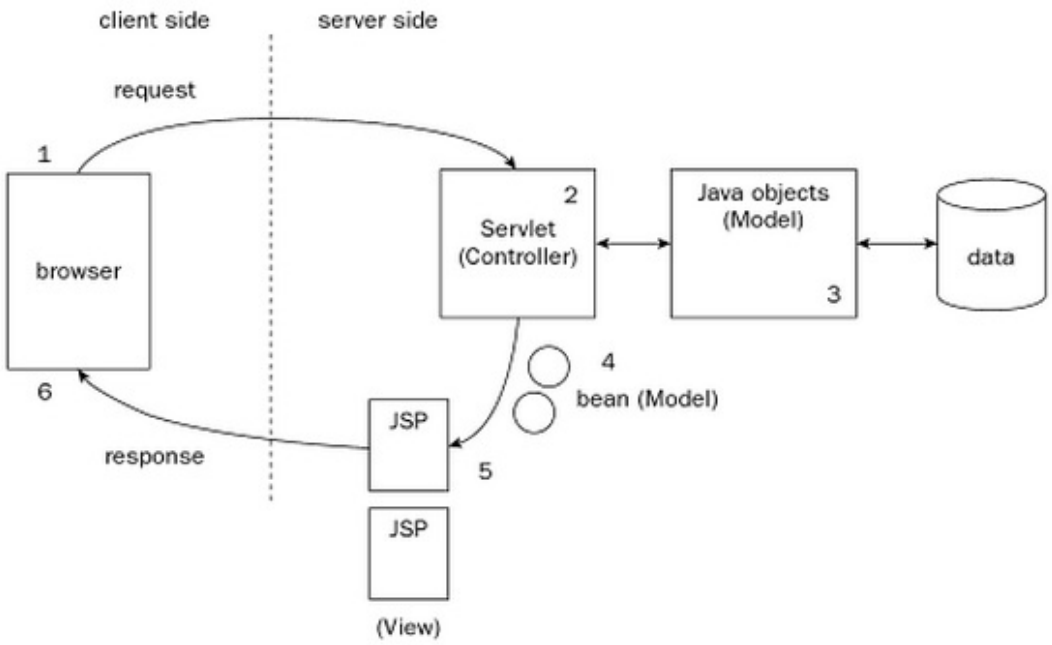

Ilustración 3.2: MVC – Modelo 2 (Chopra et al. 2005)

b) Solitario (Singleton)

Este patrón se asegura de que una clase tenga una sola instancia y provee un punto de acceso global a la misma. Su principal beneficio es brindar acceso controlado a una sola instancia de una clase (Gamma et al. 1998). Aplicado a Pretor el patrón solitario permite el acceso controlado a la única instancia que contiene las propiedades de configuración del sistema. De esta manera, se puede

mantener esta información en memoria, evitando así las consultas repetidas a la base de datos y las demoras que éstas pueden implicar.

3.1.4 Consideraciones de seguridad

A continuación se listan las principales consideraciones de seguridad que se toma en cuenta para el desarrollo del sistema Pretor. Éstas se basan tanto en requerimientos del cliente, como en los que se derivan del uso del *middleware* BOINC. Para su implementación se toma como ejemplo el código del sistema Legión.

a) Contraseñas cifradas

Las contraseñas son almacenadas en la base de datos después de ser cifradas utilizando el algoritmo MD5.

b) Log in

El ingreso al sistema requiere que el usuario se autentifique mediante su identificador y contraseña.

c) Mantener el usuario con sus privilegios durante toda la sesión

Durante todo el tiempo que dure la sesión de un usuario autenticado en el sistema, es posible verificar sus privilegios. Pretor los verifica antes de ejecutar cualquier acción en el controlador.

d) Cifrar los enlaces

El sistema Pretor cifra el texto de los enlaces que permiten la navegación hacia sus páginas.

e) Anotaciones para métodos del controlador

El sistema Pretor utiliza anotaciones para indicar el conjunto de métodos que pueden ser invocados en el controlador.

f) Bitácora

El sistema Pretor registra en la bitácora del sistema operativo, en este caso Scientific Linux 6, las acciones ejecutadas que involucren la consulta o modificación de datos por parte de un usuario, un nodo o un proceso automático del sistema.

g) Certificados para las comunicaciones con los nodos y proyectos

Un peligro latente que se deriva del uso del *middleware* BOINC es que es posible que una persona malintencionada cree un programa que se haga pasar por un administrador de cuentas, como Pretor. Éste podría asociar los nodos a un proyecto con fines deshonestos, como por ejemplo el desciframiento de contraseñas por métodos de fuerza bruta. Para evitarlo, BOINC hace uso del sistema criptográfico RSA, que permite generar una llave pública con la que se firman los nombres de los proyectos incluidos en el administrador de cuentas.

h) Transacciones parametrizadas en la base de datos

El sistema Pretor hace uso del *framework* MyBatis, el cual se encarga de realizar transacciones parametrizadas en la base de datos. Gracias a ello, se evita ataques del tipo *SQL Injection.*

i) Doble validación

El sistema Pretor valida los datos ingresados, tanto al nivel de la vista, mediante *scripts*, como del controlador, mediante métodos propios. De esta manera se contempla el hecho de que el usuario pueda desactivar el funcionamiento de los *scripts* en su navegador y explotar alguna vulnerabilidad del sistema.

j) *Cross-site scripting* (XSS)

El sistema Pretor se asegura de que los textos introducidos por la interfaz gráfica sean almacenados como tales y que no sean interpretados como comandos ejecutables por el navegador *web*, como cuando se utiliza etiquetas *javascript*. Para esto, se recurre a la librería Apache Commons Lang de la fundación Apache.

3.2 Diseño de Interfaz Gráfica

Como se mencionó en la sección de definición del problema, del capítulo 1, el sistema Pretor se concibe como una herramienta de administración de las computadoras que comprende el sistema Legión desarrollado por la DIA. Por tanto, dada la vinculación que existe entre ambos sistemas, el cliente requiere que se adecuen a los mismos estándares de diseño de interfaz gráfica.

Establecido lo anterior, en esta sección se presenta un resumen con los aspectos que se considera más importantes del diseño de interfaz gráfica del sistema Legión, que aplican al sistema Pretor. En particular, las secciones de las pantallas y los colores y formatos de texto.

3.2.1 División de la pantalla en secciones

En este apartado se indican las secciones generales en las que se dividen las pantallas del sistema Pretor. Cabe resaltar que esta división se ha tomado del diseño del sistema Legión, con el objetivo de mantener los estándares de la DIA. Las secciones se pueden ver en la ilustración 3.3.

a) Encabezado

En esta sección se indica de manera visible, al lado izquierdo, el logotipo del sistema Pretor. Así mismo, al lado derecho, aparece el logotipo de la organización para la que se hace el sistema.

b) Menú

En esta sección se lista las principales opciones del sistema a las que tiene acceso el usuario, de acuerdo a su rol, con el objetivo de facilitar la navegación.

c) Encabezado de contenido

Esta sección muestra, en la parte superior izquierda, un conjunto de enlaces que permiten al usuario identificar y desplazarse a través de la ruta seguida hasta el punto del sistema donde se encuentra actualmente. En la parte superior derecha, se muestra el nombre asociado al usuario actual. Por último, en la parte inferior izquierda, se muestra el nombre de la funcionalidad que se encuentra en uso.

d) Sección de contenido

Esta sección contiene los elementos con los que el usuario interacciona para utilizar la funcionalidad que elija.

e) Pie de página

Esta sección contiene la información de contacto del área de la PUCP encargada del desarrollo del sistema.

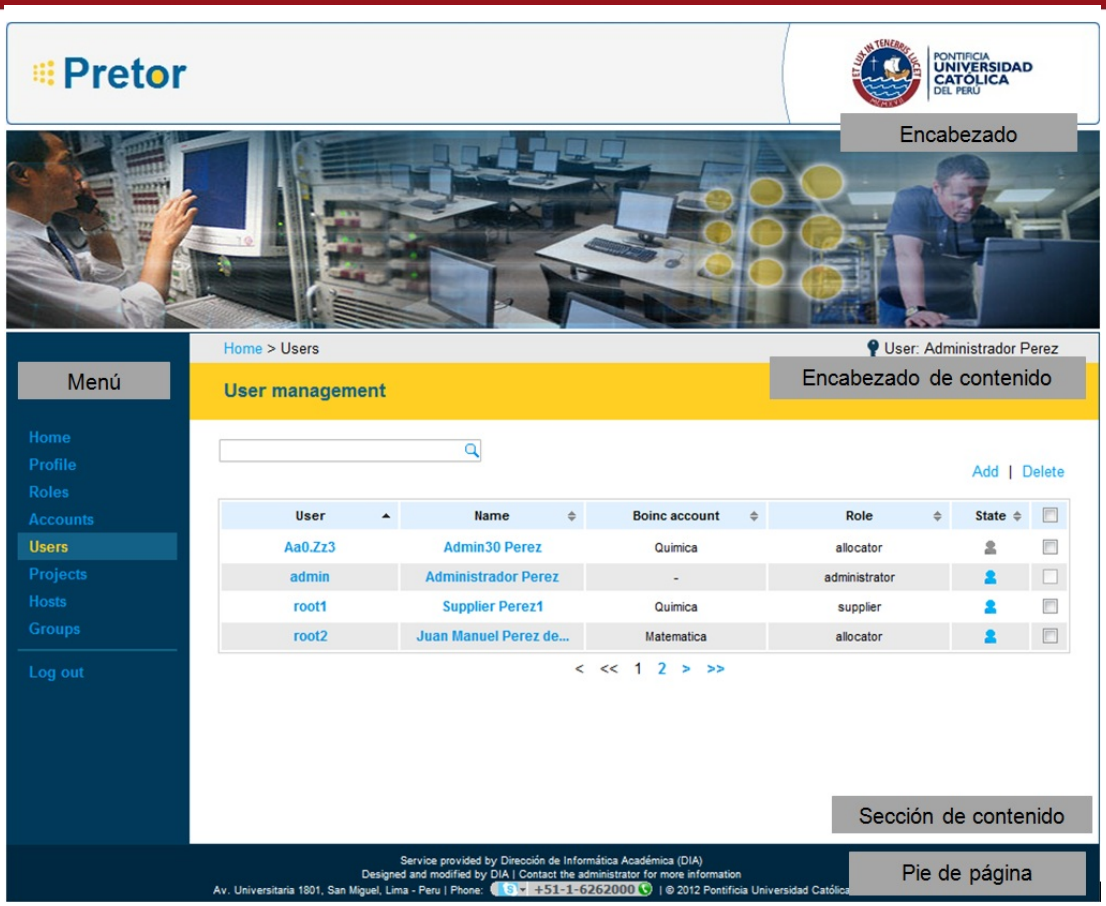

Ilustración 3.3: Secciones de pantalla

3.2.2 Colores y formatos de texto

Para estandarizar los colores y formatos de texto utilizados en los sistemas Legión y Pretor, se hace uso de las mismas hojas de estilo (CSS). De éstas, algunas son elaboración propia de la DIA, mientras que otras son parte de los componentes de las librerías *jQuery* y *jQuery-ui* que se utilizan. Un ejemplo de los colores y formatos de texto utilizados se puede apreciar en la ilustración 3.3.

4 Construcción y Pruebas

En este capítulo se detalla las tecnologías utilizadas para la construcción del sistema Pretor, justificando el porqué de su elección. Acto seguido, se describe los tipos y estrategia de prueba utilizados. Por último, se evalúa los resultados obtenidos en las pruebas.

4.1 Construcción

En esta sección se lista el conjunto de elecciones realizadas en el tema de tecnologías a utilizar para el desarrollo de la solución. Además, se justifica, para cada caso, el porqué de la elección.

4.1.1 Sistemas operativos

El sistema operativo que se utiliza en los servidores de aplicación y base de datos es Scientific Linux (SL), versión 6, debido a que es un requerimiento no funcional del cliente.

Profundizando un poco más en el punto anterior, se justifica el requerimiento del cliente, primero, en que SL es *software* libre, tal como lo exigen las políticas de la DIA. Segundo, SL es una distribución de Linux, que es considerada como más segura y estable que Windows (Harbaugh 2009); siendo estas características apreciadas por el cliente. Tercero, si bien se considera la opción de utilizar Red Hat Enterprise Linux (RHEL), debido a que ofrece soporte 24/7, se descarta debido al presupuesto que requiere. Entonces, se opta por las distribuciones CentOS y SL, consideradas por el cliente como similares a RHEL. Finalmente, se toma SL dado que, revisando el historial de versiones de ambas distribuciones, se tiene que ésta libera mayor cantidad de versiones en menor tiempo que CentOS. Esto se considera un indicador de que SL es un sistema operativo más revisado y probado que el otro.

Por otro lado, para el caso de los nodos, se utiliza una máquina virtual con sistema operativo Linux, distribución Ubuntu, configurada para utilizar solo un porcentaje determinado de los recursos totales del nodo. Esto es un requerimiento del cliente, ya que el sistema Legión lo necesita para poder establecer la *grid* de la PUCP.

4.1.2 Lenguaje de programación

El lenguaje de programación elegido para el desarrollo de la solución es Java. El estándar seleccionado es *Enterprise Edition* y el conjunto de herramientas base utilizadas es *Java Development Kit* versión 7 (JDK 7).

La razón principal por la que se elige Java es porque es un requerimiento del cliente, quien busca estandarizar los lenguajes en que se desarrollan los sistemas Legión y Pretor.

El estándar seleccionado es *Enterprise Edition 6*, primero, porque, como parte de su especificación, incluye un conjunto de tecnologías útiles para el desarrollo de aplicaciones *web*. Además, incluye también, tecnologías utilizadas en los componentes del sistema Legión que se pueden reutilizar en Pretor. A saber: JavaServer Pages 2.2 y Standard Tag Library for JavaServer Pages 1.2, para el componente de la vista; y Java Servlet 3.0, para el componente controlador (Oracle 2012).

Como herramientas de soporte para el desarrollo en Java, se utiliza el JDK 7, porque éstas permiten realizar tareas comunes relacionadas al desarrollo, como compilación, ejecución, *debugging*, sugerencias e inserción de código, generación de documentación, entre otras (Oracle 2011a). Por otro lado, el uso de estas herramientas se hace a través del entorno de desarrollo elegido, facilitando así la labor del programador.

4.1.3 Servidor de aplicaciones *web*

El servidor elegido es Apache Tomcat 7. Se define como una implementación de *software* libre de las tecnologías Java Servlet y Java Server Pages (Apache Software Foundation 2012). El motivo de su elección es, primero, que es *software* libre, lo cual está de acuerdo con las políticas de la DIA. Segundo, que implementa solamente las dos tecnologías necesarias para el funcionamiento de Pretor, indicadas en el inciso 4.1.2. Entonces, dado que implementa solo esas tecnologías, se evita el desperdicio de recursos en otras adicionales que no se utilizan, como es el caso si se usara un servidor GlassFish o JBoss, que implementan tecnologías como JavaServer Faces o JavaBeans, respectivamente. Finalmente, debido a que se cuenta con una gran cantidad de documentación e información sobre casos de uso disponible sobre éste. En la tabla 4.1 se presenta un resumen de lo indicado.

	Apache Tomcat 7	**GlassFish 3**	**JBoss AS 4**
***Software* libre**	Sí	Sí	Sí
Implementa tecnologías Java Servlet y Java Server Pages	Sí	Sí	Sí
Implementa otras tecnologías no requeridas	No	Sí	Sí
Documentación e información sobre casos de de uso disponible	Sí	Sí	Sí

Tabla 4.1: Comparación entre servidores de aplicaciones web

4.1.4 Servidor de base de datos

El servidor de base de datos elegido es MySQL, versión 5. Las razones de su elección son varias. Primero, porque es *software* libre, lo cual está de acuerdo con las políticas de la DIA. Segundo, dispone de un conjunto de funciones ya implementadas, las cuales facilitan la tarea de programación del presente proyecto de fin de carrera. Por ejemplo, funciones de búsqueda y ordenamiento de direcciones IPv4 o de conversión de fechas a segundos. Tercero, incluye características que permiten optimizar las consultas que implican el uso simultáneo de muchas tablas, como es el caso de *inner* y *outer joins*. Esto es importante, ya que este tipo de consultas es frecuente en el presente proyecto. Finalmente, al igual que otras bases de datos, dispone de herramientas para facilitar su administración. En este caso particular, se usa la aplicación MySQL WorkBench.

La tabla 4.2 muestra un cuadro comparativo ente MySQL y otras dos bases de datos alternativas.

	MySQL 5	**SQLite 3**	**FireBird 2**
***Software* libre**	Sí	Sí	Sí
Incluye funciones ya implementadas y útiles al proyecto	Sí	Parcialmente	Parcialmente
Incluye características para optimizar consultas de muchas tablas	Sí	Parcialmente	Sí
Dispone de herramientas administrativas	Sí	Sí	Sí

Tabla 4.2: Comparación entre servidores de base de datos

Por otra parte, con el objetivo de facilitar el trabajo con la base de datos MySQL, se utiliza el *framework* MyBatis. Este incluye la herramienta MyBatis Generator, que permite generar código Java para realizar operaciones de tipo CRUD sobre cada una de las tablas de la base de datos, ocultando los detalles técnicos (MyBatis 2011). Cabe resaltar que si bien esto supone una gran ayuda para el programador, en algunos casos es necesario extender las clases generadas por MyBatis con

otras nuevas para poder realizar transacciones elaboradas. Esto último permite aprovechar funciones propias de la base de datos, que dan como resultado un mejor desempeño que el que se obtiene al utilizar solo código de la aplicación web. Además, ayuda a dividir la carga de procesamiento entre el servidor de la aplicación *web* y el de base de datos. Finalmente, hace que el código sea fácil de mantener, puesto que las clases generadas por el *framework* no son alteradas manualmente, por lo que pueden ser recreadas de manera automática en caso se desee modificar la estructura de la base de datos.

4.1.5 Entorno de desarrollo

El entorno de desarrollo elegido es NetBeans. Los criterios tomados en cuenta para su elección son, primero, la experiencia del programador en el uso del mismo. Segundo, que hace uso de manera automática de las herramientas proporcionadas por el JDK 7. Tercero, que permite inicializar y detener de manera automática el servidor de aplicación Apache Tomcat. Cuarto, que cuenta con herramientas que facilitan la ejecución individual y automática de cada una de las pruebas codificadas. Quinto, que cuenta con ventanas individuales que permiten hacer el seguimiento del contenido de cada una de las variables utilizadas y de los mensajes entre el cliente y el servidor web. Sexto, que contiene funcionalidades que facilitan el uso del gestor de versiones. Por último, que permite llevar un historial de los cambios realizados de manera local a un archivo y compararlo con sus estados anteriores.

4.1.6 Sistema de control de versiones

El sistema de control de versiones utilizado es Apache Subversion, versión 1.7. Este es el gestor del que dispone la DIA para todos sus proyectos.

4.1.7 *Middleware* BOINC

Se utiliza este *middleware,* por ser el que actualmente permite establecer la *grid* de la DIA. Además, incluye un conjunto de mensajes XML, que facilitan la obtención de datos de los servidores de proyectos científicos de la PUCP y la administración de los nodos de la *grid.* Los mensajes se pueden ver en el anexo 22.

4.1.8 Librerías adicionales

El sistema Pretor hace uso de un conjunto de librerías adicionales que permiten reutilizar código elaborado por terceros para extender sus funcionalidades. En la tabla 4.1 se listan las principales.

Librería	Descripción
JCaptcha	Librería que ofrece funcionalidades que buscan reconocer que efectivamente es un ser humano el que está ejecutando una operación en el sistema.
JFreeChart	Librería que permite generar gráficos con información obtenida a partir de los datos del sistema.
Log4J	Librería que ofrece funcionalidades para la gestión de la bitácora del sistema.
JavaMail	Librería que permite gestionar el envío de correos electrónicos desde el sistema.
Jquery y jQuery UI	Librerías que pemiten incluir elementos interactivos como parte de la vista del sistema, como pestañas o combos desplazables. Además, también se utilizan para realizar validaciones de los datos ingresados en los formularios del sistema, a nivel del cliente *web*.

Tabla 4.3: Librerías adicionales que utiliza Pretor

4.1.9 Componentes reutilizados del sistema Legión

El sistema Pretor, dada su afinidad con el sistema Legión, reutiliza y adapta componentes elaborados para este último. Esto permite agilizar el desarrollo. En la tabla 4.4 se lista los principales componentes reutilizados.

Cabe resaltar también, que el sistema Legión, en general, se toma como modelo para el desarrollo del sistema Pretor, aprovechando su característica de *software* de código abierto. Adicionalmente, se cuenta con el consentimiento expreso y constante apoyo de sus creadores.

Componente	Descripción
Legion table	Componente que permite presentar fácilmente una tabla con un conjunto de datos de diferentes tipos en la vista del sistema. Durante la adaptación al sistema Pretor se agregan las características de: ordenamiento por columnas y encabezados de dos niveles.
Base Controller	Componente que se encarga del manejo de la seguridad del sistema, verificando los mensajes recibidos desde el cliente *web.*

Tabla 4.4: Componentes reutilizados de Legión

4.1.10 Librerías XML

Dado que BOINC se comunica con los clientes y servidores de proyectos mediante mensajes XML-RPC, es indispensable contar con una librería adecuada para su procesamiento. Para elegirla, primero, se busca en la *web* aquellas que sean libres y recomendadas por varios programadores. Así se obtiene: StAX, SAX, DOM y XOM. Luego, se realiza un análisis comparativo de éstas, el cual se presenta en la tabla 4.3. Adicionalmente, se ejecuta una prueba sencilla para comprobar empíricamente la velocidad que ofrecen las librerías, la que se muestra en el anexo 23.

A partir de los resultados resumidos en la tabla 4.5, se decide utilizar la librería SAX para la lectura de los mensajes y StAX para su escritura. Primero, se descarta XOM, dado que no posee soporte para atributos de tipo CData, los cuales son utilizados por BOINC. Además, XOM, a diferencia de las otras tres librerías, no viene incluida en el JDK 7 y es actualizada con menor frecuencia. Segundo, entre las tres restantes, se evalúa la velocidad de lectura y escritura mediante un programa simple elaborado como ápice arquitectónico, según XP. Se considera crítico este factor, dado que el cliente requiere que el sistema Pretor permita una interacción rápida con los nodos y servidores de proyectos.

Propiedad a evaluar	StAX	SAX	DOM	XOM
Facilidad de uso	Alta	Media	Alta	Alta
CPU y eficiencia de memoria	Alta	Media	Varía según el tamaño del documento procesado.	Alta en memoria
Lee XML	Sí	Sí	Sí	Sí
Escribe XML	Sí	No	Sí	Sí
Soporte para CData	Sí	Sí	Sí	No
Viene incluido en el JDK 7	Sí	Sí	Sí	No
Última actualización	Se actualiza constanteme nte	Se actualiza constanteme nte	Se actualiza constantemente	06-Feb-2011
Velocidad de lectura	Media	Alta	Baja	No se evaluó.
Velocidad de escritura	Alta	No escribe.	Baja	No se evaluó.

Tabla 4.5: Comparación de librerías XML

4.2 Pruebas

En esta sección se presenta los tipos y estrategia de prueba utilizados, justificando el porqué de su elección. Además, se muestra y discute los resultados obtenidos tras la ejecución de las pruebas.

4.2.1 Tipos de prueba utilizados

Para el sistema Pretor se consideran dos tipos de prueba: de integración y de carga.

a) Pruebas de integración

El objetivo de estas pruebas es evaluar y detectar errores durante la ejecución de las funcionalidades requeridas por el cliente, entendidas como la interacción entre los componentes del sistema.

b) Pruebas de carga

El objetivo de estas pruebas es evaluar el comportamiento del sistema cuando está sometido a un determinado nivel de uso, en un ambiente controlado.

4.2.2 Estrategia de pruebas

A continuación se presenta la estrategia utilizada para cada uno de los tipos de prueba indicados.

a) Pruebas de integración

Dado que los requerimientos del sistema son obtenidos utilizando el método de prototipos, no se cuenta con casos de uso o algún otro tipo de clasificador al cual asociar las pruebas de integración. Es por ello que se decide elaborar historias de usuario que describan la funcionalidad requerida por el usuario, para luego poder probarla. Estas se pueden ver en el anexo 24.

Además, con el objetivo de garantizar una correcta verificación, se utiliza la técnica de clases de equivalencia para los casos en que se prueben historias de usuario que involucran formularios con muchos campos. Esta técnica genera el conjunto reducido de pruebas con la mayor probabilidad de descubrir errores (Myers en IBM 2009). Las tablas utilizadas se pueden ver en el anexo 25.

Finalmente, siguiendo las recomendaciones de la metodología XP, se utiliza un conjunto de herramientas que permiten automatizar las pruebas propuestas tanto por el desarrollador como por el cliente. Estas se listan en la tabla 4.6. Sin embargo, el proceso de automatización no se pudo para realizar para todas las historias de usuario, por lo que en algunos casos se recurre a pruebas manuales.

Herramienta	Descripción
JWebUnit 3.0	Permite automatizar pruebas de integración, simulando la interacción del usuario con la interfaz web del sistema o la del nodo con los controladores.
DbUnit 2.4	Se utiliza para garantizar que la base de datos se encuentre en un estado conocido antes de ejecutar cualquier prueba. Se encarga de limpiar y actualizar los datos de la base de manera automática.

Tabla 4.6: Herramientas para automatización de pruebas

b) Pruebas de carga

Considerando que el sistema Pretor se diseña para atender las consultas de aproximadamente 500 computadoras, se decide evaluar su desempeño en un ambiente controlado. Para ello, se instala el sistema en un servidor con las características indicadas en la tabla 4.7. A continuación, se une a éste un total de 38 computadoras, con las características indicadas en la tabla 4.8, configurados para realizar consultas al servidor a intervalos de un minuto. Cabe resaltar que las computadoras son máquinas virtuales clonadas, por lo que las características son las mismas en todos los casos.

Para observar el comportamiento del servidor web, se recurre a la herramienta Htop, que muestra, principalmente, el porcentaje de consumo de memoria y del procesador. Por otra parte, para observar las respuestas a los nodos y el tiempo que toman, se recurre a la bitácora del sistema. Adicionalmente, se cuenta con el apoyo del administrador de la *grid*, quien se encarga de unir simultáneamente las 38 computadoras disponibles al servidor web.

Número de procesadores	1
Modelo de cada procesador	Quad-Core AMD Opteron(tm) Processor 2356
Velocidad de cada procesador en MHz	2300
Memoria RAM	2015 MB
Sistema operativo	Scientific Linux 6

Tabla 4.7: Características del servidor web de prueba

Número de procesadores	1
Modelo de cada procesador	Intel(R) Core(TM) i5-2500 CPU
Velocidad de cada procesador en MHz	3300
Memoria RAM	1 MB
Sistema operativo	Scientific Linux 6
Versión de cliente BOINC	6.12.43

Tabla 4.8: Características de las máquinas virtuales clonadas

4.2.3 Resultados de las pruebas

A continuación, se presenta los resultados obtenidos para cada tipo de prueba.

a) Pruebas de integración

En este caso todas las pruebas, tanto manuales como automáticas, son repetidas hasta que sean superadas exitosamente. Los resultados de las mismas se pueden ver en el anexo 26.

b) Pruebas de carga

Inicialmente, se observa un porcentaje de uso del 100% del procesador y una cota superior de 50% de uso de la memoria al unir simultáneamente las 38 computadoras al sistema. Sin embargo el tiempo de respuesta supera los diez minutos y el mensaje presenta errores. Al revisar la bitácora y el código fuente, se detecta un cuello de botella, ya que se intenta leer el archivo de conexión a base de datos cada vez que se intenta responder a una computadora.

Tras corregir el problema anterior, se observa nuevamente un porcentaje de uso del 100% del procesador y una cota superior de 50% de uso de la memoria al unir simultáneamente las 38 computadoras al sistema. El porcentaje de uso del procesador disminuye hasta el 3% y las respuestas son enviadas en menos de un minuto. Luego de unidos los nodos, la carga del procesador no supera el 12% y todas las consultas son atendidas en menos de medio minuto. El uso de memoria se mantiene constante y no supera el 50%.

5 Observaciones, conclusiones y recomendaciones

En esta sección final se presenta las observaciones a los eventos más importantes del desarrollo del sistema. Luego, se detalla las conclusiones obtenidas. Para terminar, se brinda una lista con recomendaciones para trabajos futuros que permitan mejorar el sistema Pretor.

5.1 Observaciones

Durante el desarrollo del proyecto de fin de carrera se obtiene un conjunto valioso de experiencias en el uso de las técnicas, metodologías y herramientas. En los siguientes párrafos se detalla las tres experiencias consideradas más importantes por el desarrollador.

Primero, el proyecto manifiesta un claro desfase de tiempo. En el plan inicial del proyecto se estimó un esfuerzo de 743 horas, a lo largo de 206 días. El plan corregido y actualizado requiere de un esfuerzo de 1571 horas, a lo largo de 302 días. Esto se debe, primero, a que la metodología XP permite que los requerimientos obtenidos varíen en el tiempo, pudiendo modificarse o

incrementarse. Segundo, también se debe a que, dado que el producto es de carácter innovador, es posible que se descubran nuevos requerimientos que agregan valor al cliente durante el desarrollo o que surjan problemas técnicos no previstos. Por último, como tercera causa, se tiene que, al inicio del proyecto, no se contempló el tiempo y esfuerzo requeridos para crear pruebas automáticas del sistema. Sin embargo, a pesar del desfase indicado, XP garantiza que las funcionalidades obtenidas son de valor para el cliente, puesto que este se encuentra siempre presente durante el desarrollo, evaluándolas.

Segundo, las pruebas automatizadas del sistema, si bien requieren un esfuerzo adicional de programación, son útiles para verificar el funcionamiento del sistema y reducir considerablemente la cantidad de errores. En particular, durante el desarrollo, ocurrió que en las etapas iniciales el cliente se dedicó a ejecutar de manera manual y repetida las pruebas de integración. Sin embargo, conforme se incrementaban las funcionalidades, era cada vez más tedioso volver a probar todo el sistema para asegurarse que funcionaba correctamente. Es por ello que se decidió automatizar las pruebas, siguiendo las recomendaciones de la metodología XP.

Tercero, como última experiencia importante, se tiene que el método de prototipos es una herramienta útil para la captura de requerimientos, que permite involucrar al cliente y transmitirle fácilmente la idea que se tiene del sistema. En particular, los prototipos elaborados para este proyecto de fin de carrera fueron el elemento clave que permitió que el cliente aprobara su desarrollo.

Para concluir, se reconoce la constante colaboración del personal de la DIA para la realización de este proyecto. En particular, el código y diseño del sistema Legión que se toma como base para Pretor. También, la consultoría en temas de bases de datos y programación que ofrece el administrador de los sistemas de la DIA. Además, la consultoría en temas de seguridad y programación junto con las experiencias previas con el sistema BOINC que proveen los desarrolladores del sistema Legión. Finalmente, el apoyo constante que brinda el asesor, sin lo cual no sería posible realizar el presente proyecto de fin de carrera.

5.2 Conclusiones

A partir de las experiencias que se obtiene durante el desarrollo del proyecto, se concluye lo siguiente:

- La administración de una *grid* computacional puede representar una tarea muy compleja dependiendo del tamaño de la misma, medido en términos de la cantidad de recursos que contiene. A partir de lo que se indica en este proyecto se concluye que una solución factible es recurrir a herramientas informáticas, como el mecanismo administrador de cuentas del *middleware* BOINC, que permitan automatizar las labores administrativas.

- La elaboración de prototipos facilita la comunicación con el cliente y mejora la comprensión global del sistema. Al construirlos de manera conjunta, entre desarrollador y cliente, es posible realizar una mejor toma de requerimientos funcionales. Sin embargo, esto no quita la necesidad de elaborar también un documento que detalle las funcionalidades que implica cada prototipo. Este se utiliza tanto para dimensionar el alcance del proyecto como para poder verificar y validar el producto.

- Las herramientas de desarrollo web de código abierto facilitan, agilizan y mejoran la construcción del sistema. Primero, facilitan, porque al ser de código abierto, es posible adaptarlas de acuerdo a las necesidades del cliente sin incurrir en nuevos costos. Segundo, agilizan, porque transmiten el conocimiento generado por otros programadores, evitando el tener que crear nuevas soluciones para problemas ya resueltos. Por último, mejoran, porque permiten enriquecer las interfaces de usuario con elementos previamente elaborados.

- Las pruebas automatizadas facilitan la verificación del sistema y reducen la cantidad de errores considerablemente. En particular, se obtiene una mayor ventaja cuando las historias de usuario a probar involucran formularios con una gran cantidad de campos, como es el caso de las de los módulos de grupos y nodos.

5.3 Recomendaciones y Trabajos futuros

Con respecto a las posibles mejoras del sistema, se ha identificado cuatro horizontes u objetivos generales. Primero, reducir el nivel de intervención humana en el sistema. Segundo, integrar los sistemas Legión y Pretor. Tercero, adaptar Pretor para que se pueda utilizar a través de dispositivos móviles. Cuarto, agregar gráficos con datos de la evolución histórica de los recursos del sistema.

El sistema Pretor se desarrolla con el objetivo de dar soporte a las labores de administración de la *grid* computacional de la PUCP. Para ello, se elabora un conjunto de funcionalidades que requieren, en su mayoría, de intervención humana para poder ejecutarse. En ese sentido, se sugiere un horizonte de mejora del sistema orientado a reducir el nivel de intervención humana. En particular, se tiene los siguientes casos:

- Incluir funcionalidades que permitan elaborar un calendario con los proyectos científicos y los grupos de computadoras que los deben apoyar. De esta manera, bastaría con que un administrador de la *grid* configure en determinada fecha el calendario de proyectos y grupos, para que luego se dedique solo a supervisar que todo se ejecuta según lo planificado, de manera automática. Adicionalmente, se podría incluir funcionalidades encargadas de monitorizar la ejecución, que envíen mensajes automáticos en cuanto surja algún problema.

- Incluir funcionalidades que permitan que el propio sistema determine el calendario de asignación de proyectos a grupos. Todo ello de manera automática, tomando en cuenta la necesidad de recursos de los proyectos, su prioridad y la disponibilidad y capacidad de las computadoras de la *grid*. De esta manera, se reduce aún más el nivel de intervención humana en el sistema, limitando la labor del administrador a recibir mensajes automáticos en caso surjan problemas durante la ejecución para poder tomar las acciones correspondientes.

Con respecto al segundo horizonte, dada la afinidad que existe entre los sistemas Legión y Pretor, se plantea su integración en un solo sistema, ya que ambos cubren

dos aspectos diferentes y necesarios para poder crear y mantener un VCSC en la PUCP.

En el tercer horizonte, se considera la adaptación del sistema Pretor para que pueda ser utilizado mediante dispositivos móviles, tomando en cuenta su nivel de difusión actual y las facilidades de conexión inmediata y ubicua que ofrecen. A nivel técnico, la adaptación implica incluir funcionalidades gráficas que exploten las características de las pantallas táctiles y realcen el aspecto visual del sistema.

Finalmente, en el cuarto horizonte, se propone modificar la estructura actual del sistema, de modo que permita llevar un registro diario del estado de los recursos del sistema. De esta manera, es posible, por ejemplo, presentar la evolución de la contribución de créditos de una determinada área académica o el nivel de consumo de créditos en un proyecto, a lo largo del último mes. También, se puede dar a conocer en cuáles días del mes estuvo inactiva una computadora o grupo de computadoras.

6 Bibliografía

ABRAHAMSSON, Pekka et al.

2006 *Extreme Programming and Agile Processes in Software Engineering.* Oulu: Springer.

2002 *Agile software development methods. Review and analysis.* Espoo: Otamedia Oy.

APACHE SOFTWARE FOUNDATION

2012 *Apache Tomcat.* Consulta: 07 de abril de 2012. <http://tomcat.apache.org/>

BOINCSTATS TEAM

2011 *Welcome to the BOINC Account Manager (BAM!).* California: Universidad de California. Consulta: 10 de setiembre de 2011. <http://boincstats.com/bam/>

CEREJO, Lyndon

2010 *Design Better and Faster With Rapid Prototyping.* Consulta: 24 de marzo de 2011. <http://www.smashingmagazine.com/2010/06/16/design-better-faster-with-rapid-prototyping/>

CHOPRA, Vivek et al.

2005 *Beginning JavaServer Pages.* Indiana: Wiley Publishing Inc.

COOPER, Robert y Scott Edget

2011 *Stage-Gate – Your Roadmap for New Product Development.* Product Development Institute Inc. Consulta: 20 de setiembre de 2011. <http://www.prod-dev.com/stage-gate.php>

2007 *Generating breakthrough new product ideas.* Canadá: Product Development Institute Inc.

DÍAZ, Oscar

2008 "Desktop Grid Computing con BOINC". Ponencia presentada en el evento Linux Week 2008. Lima. Consulta: 07 de setiembre de 2011. <http://tuxpuc.pucp.edu.pe/noticia/linux-week-2008-desktop-grid-computing-con-boinc>

DIRECCIÓN DE INFORMÁTICA ACADÉMICA

2011 *Legión*. Lima: Pontificia Universidad Católica del Perú. Consulta: 30 de agosto de 2011. <http://legion.pucp.edu.pe>

2006 *Presentación.* Lima: Pontificia Universidad Católica del Perú. Consulta: 10 de setiembre de 2011. <http://dia.pucp.edu.pe/portal/content/view/99/938/>

EVANS-TOYNE, Sarah et al.

2011 *New DIY supercomputer saves £1,000s.* Londres: Universidad de Westminster. Consulta: 10 de setiembre de 2011. <http://www.westminster.ac.uk/about/news-and-events/news/2011/university-of-westminter-launches-new-diy-supercomputer-saving-hundreds-of-thousands-of-pounds>

FONSECA, Pablo y Oscar Díaz

2011 *Legión*. Lima: Pontificia Universidad Católica del Perú. Consulta: 10 de setiembre de 2011. <http://legion.pucp.edu.pe/wiki>

FOWLER, Martin

2004 *Is Design Dead?.* Consulta: 29 de marzo de 2012. <http://martinfowler.com/articles/designDead.html>

GAMMA, Erich et al.

1998 *Design Patterns. Elements of Reusable Object-Oriented Software.* Baam: Addison Wesley Longman, Inc.

HARBAUGH, Logan G.

2009 *Choosing the best server OS: Linux vs. Windows comparisons.* Consulta: 14 de abril de 2012 <http://searchdatacenter.techtarget.com/tip/Choosing-the-best-server-OS-Linux-vs-Windows-comparisons>

IBERICO, Martín Alberto

2009 *Administrador de proyectos de grid computing que hacen uso de la capacidad de cómputo ociosa de laboratorios informáticos.* Tesis de licenciatura en Ciencias e Ingeniería con mención en Ingeniería Informática. Lima: Pontificia Universidad Católica del Perú, Facultad de Ciencias e Ingeniería.

IBM

2012 *World community grid.* Consulta: 10 de marzo de 2012. <http://www.worldcommunitygrid.org>

2009 *Guideline: Equivalence Class Analysis.* Consulta: 07 de abril de 2012. <http://epf.eclipse.org/wikis/xp/xp/guidances/guidelines/equivalence_class_analysis_E178943D.html>

JEFFRIES, Ronald E.

2012 *What is Extreme Programming?* Consulta: 24 de marzo de 2012. <http://xprogramming.com/what-is-extreme-programming>

LOMBRAÑA, Daniel

2011 *Jarifa*. Extremadura: Universidad de Extremadura. <https://github.com/teleyinex/jarifa/wiki>

2010 “Jarifa: setting up an institutional desktop grid system”. Clase maestra sobre computación voluntaria. Beijing. Consulta: 06 de setiembre de 2011. <http://indico.ihep.ac.cn/conferenceDisplay.py?showSession=all&showDate=all&fr=no&confId=1454>

MALTON, Christopher y Rytis Slatkevius

2010 "Unofficial BOINC Wiki". Consulta: 30 de abril de 2012.
<http://www.boinc-wiki.info/>

MEYERS, Peter

2012 *25-point Website Usability Checklist.* User-Effect, Inc. Consulta: 05 de abril de 2012.
<http://www.usereffect.com/topic/25-point-website-usability-checklist>

MYBATIS

2011 *Introduction to MyBatis Generator.* Consulta: 05 de abril de 2012.
<http://www.mybatis.org/generator/>

ORACLE

2012 *Java EE.* Consulta: 07 de abril de 2012.
<http://www.oracle.com/technetwork/java/javaee/overview/index.html>

2011a *Java Platform Overview. JRE and JDK.* Consulta: 07 de abril de 2012.
<http://docs.oracle.com/javase/7/docs/technotes/guides/index.html#jre-jdk>

2011b *Top 10 Reasons to Choose MySQL for Web-based Applications.*

P-GRADE Portal

2010 *User manual. An introduction without tears.* Consulta: 15 de abril de 2012.
<http://portal.p-grade.hu/manual/user/v210/UsersManualRelease2_10.html>

PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ (PUCP)

2012 *Información económica.* Consulta: 25 de marzo de 2012.
<http://www.pucp.edu.pe>

QUINTANA, Juan Carlos

2009 "Investigación con Legión". *Neo.* Lima, año 1, número 3, pp. 1-4.

RÍOS, Genghis et al.

2009 "Legión – Sistema de Computación en Grid". Ponencia presentada en el evento Conferencia Latinoamericana de Computación de Alto Rendimiento. Mérida. Consulta: 07 de setiembre de 2011. <http://erevistas.saber.ula.ve/index.php/memoriasula/article/viewArticle/104>

SUPERINTENDENCIA NACIONAL DE ADMINISTRACIÓN TRIBUTARIA (SUNAT)

1994 *Decreto Supremo Nº 122-94-EF*. *Reglamento de la ley del impuesto a la renta.* 19 de setiembre.

TANEMBAUM, Andrew S.

2008 *Modern Operating Systems.* Tercera edición. New Jersey: Prentice Hall.

UNIVERSIDAD DE CALIFORNIA

2012a *BOINC.* California: Universidad de California. Consulta: 01 de mayo de 2012. <http://boinc.berkeley.edu/trac/wiki>

2012b *SETI@home.* California: Universidad de California. Consulta: 01 de mayo de 2012. <http://setiathome.berkeley.edu/>

2007 *Create a Virtual Campus Supercomputing Center (VCSC)*. California: Universidad de California. Consulta: 15 de setiembre de 2011. <http://boinc.berkeley.edu/trac/wiki/VirtualCampusSupercomputerCenter>