ADR

Septiembre 2010

Indice

[Indice 2](#_Toc273362682)

[1 Arquitectura 4](#_Toc273362683)

[1.1 Arquitectura general 4](#_Toc273362684)

[1.2 Lenguaje de programación 6](#_Toc273362685)

[1.3 Tecnologías 7](#_Toc273362686)

[1.4 Motor de bases de datos 8](#_Toc273362687)

[1.5 Comparativa en precios con Oracle 13](#_Toc273362688)

[1.6 Ponderación de Motores 14](#_Toc273362689)

[1.7 Elección del Motor 15](#_Toc273362690)

[1.8 Middleware de base de datos 18](#_Toc273362691)

[Arquitectura del sistema con pgpool-II 18](#_Toc273362692)

[1.9 Sistema operativo 20](#_Toc273362693)

[1.10 Servidor de aplicación 22](#_Toc273362694)

[2 Volumen de datos estimado para el sistema 23](#_Toc273362695)

[3 Infraestructura desarrollo interna 24](#_Toc273362696)

[3.1 Esquema de la infraestructura 24](#_Toc273362697)

[3.2 Herramientas para desarrollo 25](#_Toc273362698)

[3.2.1 Configuración de servidores 25](#_Toc273362699)

[Servidor 1 25](#_Toc273362700)

[Servidor 2 25](#_Toc273362701)

[3.2.2 Compra de servidores para desarrollo 27](#_Toc273362702)

[Servidor 1 27](#_Toc273362703)

[Servidor 2 27](#_Toc273362704)

[3.3 Estaciones de trabajo 28](#_Toc273362705)

[3.3.1 Opciones de Compra Para el Equipo de Desarrollo 29](#_Toc273362706)

[3.3.2 Tabla ponderación de equipos Terminales Para desarrollo 30](#_Toc273362707)

[3.3.3 Selección de Laptop para líder de proyecto 30](#_Toc273362708)

[3.4 Sistema Operativo Para las estaciones de trabajo 31](#_Toc273362709)

[4 Infraestructura Producción 32](#_Toc273362710)

[4.1 Esquema de la infraestructura 32](#_Toc273362711)

[4.2 Disponibilidad y Rendimiento 32](#_Toc273362712)

[4.3 Esquema de Disponibilidad 7 x 24 33](#_Toc273362713)

[4.4 Configuración de servidores para producción 33](#_Toc273362714)

[4.5 Software Necesario 34](#_Toc273362715)

[4.6 Elección de servidores para producción 35](#_Toc273362716)

[4.6.1 Servidor Tipo A, cantidad: 3 35](#_Toc273362717)

[4.6.2 Servidor Tipo B 35](#_Toc273362718)

[4.7 Storage Nas 36](#_Toc273362719)

[4.8 Nivel Raid 37](#_Toc273362720)

[4.9 Integración Con Sistema de Comunicaciones 38](#_Toc273362721)

[5 Oficinas para el desarrollo 39](#_Toc273362722)

[5.1 Puestos 39](#_Toc273362723)

[5.2 Plano de la Planta alquilada 40](#_Toc273362724)

[5.2 Hosting de Servers 41](#_Toc273362725)

[6 Empleados 42](#_Toc273362726)

[6.1 Roles particulares 42](#_Toc273362727)

[7 Costos / Capital Inicial 43](#_Toc273362728)

[8 Costos Mensuales 45](#_Toc273362729)

[8.1 Por Mes 45](#_Toc273362730)

[8.2 Totales 45](#_Toc273362731)

[9 Entregas del proyecto 47](#_Toc273362732)

[9.1 Detalle en Las Entregas 48](#_Toc273362733)

[10 Costo y Formas de Pago 50](#_Toc273362734)

[11 Ganancias Esperadas 51](#_Toc273362735)

# Arquitectura

## Arquitectura general

**Arquitectura de tres capas con cliente desktop.**

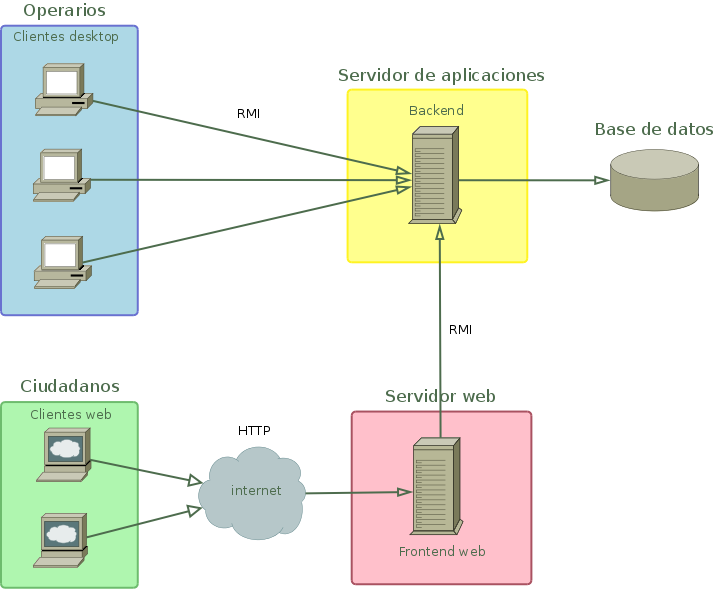
Cliente Desktop: Se optó por un cliente Desktop por sobre uno Web debido a que se trata de un sistema crítico en el cual se deben atender a las peticiones entrantes con la mayor velocidad y calidad posible.

Una aplicación Desktop puede poseer una interfaz gráfica más robusta, interactiva y performante que la que podría lograrse con tecnologías web, cosa que es imprescindible para los usuarios intensivos del sistema, en este caso los operadores, ya que el rendimiento de los mismos estará relacionado con la interacción con el sistema.  
El número de estos operadores es limitado y conocido, las PCs que utilizarán serán parte de la organización y serán mantenidas por personal interno, por lo cual no habrá mayores inconvenientes para la instalación y actualización del cliente desktop, que es una de las desventajas que posee esta arquitectura frente a una Web.

Otra ventaja de la elección de una arquitectura Desktop frente a una Web, está relacionada con la productividad en el desarrollo, ya que el tiempo en desarrollar un cliente Desktop es inferior que desarrollar un cliente Web.

Multicapa: A su vez no es conveniente que el cliente desktop se conecte directamente a la base de datos, el sistema tiene la magnitud suficiente para justificar el uso de un application server para ejecutar una aplicación de backend que brinda más robustez y seguridad.  
Este tipo de arquitectura mantiene el control del acceso a la base de datos, así se pueden controlar los problemas que la concurrencia pueda traer.  
Esta aplicación además de tener acceso a la base de datos, tendrá prácticamente toda la lógica y las reglas de negocio de la aplicación. De esta forma se puede tener una alta integridad, mantener el servidor en un ambiente seguro es más fácil de lograr que el resto de las Pcs donde se ejecutan las aplicaciones cliente. También será más fácil el mantenimiento ya que la simplicidad de la interfaz reduce los errores y la necesidad de actualización, así el único componente a actualizar será la aplicación del servidor.

Sitio Web: Es necesario desarrollar un sistema web para que los ciudadanos puedan consultar el estado de sus trámites y comunicarse con los operadores por medio de chat. Esta será una aplicación independiente que se ejecutará en un application server al igual que el backend. Por lo que pude correr en el mismo servidor o en otro independiente para dar escalabilidad.  
Esta aplicación web tendrá la menor cantidad de lógica posible y se conectará al mismo backend que el cliente desktop para consumir servicios. Así se reutiliza la lógica del backend ya que el tipo de consultas que se hace son similares y se unifica el acceso a la base de datos.



## Lenguaje de programación

**Lenguaje de programación Java**

Tanto el cliente desktop como el backend y la aplicación web serán desarrollados en **Java**, las razones por las cuales se eligió este lenguaje sobre otros disponibles son:

Amplia disponibilidad de recursos capacitados: Es uno de los lenguajes más utilizados y es fácil encontrar personal con experiencia en el mercado. No existe la necesidad de capacitar ingresantes. Estos factores incrementan la posibilidad de éxito del proyecto.

Madurez y soporte: Está establecido en el mercado y fue desarrollado a través de los años por Sun, ahora por un comité formado por importantes empresas y siempre se mantuvo la compatibilidad entre versiones.

Disponibilidad de Frameworks y librerías: Existe una vasta selección de herramientas maduras para el desarrollado, tanto para la parte de servidor como de presentación. Al utilizar tecnologías conocidas y probadas que resuelven los problemas más frecuentes de los sistemas transaccionales evitamos “reinventar la rueda” reduciendo el tiempo y los riesgos.

Multiplataforma: Se consideró importante para el desarrollo de este sistema la independencia de plataforma, la capacidad de remplazar o combinar diferentes sistemas operativos sin ningún tipo de desarrollo. Descartando así los lenguajes que compilen a código máquina y los lenguajes que no estén soportados en los sistemas operativos más comunes.

Performance: Si bien java no se compila a código máquina, se compila a bytecode y es ejecutado por una máquina virtual. Esta máquina virtual es la Java HotSpot Server VM que tiene la funcionalidad JIT (Just in time) compilation esta capacidad analiza el código que se ejecuta reiteradamente y lo optimiza en memoria haciendo las sucesivas ejecuciones más rápidas, a veces incluso que si se hubiera compilado a máquina. Esta ventaja es válida para algunos tipos de operaciones más que otras (por ejemplo el acceso a entrada/salida), mucho más frecuentes en un sistema transaccional que las que no (operaciones de punto flotante).

Costos: No posee costos de licencias y, al ser uno de los lenguajes más utilizados, el personal capacitado es más económico que en otros lenguajes más específicos.

## Tecnologías

Se programará utilizando la versión *Standard Edition del JDK 1.6* y se utilizarán las siguientes librerías y tecnologías.

SWING: Para la interfaz de usuario en el cliente desktop. Es muy extensible y presenta mayor facilidad que la alternativa AWT. Es totalmente independiente del sistema operativo y en las versiones más recientes imita el look and feel nativo.  
Si bien posee algunas desventajas como un alto consumo de recursos del sistema y lentitud con respecto a aplicaciones nativas, la aplicación requerida no necesita de componentes gráficos “pesados”, con lo cual lo requerimientos pueden ser satisfechos por SWING, y además se tiene la ventaja de que las implementaciones de los componentes son cien por ciento java puro, lo cual permite adaptar fácilmente el aspecto y comportamiento de tales componentes, facilitando así el trabajo de los programadores, lo cual deriva en una mayor productividad

Tomcat: Para el Application server, se eligió JakartaTomcat para ejecutar el servicio al que se conectarán los usuarios. Si bien el backend de nuestra aplicación no hará uso de servlets (pero si la aplicación web), Tomcat nos brinda un ambiente de ejecución controlado y configurable. Cada vez que una petición llegue al servidor se utilizará un thread para ejecutar la lógica de negocio. Manejar el ciclo de vida, la planificación y comunicación de threads es menos costoso que lo equivalente para procesos, esto aliviará la carga del servidor y hará el sistema más performante y escalable.

RMI: La comunicación entre los clientes desktop y web con el backend será a través de RMI (RemoteMethodInvocation), este es un protocolo nativo de Java SE, es de simple aplicación y eficiente.  
No debemos implementar un complejo protocolo propio ni debemos preocuparnos por la forma de codificar nuestros objetos para transmitirlos y tenemos la certeza de que funciona.  
Serializa los datos en streams de bytes que es más rápido al ser transmitido en una red que hacerlo en texto como lo haría un web service. No necesitamos la ventaja de un web service (independizar el cliente del servidor) porque ambos serán desarrollados en java, lo que nos da la libertad de elegir un protocolo nativo.

Hibernate: Como framework de ORM para la persistencia. Se decidió utilizar una herramienta de ORM (objectrelationalmapping) para reducir los problemas de desarrollo que acarrea lidiar con la integración de un lenguaje orientado a objetos y una base de datos relacional. La lógica de la aplicación se modela en objetos de negocio que son difíciles de volcar en un modelo relacional, demandando complejos queries y storeprocedures difíciles de desarrollar y mantener. Queremos evitar esto en la medida posible, seleccionando un ORM. Hibernate es el elegido por ser el más usado y probado, se sabe que es ampliamente configurable, de rendimiento aceptable y fácil aplicación. Otras tecnologías como EJB (también útil para la comunicación) presentan mayor complejidad y no presentan ventajas para el tamaño de nuestro sistema.

Otras: El resto de las tecnologías necesarias se decidirán sobre la marcha del desarrollo por el arquitecto de la aplicación.

## Motor de bases de datos

**Tabla Comparativa de Motores: Postgres vs MySql**

|  | MySQL | PostgreSQL |
| --- | --- | --- |
| Velocidad | **El motor MyISAM de MySQL funciona más rápido que PostgreSQL sobre los queries simples y cuando la concurrencia es baja o sigue ciertos patrones (por ejemplo, los inserts se realizan en tablas optimizadas y sin bloqueos, count (\*) es muy rápido).**  **Su principal objetivo de diseño fue la VELOCIDAD. Se sacrificaron algunas características esenciales en sistemas más "serios" con este fin.** | **La ventaja en velocidad de PostgreSQL puede ver drásticamente en un ambiente de grandes procesadores multi-core. Aunque, Generalmente, es más lento que MySQL**  **Postgres intenta ser un sistema de bases de datos de mayor nivel que MySQL, a la altura de Oracle, Sybase o Interbase.** |
| Utilización de Recursos | **MySQL se centra en tecnologías comunes de bajo rendimiento y el uso de hardware básico.**  **Consume MUY POCOS RECURSOS, tanto de CPU como de memoria.** | **La configuración por defecto de PostgreSQL fue diseñada para ejecutarse en sistemas con poca memoria.** |
| Compresión de datos | **Hasta la versión 5.1 de MySQL sus motores de almacenamiento de alto rendimiento no soportan compresión sobre la marcha.** | **PostgreSQL puede comprimir y descomprimir sus datos sobre la marcha con un rápido sistema de compresión para encajar más datos en un espacio de disco asignado. La lectura de datos tarda menos IO, por lo que lee datos con mayor rapidez.** |
| Concurrencia |  | **PostgreSQL escala mucho mejor, tanto en términos de la utilización de un hardware de alto rendimiento, y al hacer frente a la concurrencia.** |
| Asíncrona I/O | **MySQL carece de soporte Async, aunque algunos controladores se han creado para tratar de superar esta deficiencia (perlruby).** | **PostgreSQL soporta una completa API asincrónica para el uso de las aplicaciones cliente.** |
| COUNT(\*) |  | **PostgreSQLCOUNT(\*) es muy lento porque en lugar de contar las filas utilizando un índice de exploración, debe de recorrer toda la tabla secuencialmente. PostgreSQL implementa COUNT(\*) de esta manera debido a la forma en la cual el MVCC (sistema de concurrencia del PostgreSQL) almacena la visibilidad de las transacciones en la fila de datos y no en el índice.** |
| ACID (Atomicidad, Coherencia, Aislamiento y Durabilidad | **InnoDB de MySQL proporciona cumplimento de ACID a nivel del motor** | **PostgreSQL es plenamente compatible con ACID.** |
| Facilidad de uso | **MySQL tiene más “gotchas”(característica o función que funciona como se ha publicado – pero no como se esperaba.) que PostgreSQL** |  |
| Procedimientos almacenados | **Soportan procedimientos almacenados.** | **Soportan procedimientos almacenados.**  **Tiene Mejor Soporte para procedimientos en el servidor.** |
| Triggers | **Soportan triggers.** | **Soportan triggers.**  **Tiene mejor soporte para triggers.** |
| Replicación y Alta Disponibilidad | **Soportan replicación**  **A partir de la versión 5.1, MySQL soporta dos formas de replicación; replicación basada en declaración (SBR) y replicación basada en la fila (RBR).** | **Soportan replicación**  **PostgreSQL es modular por su diseño, y la replicación no está en el núcleo. Hay varios paquetes que permiten la replicación en PostgreSQL: PGCluster, Slony-I, DBBalancer, pgpool, PostgreSQLtablecomparator** |
| Particionado | **MySQL soporta varios tipos de particionamiento horizontal: RANGE, LIST, HASH, KEY.** | **PostgreSQL sólo es compatible con particionado RANGE y LIST.** |
| Motores de almacenamiento de datos | **MySQL 5.1 soporta nativamente 9 motores de almacenamiento: MyISAM, InnoDB, NDB, Cluster, MERGE, MEMORY (HEAP), FEDERATED, ARCHIVE, CSV, BLACKHOLE** | **PostgreSQL soporta un motor, por defecto su sistema de almacenamiento Postgres (Postgres Storage System).** |
| Licencias | **El código fuente de MySQL está disponible bajo los términos de la Licencia Pública General de GNU, que también se inscribe en las definiciones de Software Libre y Open Source.**  **También está disponible bajo un acuerdo de licencia propietaria, que está normalmente destinado a ser utilizado por aquellos que desean incorporar código de MySQL sin tener que liberar el código fuente para toda la aplicación. Esto significa que MySQL se puede distribuir con o sin código fuente, pero para no distribuir el código fuente se requiere el pago de MySQL AB para una licencia comercial.** | **PostgreSQL viene con un estilo de licencia BSD, que se inscribe en la definición de Software Libre y Open Source.** |
| Desarrollo | **MySQL es propiedad y está patrocinado por una sola empresa con fines de lucro, la empresa sueca MySQL AB, que posee los derechos de autor a la mayoría del código.** | **PostgreSQL no es controlada por una sola empresa, sino que se basa en una comunidad global de desarrolladores y empresas para desarrollarlo.** |
| Escalabilidad | **No implementa una buena escalabilidad** | **Es capaz de ajustarse al número de CPUs y a la cantidad de memoria que posee el sistema de forma óptima, haciéndole capaz de soportar una mayor cantidad de peticiones simultáneas de manera correcta** |
| Tamaño de Registros | **No hay límites en el tamaño de los registros** | **Limite del tamaño de cada fila de las tablas a 8k (se puede ampliar a 32k recompilando, pero con un coste añadido en el rendimiento).** |
| Integración con PHP | **Mejor integración con PHP** | **Menos funciones en PHP** |
| Subselect | **No soporta transacciones, "roll-backs" ni subselects** | **Soporta Subselects** |
| Integridad de Datos | **PostgreSQL ofrece una garantía de integridad en los datos mucho más fuerte que MySQL** |  |

**Benchmark SPEC**: Comparación entre PostgreSQL 8.2 y MySQL 5.0

**Caso 1**: 8 cores, 2 instancias de servidores J2EE.



**Caso 2**: 12 cores, 3 instancias de servidores J2EE



PostgreSQL presenta menor rendimiento en sistemas chicos pero presenta mayor escalabilidad ya que al agregarle carga y mejorando el hardware supera a MySQL.

Lo que se puede ver haciendo un análisis de estos benchmarks es el impacto de agregar concurrencia al motor.

Si nos movemos del Caso 1 al Caso 2 la diferencia es que se agregó un 50% más de concurrencia y un 50% más de procesadores, un motor ideal no hubiese perdido rendimiento y hubiese obtenido la misma cantidad de puntos (se espera que sea lineal en el peor caso, haciendo el comportamiento predecible).

El resultado del cambio es que PostgreSQL perdió un 4,57%, que si lo comparamos con el 66,13% que perdió MySQL notamos no solo que PostgreSQL tuvo más puntos sino que además escala mucho mejor, lo que consideramos más importante.

¿Por qué es tan importante la escalabilidad? A este punto del desarrollo de la solución no está completamente decidido el hardware del servidor y no es bueno seleccionar el motor únicamente porque se adapte a nuestro hardware.

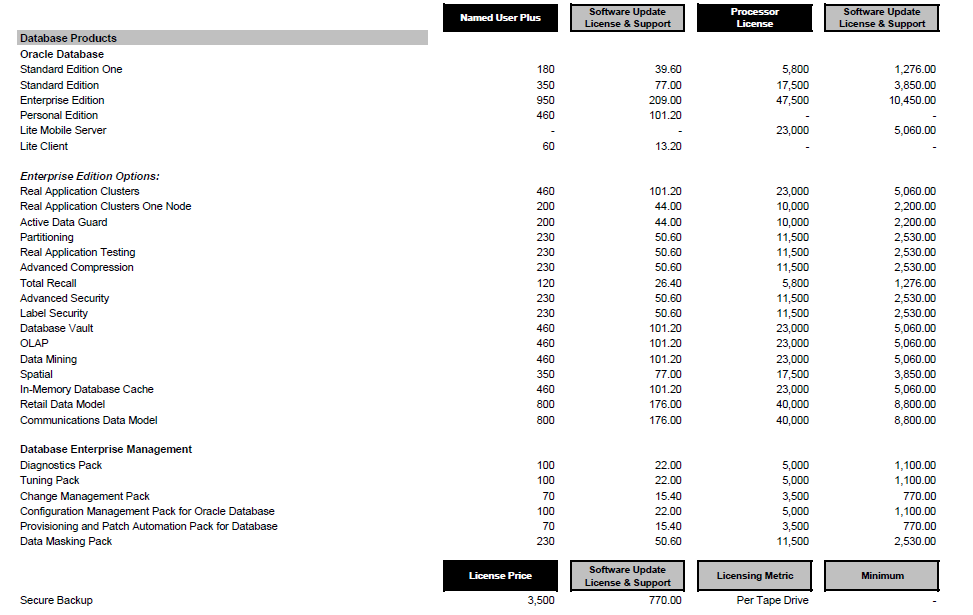
Por la magnitud del sistema lo más probable que el servidor elegido se parezca más al primer caso que al segundo, sin embargo en un futuro la concurrencia del sistema podría crecer. Si esto sucede queremos que la solución sea simplemente adquirir un servidor con más prestaciones y no tener que optimizar queries, o peor, tener que cambiar de motor.

Balanceando ambos puntos de vista, el de puro rendimiento en el hardware actual y el de prevención al crecimiento del uso, optamos por PostgreSQL que no pierde por mucho contra MySQL en el corto plazo pero nos garantiza la escalabilidad a largo plazo.

Fuente: <http://www.spec.org/jAppServer2004/results/res2007q3/jAppServer2004-20070606-00065.html>

## Comparativa en precios con Oracle

Esta es una vista a los precios de las nuevas opciones de Oracle:

**NamedUser**

En contraposición, postgreSQL presenta una licencia BSD. La licencia BSD es la [licencia de software](http://es.wikipedia.org/wiki/Licencia_de_software) otorgada principalmente para los sistemas [BSD](http://es.wikipedia.org/wiki/BSD) (*Berkeley Software Distribution*). Es una [licencia de software libre permisiva](http://es.wikipedia.org/wiki/Licencia_de_software_libre_permisiva) como la licencia de [OpenSSL](http://es.wikipedia.org/wiki/OpenSSL) o la [MIT License](http://es.wikipedia.org/wiki/MIT_License). Esta licencia tiene menos restricciones en comparación con otras como la [GPL](http://es.wikipedia.org/wiki/Licencia_p%C3%BAblica_general_de_GNU) estando muy cercana al dominio público. La licencia BSD al contrario que la GPL permite el uso del código fuente en software no libre.

## Ponderación de Motores

|  |  |  |  |
| --- | --- | --- | --- |
|  | Total Punaje | MySQL | PostgreSQL |
| Robustez | 20 | 10 | 20 |
| Escalabilidad | 10 | 5 | 10 |
| Velocidad | 20 | 20 | 10 |
| Licencias | 5 | 3 | 5 |
| Disponibilidad | 10 | 6 | 8 |
| Compresión | 5 | 0 | 5 |
| Utilización de Recursos | 20 | 20 | 15 |
| Facilidad de Uso | 10 | 10 | 6 |
| Total | 74 | 79 |  |

## Elección del Motor

**PostgreSQL9.0**

Este motor permite optimizar su rendimiento de manera completa ya que provee un gran número de configuraciones para asegurar un uso óptimo de los recursos. Optimizar PostreSQL en un server con una alta capacidad de cómputo provee una solución adecuada para aplicaciones donde se exija un número elevado de transacciones concurrentes. Cada vez que PostgreSQL recibe una petición desde un cliente (aplicación) se crea un proceso que es el encargado de atender todas las peticiones desde el cliente. Debido a esto, es muy simple configurar un pool de conexiones que estén preparadas para poder recibir peticiones desde los clientes.

Como referencia, un servicio PostgreSQL configurado de manera óptima en un servidor con 8GB de memoria RAM y un procesador QuadCore de 2.5 Ghz, es capaz de procesar más de 8000 transacciones simultáneas.

**Acerca de PostgreSQL**

PostgreSQL es un poderoso motor de base de datos open source. Tiene más de 15 años de desarrollo activo y una probada arquitectura que ha ganado una fuerte reputación por confiabilidad, integridad de datos, y corrección. Es compatible con los SO más conocidos, incluyend Linux, UNIX (AIX, BSD, HP-UX, SGI IRIX, Mac OS X, Solaris, Tru64), y Windows. Es ACID ‘compliant’, soporta foreignkeys, joins, views, triggers, y storedprocedures (en muchos lenguajes). Incluye la mayoría de los tipos de datos SQL 2008, incluyendo INTEGER, NUMERIC, BOOLEAN, CHAR, VARCHAR, DATE, INTERVAL, and TIMESTAMP. También soporta almacenamiento de grandes objetos de datos binarios, incluyendo imagenes, sonidos y video. Tiene interfaces nativas de programación para C/C++, Java, .Net, Perl, Python, Ruby, Tcl, ODBC, entre otros, y [documentación](http://www.postgresql.org/docs/manuals/#_blank) excepcional.

Hay sistemas activos de PostgreSQL en ambientes de producción que soportan hasta 4 TB de datos.

Los límites generales de PostgreSQL pueden encontrase en esta tabla:

|  |  |
| --- | --- |
| **Limite** | **Valor** |
| Tamaño Máximo de base de datos | Ilimitado |
| Tamaño Máximo de tabla | 32 TB |
| Tamaño Máximo de Fila | 1.6 TB |
| Tamaño Máximo de campo | 1 GB |
| Cantidad Máxima de filas por tabla | Ilimitado |
| Cantidad máxima de Columnas por tabla | 250 - 1600 dependiendo del tipo |
| Cantidad Máxima de índices por tabla | Ilimitado |

PostgreSQL ha obtenido el reconocimiento de sus usuarios y de la industria, incluyendo el ‘Linux New Media Award’ por mejor sistema de bases de datos y cinco veces ganador del 'The Linux JournalEditors' ChoiceAward’ por mejor DBMS.

Prestaciones – Standards

Su implementación respeta fuertemente el standard ANSI-SQL:2008. Tiene soporte total para subconsultas (incluyendo subselects en clausulas FORM), ‘read-committed’ y serializable como niveles de aislamiento.

Las prestaciones para la integridad de datos incluyen primarykeys (compuestas), foreignkeys con restricciones y updates/deletes en cascada, checkconstraints, uniqueconstraints, y notnullconstraints.

También tiene muchas extensiones y características avanzadas. Entre ellas se encuentran columnas auto-incrementables por medio de secuencias, y LIMIT/OFFSET permitiendo el retorno de result sets parciales. PostgreSQL soporta índices compuestos, únicos, parciales y funcionales los cuales pueden utilizar cualquiera de sus métodos de almacenamiento: árbol-B, árbol-R, ahsh, o GiST.

La indexación [GiST](http://www.sai.msu.su/~megera/postgres/gist/doc/intro.shtml#_blank) (GeneralizedSearchTree) es un sistema avanzado que tare una gran variedad de algoritmos de ordenamiento y búsqueda incluyendo árbol-B, árbol B+, árbol R, árbol de sumas parciales, arboles B+ rankeados y muchos otros. También provee una interfaz que permite la creación de tipos de datos customizados, así como también métodos de búsqueda extensibles. Es por ello GiST ofrece la flexibilidad de especificar que guardar, como guardarlo y la habilidad de definir nuevas formas para buscar.

Otras características avanzadas son herencia de tablas, sistema de reglas, eventos de bases de datos. La herencia de tablas pone un enfoque orientado a objetos en la creación de tablas, permitiendo a los diseñadores de bases de datos a derivar nuevas tablas de otras existentes, tratándolas como clases base. Más aun, PostgreSQL soporta herencia simple y múltiple de esta manera.

El sistema de reglas, también llamado el sistema de reescritura de consultas, permite al diseñador de bases de datos crear reglas que permitan identificar las operaciones concretas en una misma tabla o vista, y dinámicamente transformarlas en operaciones alternativas cuando se procesan.

El sistema de eventos es un sistema de comunicación entre procesos en el que los mensajes y eventos se puede transmitir entre los clientes utilizando los comandos LISTEN y NOTIFY, permitiendo una comunicación peer-to-peer y coordinación avanzada sobre eventos de bases de datos. Dado que las notificaciones se pueden emitir a partir de disparadores y SP, los clientes de PostgreSQL pueden monitorear los eventos de base de datos como pueden ser updates de la tabla, inserts o deletes a medida que ocurren.

Altamente customizable

PostgreSQL ejecuta storedprocedures en más de una docena de lenguajes de programación, como Java, Perl, Python, Ruby, Tcl, C/C++, y su propio PL/pgSQL, que es similar a la de Oracle PL/SQL. Dentro de su biblioteca de funciones estándar hay cientos de funciones integradas que van desde las matemáticas básicas y las operaciones sobre strings, a la criptografía y la compatibilidad con Oracle. Triggers y storedprocedures pueden ser escritos en C y se cargan en la base de datos como una biblioteca, lo que permite una gran flexibilidad en la ampliación de sus capacidades. Del mismo modo, PostgreSQL incluye un framework que permite a los desarrolladores definir y crear sus propios tipos personalizados de datos junto con funciones y operadores que definen su comportamiento. Como resultado, una gran cantidad de tipos de datos avanzados se han creado, que van desde direcciones de red hasta incluso los tipos de datos de ISBN / ISSN (International Standard Book Number/International Standard Serial Number), los cuales pueden ser opcionalmente añadidos al sistema.

Así como hay muchos lenguajes procedurals soportados por PostgreSQL, hay muchas bibliotecas que sirven de interfaz, permitiendo a varios lenguajes ser compilados e interpretados para interactuar con PostgreSQL. Hay interfaces para Java (JDBC), ODBC, Perl, Python, Ruby, C, C++, PHP, Lisp, Scheme, y Qt solo por nombrar algunos.

El código de PostgreSQL se encuentra disponible bajo una licencia de código: ‘[PostgreSQL License](http://www.opensource.org/licenses/postgresql)’. Esta licencia brinda la libertad de usar, modificar y distribuir PostgreSQL en cualquier forma, con código abierto o cerrado.

## Middleware de base de datos

**Pgpool I-II 2.3.3**

**pgpool-II** es un middleware que funciona entre servidores PostgreSQL y un cliente de base de datos PostgreSQL. Ofrece las siguientes características.

**Agrupación de conexiones:** pgpool-II guarda conexiones existentes con los servidores PostgreSQL, permitiendo volver a usarlas cuando se necesita una nueva conexión con las mismas propiedades (es decir, nombre de usuario, bases de datos, versión del protocolo). Reduce overhead en la conexión y mejora el rendimiento global del sistema.

**Replicación:** pgpool-II puede administrar varios servidores PostgreSQL. Con el uso de la función de replicación se pueden crear backups en tiempo real en dos o más discos físicos, para que el servicio pueda continuar sin detención de servidores, en caso de un fallo de disco.

**Load Balance:** Si una base de datos se replica, la ejecución de una consulta SELECT en cualquier servidor devolverá el mismo resultado. pgpool-II utiliza la función de replicación para reducir la carga sobre cada servidor PostgreSQL mediante la distribución de las consultas SELECT entre varios servidores, mejorando el rendimiento global del sistema. El rendimiento mejora en proporción al número de servidores PostgreSQL. El balanceo de carga funciona mejor en una situación donde hay una gran cantidad de usuarios ejecutando muchas consultas al mismo tiempo.

**Soporte a ConexionesExtra**: Existe un límite en el número máximo de conexiones simultáneas con PostgreSQL, y las conexiones son rechazadas después de alcanzar este número. pgpool-II también tiene un límite en el número máximo de conexiones, pero las conexiones adicionales se pondrán en cola en lugar de devolver un error inmediatamente.

**Consultas Paralelas:** Utilizando la función de consultas en paralelo, los datos pueden ser divididos entre los múltiples servidores, de modo que una consulta se puede ejecutar en todos los servidores al mismo tiempo para reducir el tiempo de ejecución global. Se obtienen mejores resultados cuando se trabaja sobre grandes volumenes de datos.

Arquitectura del sistema con pgpool-II

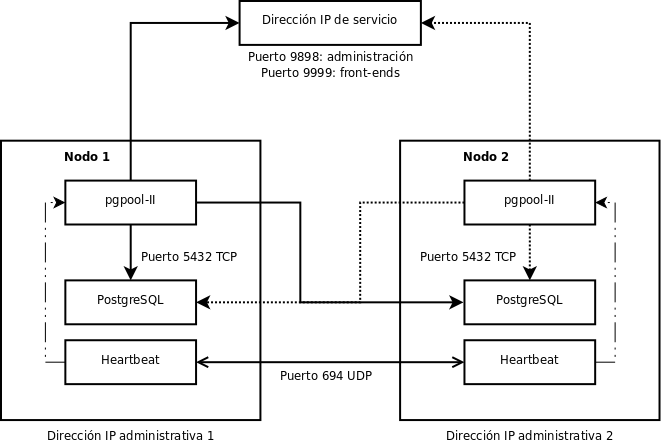
En el clúster se persiguen dos objetivos: uno, la alta disponibilidad, y dos, el rendimiento. La funcionalidad que persigue el clúster, en este punto, es únicamente la de servidor de base de datos, pero lo hace a través de tres aplicaciones:

PostgreSQL, el sistema gestor de bases de datos (S.G.B.D.)

pgpool-II, el middleware que gestiona la alta disponibilidad de los servidores de PostgreSQL.

Heartbeat, un software que usaremos para dar alta disponibilidad a pgpool-II y a la dirección IP de servicio.

Esta configuración nos permite obtener alta disponibilidad de todos los servicios y recursos en las dos máquinas destinadas a este clúster. El diagrama de la arquitectura resultante sería el siguiente:



## Sistema operativo

**Debian GNU/Linux 5.0 para servidores**

Libre: Al tener la licencia GPL es software libre lo cual asegura transparencia.

Calidad: Debian es reconocido por tener altos estándares de calidad en cuanto a la estabilidad. Se lo considera uno de los sistemas operativos más seguros y estables.

Mantenimiento: El uso de un administrador de paquetes permite que el mantenimiento sea sencillo y versátil.

Ciclo de release: La versión 5 es actualmente la versión estable, lo que asegura, según las políticas de Debian, que cualquier actualización será de estabilidad, rendimiento o seguridad y nunca se agregarán nuevas funcionalidades que puedan romper con la compatibilidad hacia atrás.

**Comparación de los sistemas operativos comerciales vs Debian GNU / Linux**

La flexibilidad, la versatilidad.  
El comportamiento de un sistema GNU / Linux se define tradicionalmente en un conjunto bien documentado de archivos de texto de configuración claros que permiten un alto grado de personalización. En caso de cualquier requisito de personalización que va más allá de las amplias posibilidades de los archivos de configuración, programas individuales o incluso todo el sistema puede ser re-compilado con el código fuente modificado a cualquier medida. La flexibilidad del código abierto GNU / Linux va incluso tan lejos que no sólo puede hacer que se ejecute  
en el i386 CPU y sucesores, (Pentium, Athlon), sino también en 15 arquitecturas absolutamente diferentes, tales como Motorola 68K, PowerPC (Mac) y SPARC (Estaciones de trabajo Sun).

Estabilidad, fiabilidad.  
Un sistema operativo utiliza profesionalmente se espera que funcione bien, siempre y cuando el usuario / administrador desea que se ejecute. Los sistemas de servidor se ejecutan normalmente 24x7, y los sistemas GNU / Linux de servidor a menudo se mantienen activos varios cientos de días, sino incluso varios años. Un servidor GNU / Linux normalmente cumple su asignaciones hasta que una pieza de hardware falla o el sistema es apagado / reiniciado por el administrador.

Seguridad.  
Unix era un sistema operativo multiusuario, por lo que es GNU / Linux desde que se inventó. Los sistemas operativos de Microsoft se han diseñado en tiempos de los ordenadores personales, originalmente carente de cualquier tipo de restricciones de acceso. Ésta es una de varias razones por los usuarios de Windows son molestadas por los virus y otros programas maliciosos en órdenes de magnitud más probable que Usuarios de GNU / Linux.

Compatibilidad, conectividad.  
GNU y Linux se inventaron sobre Internet, basados en la cooperación entre los voluntarios  
los desarrolladores se comunican entre sí por correo electrónico. Las primeras características del kernel de Linux y de la suite de software GNU es su conectividad. Por ahora, esta conectividad ha llegado a un incomparable nivel de madurez y calidad. Cualquier otro sistema operativo moderno sólo ha tratado de adaptar un similar nivel de interoperabilidad, pero desde ya incomparable. El software libre y código abierto se basa en protocolos abiertos en lugar de formatos propietarios cerrados.

Eficiencia.  
El núcleo de Linux se puede compilar en tan poca capacidad como unos pocos cientos de kilobytes.  Linux funciona en cualquier sistema de la manera más eficiente, a elección del usuario. En los sistemas de servidor no hay necesidad de una interfaz gráfica de usuario.  
Los servidores GNU / Linux pueden ser mantenidos y totalmente controlados por una consola simple, pero potente, que se puede acceder por métodos seguros (ssh) en todo el Internet, sin inconvenientes, por lo que su eficiencia sigue siendo la elección del usuario. No hay necesidad de mejorar el hardware con cada nueva versión.  
Una vez que el sistema funciona bien, funciona bien mientras dura el hardware.

Rendimiento.  
El eficiente funcionamiento de sistemas Linux a menudo han sido punto de referencia y comparación con productos de servidor Microsoft. En los sistemas de servidor de MS, que cuestan mucho dinero sólo para poder ejecutarlos, tienen muy pocas ventajas sobre los sistemas estándar de escritorio de Microsoft. Están diseñados para ser los productos más eficaces de Microsoft, tales como servidores de archivos o servidores de bases de datos u otras aplicaciones.

Mantenimiento.  
Como se dijo en el ítem de estabilidad, el sistema simplemente se ejecuta. Una vez que está corriendo y haciendo su trabajo, usted puede esperar a continuar haciendo eso durante años, a menos que su hardware falle. Un sistema Debian puede actualizar fácilmente su software a través de Internet todas las noches. Se puede pensar que esto podría no ser suficiente. Debian GNU / Linux da las decisiones en efectivo al propietario.

Instalación.  
Ahora bien, esta es la parte mala de él. Cualquier usuario con experiencia puede instalar Windows, y lo ha hecho, probablemente  
ya. La instalación de Debian GNU / Linux y la conversión de la toma a alguien que sabe  
su camino. Windows tiende a requerir la reinstalación después de un tiempo determinado tan pronto como las cosas no funcionan  
correctamente más. Este no es el caso de los sistemas Debian. Una vez instalado, simplemente correr.

Costo.  
Debian GNU / Linux es gratuito, pero se debe tener en cuenta su instalación y configuración. Por lo demás no presenta costos adicionales. Mientras que una licencia de Microsoft Windows Server Std 2008 32bit/x64 tiene un costo aproximado de 4400 pesos.

## Servidor de aplicación

**Apache Tomcat 6.0**

El servidor de aplicación será Apache Tomcat 6.0. Este contenedor es el más utilizado para aplicaciones web desarrolladas en Java y puede correr en la mayoría de los sistemas operativos sin dificultades de configuración. No tiene costos de licencias.

Tomcat 6.x

Implementado de Servlet 2.5 y JSP 2.1

Soporte para *Unified Expression Language* 2.1

Diseñado para funcionar en [Java SE 5.0](http://es.wikipedia.org/wiki/Java_SE) y posteriores

Soporte para [Comet](http://es.wikipedia.org/wiki/Comet) a través de la [interfaz](http://es.wikipedia.org/wiki/Interfaz_de_usuario) CometProcessor

# Volumen de datos estimado para el sistema

|  |  |
| --- | --- |
| **Estimación de volumen** | |
| **Tamaño medio de fila** | 50 Bytes |
| **Cantidad de tablas** | 35 |
| **Consultas diarias aproximadas** | 2000 |
| **Instalación PostgreSQL** | 200 MB |
| **Crecimiento anual Máx. estimado de la DB** | 1750 MB |
| **Tamaño de archivos de log Aproximado** | 300 MB anuales |
| **Tamaño SP y Configuración estimado** | 10MB |
| **Coeficiente de Seguridad** | 30.00% |
| **Total Anual por instancia** | 29380 MB ≈**30 GB** |

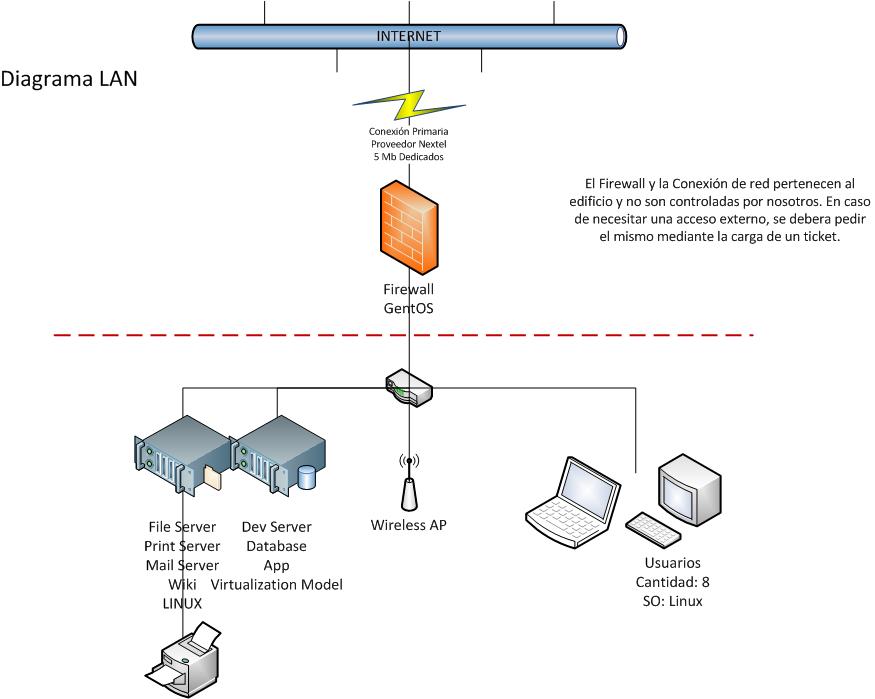
Este valor es válido para el primer año del funcionamiento del sistema. Si se estima un crecimiento anual del 20% y tiempo de vida del sistema de 5 años. Consideramos el siguiente cálculo:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Año** | **1** | **2** | **3** | **4** | **5** | **Total** |
| **Factor** | 1 | 1,2 | 1,44 | 1,728 | 2,0736 | 7,4416 |
| **GB** | 30 | 36 | 43,2 | 51,84 | 62,208 | 233,248 |

El tamaño necesario para albergar la base de datos no debería ser mayor a los 250GB, cualquier configuración de disco no menor a esta capacidad será suficiente para estos requerimientos.

# Infraestructura desarrollo interna

## Esquema de la infraestructura



## Herramientas para desarrollo

### Configuración de servidores

Se dispondrá de 2 servidores

Servidor 1

**Roles:** Mail, Dhcp, Dns, Ldap, File Server, Print Server, Wiki, BTS

**Email**: Se instalará un servidor Postfix con soporte de protocolos SMTP e IMAP,

**Networking**:

* + DNS: Bind9 o similar.
  + DHCP: Proporcionado por el SO.
  + LDAP: OpenLDAP

**SCM**: Se necesita un Source control manager, se eligió git. Esta herramienta de versionado es la más avanzada de su clase. Es distribuido, cada desarrollador tiene su propia copia del código, y es mucho más rápido que otros. Las operaciones de branch y merge difíciles en otros son triviales en git.

**Documentación**: Se proporcionará al equipo una herramienta para el intercambio de información rápida e informal, Twiki.

**BTS**: El Bug tracking system seleccionado es Mantis.

**Maven**: Se instalará un repositorio local de JakartaMaven, para proveer paquetes de librerías y los paquetes propios de la aplicación.

Servidor 2

**Roles:** Entorno de Desarrollo mediante virtualización con xen.

Se configurarán varias máquinas virtuales para crear ambientes de desarrollo y testing. Gracias a la virtualización es fácil la creación de nuevas máquinas virtuales para configurar entornos rápidamente.

Se comenzará con una configuración de tres servidores virtuales.

**Ambiente de Desarrollo**: Se mantendrá un ambiente de aplicaciones similar al productivo pero que estará a disposición de los desarrolladores para probar el sistema.

**Ambiente de QA**: Similar al de desarrollo pero se utilizará para el testing integral de la aplicación y deberá mantenerse estable, las versiones que pasen los test en este ambiente se consideraran lo suficientemente estable para ser instalados en producción.

**Base de datos**: Se instalará la misma base de datos seleccionada para el sistema. Se utilizará para desarrollo y testing.

Todas estas herramientas son de licencia libre y gratuitas, pero como contrapartida, debe planificarse su configuración.

### Compra de servidores para desarrollo

Servidor 1

En el primer servidor corren las herramientas para el desarrollo y se almacena información importante. No es necesario una performance muy alta pero sí una alta disponibilidad. Este equipo contará con fuente redundante y 3 discos configurados en RAID 5.

Se seleccionó un servidor Dell PowerEdge 2970 con las siguientes características:

|  |  |  |
| --- | --- | --- |
| Cantidad | **Componente** | **Descripción** |
| 1 | Procesador | Quad Core AMD Opteron™ 2378,4x512K Cache,2.4GHz,1Ghz HyperTransport |
| 2 | Memoria | 2GB 800MHz, total 4GB |
| 1 | Controladora RAID | PERC 6/I, x8 Backplane |
| 3 | Discos rígidos | 250GB 7.2K RPM SATA HotPlug RAID 5 |
| 2 | Fuente | Redundante |
| 3 | Años de garantía | ProSupport para IT, con servicio telefónico 24/7 y con respuesta al día siguiente laborable de un técnico en sitio |
|  | Precio | **$15,496.00** |

Link: <http://tinyurl.com/adr-server-dev1>

Servidor 2

Este será un servidor de máquinas virtuales por lo que lo más importante es la performance. Las imágenes de las máquinas virtuales se guardarán periódicamente y no hay datos importantes por lo que la alta disponibilidad no es tan importante. Se configuró el mismo modelo que el servidor anterior pero con un procesador más y el doble de memoria. Se remplazó el RAID5 por un RAID0 de dos discos y no se utilizó fuente redundante.

|  |  |  |
| --- | --- | --- |
| Cantidad | **Componente** | **Descripción** |
| 2 | Procesador | Quad Core AMD Opteron™ 2378,4x512K Cache,2.4GHz,1Ghz HyperTransport, total 8 cores. |
| 8 | Memoria | 1GB 800MHz, total 8GB |
| 1 | Controladora RAID | PERC 6/I, x8 Backplane |
| 2 | Discos rígidos | 160GB 7.2K RPM SATA HotPlug RAID 0 |
| 2 | Fuente | No Redundante |
| 3 | Años de garantía | ProSupport para IT, con servicio telefónico 24/7 y con respuesta al día siguiente laborable de un técnico en sitio |
|  | Precio | **$16,323.00** |

Link: <http://tinyurl.com/adr-server-dev2>

## Estaciones de trabajo

**Requisitos Mínimos de los Equipos para Desarrollo**

Para poder trabajar de forma cómoda con las herramientas modernas los equipos usados para el desarrollo deberán poseer al menos 3GB de memoria RAM y un procesador de dos núcleos.

Es importante tener una buena respuesta del soporte técnico en caso de falla de un equipo, lo tiempos son ajustados y el remplazo de un equipo no puede retrasar el cronograma.

Los equipos deben ser compatibles con Sistemas Operativos Linux.

El precio del equipo no es una imposibilidad para la adquisición del mismo, sin embargo la empresa se puso como limite el gasto de 1500 dólares para el equipo completo (CPU y Monitor).

El display deberá ser LCD y de al menos 17”.

### Opciones de Compra Para el Equipo de Desarrollo

|  |
| --- |
| Opción Dell: Vostro 230 |
| PROCESADOR: Intel® Core™2 Duo E7500 (3MB Caché, 2.93GHz, 1066MHz FSB) |
| MEMORIA: Memoria de 4GB Dual Channel DDR3 SDRAM 1333MHz - 2DIMMs |
| TARJETA DE VIDEO: 512MB NVIDIA® GeForce® G310 (DVI + VGA + HDMI) |
| DISCO DURO: Disco Duro de 500GB Serial ATA (7200RPM) con DataBurst Caché™ |
| MONITOR: Monitor Dell E1910H - Ancho y Plano de 18.5 pulgadas |
| <http://tinyurl.com/adr-pc-dev1> |
| Costo: 1250 Dólares |

|  |
| --- |
| **Opción Clon: AMD Phenom 2** |
| MOTHERBOARD: MB GIGABYTE GA-MA785GM-US2H |
| PROCESADOR: MICRO AMD PHENOM 2 X4 945 |
| MEMORIA: 2048/800 D/B SUPERTALENT x2 |
| DISCO DURO: WD 500GB SERIAL ATA |
| MONITOR: Lcd LG 17” |
| PROVEEDOR: Fag Sistems <http://www.fagsistems.com.ar/home.php3> |
| **Costo: 970 dólares** |

### Tabla ponderación de equipos Terminales Para desarrollo

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Puntos** | **SubPuntos** | **Dell Vostro** | **Clon** |
| **Precio de compra** | 10 |  | 6 | 8 |
| **Software compatible** | 30 |  | 9 | 9 |
| **Soporte del vendedor** | 30 |  | 10 | 6 |
| **Hardware** | 30 |  |  |  |
| **Memoria** |  | 15 | 9 | 8 |
| **Procesador** |  | 15 | 9 | 7 |
| **Total** |  |  | 43 | **38** |

A partir de la tabla de ponderación, elegimos el Equipo de Dell como compra para el desarrollo del sistema

### Selección de Laptop para líder de proyecto

Se requiere un equipo de buen rendimiento pero a la vez robusto y portátil. Balanceando estos requerimientos se decidió la siguiente configuración.

|  |
| --- |
| Dell Latitude E4310 |
| PROCESADOR: Intel® Core™ i5-520M (2.40GHz, 3M cache) |
| MEMORIA: Memoria de 4.0GB, DDR3-1066 SDRAM, 2 DIMMS |
| TARJETA DE VIDEO: Intel® HD Graphicswithdynamicfrequency |
| DISCO DURO: 250GB Hard Drive, 7200RPM |
| MONITOR: 13.3" WXGA, Slate Silver |
| GARANTÍA: 3 años de ProSupport |
| <http://tinyurl.com/adr-pc-dev2> |
| Costo: $7,517.00 |

## Sistema Operativo Para las estaciones de trabajo

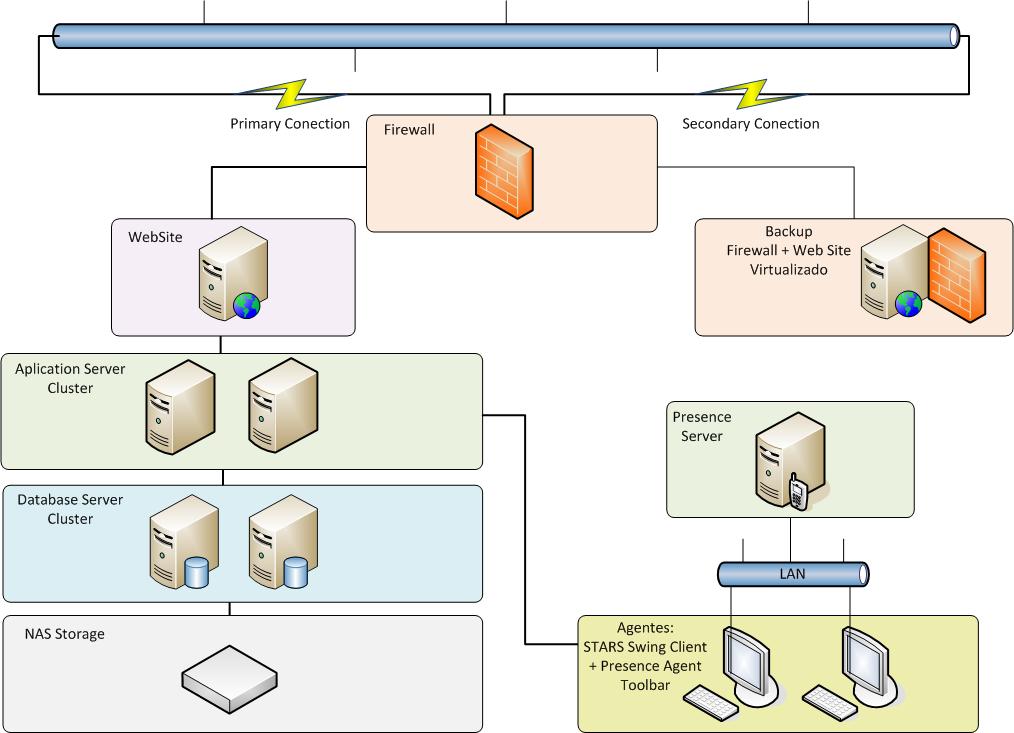
**Ubuntu 10.4 LTS recomendado**

Es un derivado de Debian por lo que provee similares características pero se desarrolla con mayor énfasis en la facilidad de uso por personas con poca experiencia en sistemas Linux. La interfaz gráfica es intuitiva y muy similar a la de MS Windows, lo que asegura que el tiempo de entrenamiento por los operarios será mínimo.

# Infraestructura Producción



## Esquema de la infraestructura



## Disponibilidad y Rendimiento

Esta infraestructura mediante clusterización permite implementar soluciones de alta disponibilidad y alto rendimiento, ya que no sólo posibilita que el tráfico y ejecución se distribuyan entre varias máquinas, sino también que permite aumentar la tolerancia ante fallos, ya que la caída de una de las máquinas no afecta el funcionamiento global del sistema, logrando ser imperceptible para el usuario.

## Esquema de Disponibilidad 7 x 24

**Doble enlace de internet**: Como primer medida, el sistema presenta un doble enlace de conexión a internet, con lo cual ante la eventual caída de uno de ellos, el siguiente se activa automáticamente, redirigiendo el trafico por la nueva salida a través del FW.

Cluster a Nivel Aplication Server: compuesto por dos nodos, para brindar alta disponibilidad de la aplicación. En caso de caído de uno de los nodos, el soft instalado, automáticamente notara la desconexión, y automatizara su proceso para seguir productivo sin necesidad de mantenimiento manual de un operario.

Cluster a Nivel Database Server: al igual que el cluster de aplicación, estará compuesto por dos nodos que brindaran una alta disponibilidad. El proceso de caída es similar al anterior.

Nivel de Raid 5 en el Storage: Este array ofrece tolerancia a fallos, pero además, optimiza la capacidad del sistema permitiendo una utilización de hasta el 80% de la capacidad del conjunto de discos. Esto lo consigue mediante el cálculo de información de paridad y su almacenamiento alternativo por bloques en todos los discos del conjunto. La información del usuario se graba por bloques y de forma alternativa en todos ellos. De esta manera, si cualquiera de las unidades de disco falla, se puede recuperar la información en tiempo real, sobre la marcha, mediante una simple operación de lógica de O exclusivo, sin que el servidor deje de funcionar. (Ver ítem 4.8)

## Configuración de servidores para producción

En el lugar físico del cliente se configurarán cuatro servidores para ejecutar el ambiente de producción.

**Firewall y proxy (1)**: Para mantener seguro el ambiente e incrementar la integridad se configurarán estas aplicaciones en un servidor a través del cual se obtendrá acceso al sistema. Debian Server.

**Web Server (1)**: La aplicación web que presta servicios a los ciudadanos se configurará en un servidor dedicado. No se espera que la demanda de esta aplicación sea muy intensa ni tampoco es crucial un alto tiempo de respuesta por lo que se eligió un servidor pequeño.

**Application server (2)**: En estos servidores se instalará la aplicación backend principal que proveerá servicios a los clientes desktop de los operarios y a la aplicación web. Se espera que la respuesta de este servicio sea rápida por lo que se eligió un servidor con buenas prestaciones.

**Database server (2)**: En estos servidores se instalará la base de datos a la cual el backend accederá al igual que el backend este servidor debe tener buena respuesta y alta disponibilidad.

**Backup (1):** Firewall + Web Server. Debian Server + Xen

## Software Necesario

**Heartbeat:** es un paquete de software creado por LINUX-HA, que corre scripts al ocurrir un evento de inicio o apagado de una pc del cluster. Funciona Utilizando gratuitous ARPs. Es extensible a configuraciones de varios servidores

**Pacemkaer:** versión 2.0.0. Se utilizara como clusterresource manager, anexado al Hertbeatpar el control del inicio y cierre de los servicios.

**Xen:** Como monitor de maquinas virtuales. Versión 4.0.1. Xen utiliza una técnica llamada [paravirtualización](http://es.wikipedia.org/wiki/Paravirtualización) para alcanzar alto rendimiento.

Las máquinas virtuales Xen pueden ser migradas en caliente entre equipos físicos sin pararlos. Durante este proceso, la memoria de la máquina virtual es copiada iterativamente al destino sin detener su ejecución.

## Elección de servidores para producción

Se eligieron dos modelos de servidores, uno para las aplicaciones menos demandantes y otro para las más demandantes, una de las cuales tiene el requerimiento especial de mayor capacidad de almacenamiento.

### Servidor Tipo A, cantidad: 3

Para el servidor 1 (Firewall, proxy) y para el servidor 2 (web server).

**Dell PowerEdge R210**

|  |  |  |
| --- | --- | --- |
| **Cantidad** | **Componente** | **Descripción** |
| **1** | Procesador | Intel® Xeon® X3440, 2.53 GHz, 8M Cache, 4 cores |
| **2** | Memoria | 2GB 1333MHz, total 4GB |
| **1** | Discos rígidos | 160GB 7.2k RPM SATA 3.5 |
| **5** | Años de garantía | Limitada en el sitio con respuesta al siguiente día laborable. |
|  | Precio | **$5,431.00** |

Link: http://tinyurl.com/adr-server-a

### Servidor Tipo B

Para el servidor 3 (backend)

**Dell PowerEdge R310**

|  |  |  |
| --- | --- | --- |
| **Cantidad** | **Componente** | **Descripción** |
| **1** | Procesador | Intel® Xeon® X3440, 2.53 GHz, 8M Cache, 4 cores |
| **2** | Memoria | 2GB 1333MHz, Dual Ranked UDIMM, total 4GB |
| **2** | Discos rígidos | 160GB 7.2k RPM SATA 3.5 en RAID 1 |
| **2** | Fuente | 400W Redundante |
| **5** | Años de garantía | Limitada en el sitio con respuesta al siguiente día laborable |
|  | Precio | **$8.159** |

Link: <http://tinyurl.com/adr-server-b>

## Storage Nas

Para el storage, se eligió un NAS Iomega de 2 TB (4 discos de 500 Gb), conectado directamente a la red lan tcp/ip.

NAS nos proveerá un bajo costo de balance de carga, tolerancia a fallos y servidor web para proveer servicios de almacenamiento.

Este nos permitirá implementar un nivel de raid 5 dentro del mismo para el resguardo de la información.

Ofrece la Posibilidad de realizar “Hot Swap”, o sea, cambiar los discos “en caliente”, con el servidor prendido y funcionando, evitando la necesidad de parar el servicio en caso de falla de harware o corrupción de datos.

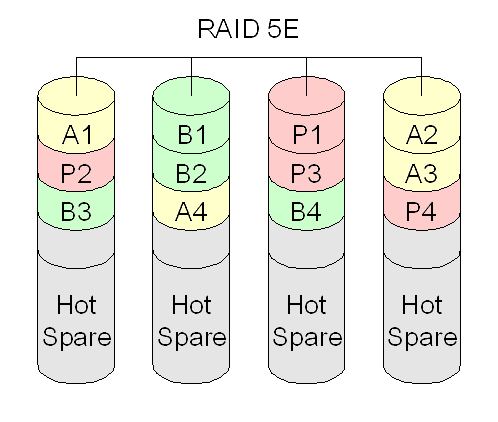


Link: <http://tinyurl.com/adr-nas>

Precio con 4 discos de 500GB: 1000 dólares

## Nivel Raid

Tendrá como estructura interna un RAID 5E (Raid 5 más un disco de espera (*hotspare*)), conformado por 4 discos Internos.



Este esquema fue elegido gracias a que permite un gran balance entre costo, velocidad, resguardo de la información y disminución del cuello de botella al repartir la información de paridad entre los diferentes discos, además posee un alto nivel de disponibilidad gracias al disco de espera extra, ya que si un disco falla el conjunto se reconstruye sobre el disco de reserva. Esto último es vital en un sistema crítico como este, el cual no puede dejar de funcionar, ya que de lo contrario se perjudicaría a cientos de ciudadanos.

## Integración Con Sistema de Comunicaciones

La integración será a nivel del cliente usando el control (no visual) Presence CRM Optimizer ActiveX (PresenceInterfaceX.ocx) y PresenceAgent.

Se utilizará un Bridge open source( “ComfyJ” por ejemplo ) para poder utilizar los controles Activex desde la aplicación STARS (Java).

Con está integración el tratamineto de la llamada será realizado a través del PresenceAgentToolbar. El sistema STARS recibirá el evento de la llamada entrante a través de la API propuesta por Presence.

Ventajas de esta integración:

Mínimo impacto en la adaptación de la aplicación STARS para su integración con Presence CRM Optimizer.

No es necesaria la integración CTI en la aplicación STARS.

Únicamente se deben programar los eventos del ActiveX necesarios para realizar el screenpopup.

Permite realizar una integración rápida y sencilla.

Permite mejor detección y corrección de problemas. Al estar definida la integración en un punto muy concreto, es fácil aislar y detectar el origen de posibles fallos de integración o ejecución.

Disponibilidad de la funcionalidad estándar incorporada en la barra de PresenceAgent, como puede ser: consulta de histórico de llamada del cliente, softphone, agendas de teléfonos, comunicados internos, etc.

Existe la posibilidad de crear botones en la aplicación corporativa o de negocio que ejecuten determinadas funciones de la barra de PresenceAgent usando métodos del control ActiveX (por ejemplo colgar, transferir, retener, etc.).

# Oficinas para el desarrollo



## Puestos

Costo por puesto: 1200$ + IVA Mensual

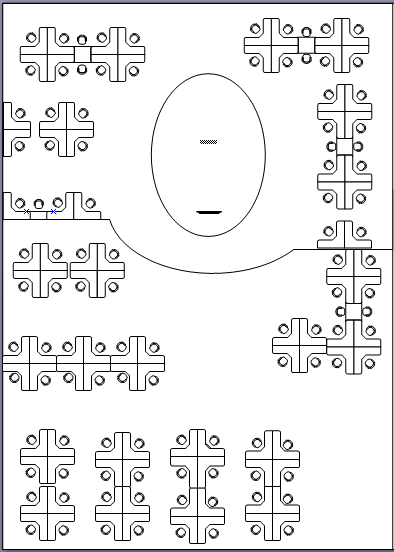
El costo incluye:

* 1 puesto de red. La conexión es dedicada de 5MB, para un total de 150 personas.
* 1 Puesto de Telefonía
* 1 Silla
* Acceso a los Servicios Básicos del edifico: baños, comedor, lugares comunes
* Sala Oval para reuniones empresariales. Con conexión de red, telefonía, espacio para 14 personas sentadas.

Dirección: Andrés Arguibel 2860, Las Cañitas

Disponibilidad: Inmediata

## 5.2 Plano de la Planta alquilada



**Beneficios de la Oficina:**

Al presentar esta disposición, favorece el trabajo de las empresas de sistemas o afines, permitiendo una gran interacción y comunicación entre los integrantes del equipo.

Los puestos no tienen división, ni son “Boxes” cerrados.

## Hosting de Servers

El Hosting de los servidores para Desarrollo será de 500$ mensual cada servidor de 4 unidades.

Alojadas en el Datacenter de Planta Baja de la misma oficina en donde se aloja la empresa físicamente.

Presenta especiales características para nuestro proyecto ya que nos permitirá realizar la instalación y configuración de los mismos sobre la marcha. No es necesario tener todo configurado para enviar los servidores al lugar donde se alojaran.

Por otro lado, nos permite una mayor rapidez de acción en caso de surgir algún tipo de inconveniente.

# Empleados

Para el total del proyecto, contaremos con 8 empleados, con los siguientes roles y cantidades:

|  |  |  |
| --- | --- | --- |
| **Rol** | **Cantidad** | **Sueldo Neto** |
| **Arquitecto / Líder** | 1 | $ 8000 |
| **IT** | 1 | $ 4000 |
| **Programador Senior** | 2 | $ 5500 |
| **Programador SemiSenior** | 1 | $ 4000 |
| **Programador Junior** | 2 | $ 3000 |
| **Tester** | 1 | $ 3000 |



## Roles particulares

Arquitecto / Líder: Sera el encargado como primer medida, de crear y configurar el ambiente para que los desarrolladores puedan realizar sus tareas. Crear/customizar frameworks necesarios para el desarrollo. Una vez que esto este finalizado, su tarea será la de líder de proyecto. Referente técnico para los desarrolladores, coordinar tareas de los mimos. Seguimiento del desarrollo día a día.

IT: Sera el encargado de instalar, configurar y mantener el ambiente de desarrollo (Repositorio, Wiki, Issue Tracker) y de instalar y configurar el ambiente de producción.

Desarrollador: Programar los casos de uso que se le asignen, dependerá del grado de seniority. Aplicar buenas prácticas de programación. Realizar Unit Testing. Llevar un control de los tickets que se asignen en el Issue Tracker.

* Programador Junior: Sera el encargado principalmente de proveer los ABM de la aplicación y aquellas partes con bajo nivel de dificultad de programación
* Programador SemiSenior: Se le asignarán principalmente las interfaces de usuario. Deberá ser el primer eslabón de consulta y ayuda para los niveles junior de seniority.
* Programador SeniOR: Estará abocado a la integración con el sistema STARS, módulos de administración y todas aquellas tareas las cuales por complejidad, excedan a los dos primeros niveles.

Tester: Desarrollar los test cases para el software. Llevar a cabo las pruebas. Crear ticktes por defectos, mejoras, etc. Utilizando el sistema de issue tracker.

# Costos / Capital Inicial

Se pedirá un Crédito de banco Santander Rio para solventar los costos de start-up.

El monto del crédito será de $150.000, con 1 mes de gracia en el pago

Características del Mismo:

* Monto máximo hasta $150.000
* Pre-aprobación dentro de las 24hs.
* La primer cuota la puede pagar hasta en 90 días.
* En caso de cancelación anticipada total/parcial se cobrará una comisión del 2% + IVA sobre  
  el monto a cancelar.

|  |  |
| --- | --- |
| Capital solicitado: | **$140.000,00** |
| Gastos de Otorgamiento: | $5.600,00 |
| IVA sobre gastos de otorgamiento: | $1.176,00 |
| Neto a acreditar: | $133.224,00 |
| Cuota inicial: | $30.846,46 |
| Cuota promedio total: | $30.511,83 |
| Plazo: | 5 meses |
| Tasa Nominal Anual: | 28,00% |
| Tasa Efectiva Anual: | 31,89% |
| Costo Financiero Total: | 75,12% |
| Tipo de tasa: | Fija |
| Moneda: | Pesos |
| Sistema de amortización: | Sistema Francés |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Cuota N° | Interés del Periodo | Capital Amortizado | Cuota Pura | Cob. Vida | Gastos Admin | IVA | Cuota Final | Saldo Total |
| 1 | $3266,67 | $26723,46 | $29990,13 | $140,77 | $0 | $715,56 | $30,846,46 | $113,276,54 |
| 2 | $2643,12 | $27,347,01 | $29990,13 | $113,90 | $0 | $578,97 | $30683,00 | $85,929,53 |
| 3 | $2005,02 | $27,985,11 | $29990,13 | $86,40 | $0 | $439,20 | $30515,73 | $57944,42 |
| 4 | $1352,04 | $26,638,09 | $29990,13 | $58,26 | $0 | $296,16 | $30344,55 | $29306,33 |
| 5 | $683,81 | $29306,32 | $29990,13 | $29,47 | $0 | $149,79 | $30169,39 | $0,01 |

Los Costos de start-up estarán dados por los siguientes ítems:

* Las oficinas son alquiladas
* Los servidores del cliente se compraran a partir del 4 mes de proyecto
* Los servidores propios
* Las computadoras para el equipo de desarrollo
* Otros Temas Hardware

# Costos Mensuales



## Por Mes

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Período |  | Proyecto | | | | | |
| Concepto | **Start-up** | **1** | **2** | **3** | **4** | **5** | **6** |
| Recursos Humanos |  | $ 37.408 | $ 71.698 | $ 77.932 | $ 77.932 | $ 54.033 | $ 54.033 |
| Equipamiento | $ 79.200 | $ - | $ 51.000 | $ - | $ - | $ - | $ - |
| Hosting Servidores |  | $ 1.000 | $ 1.000 | $ 1.000 | $ 1.000 | $ 1.000 | $ 1.000 |
| Alquiler de Oficina | $ 0 | $ 12.000 | $ 12.000 | $ 12.000 | $ 12.000 | $ 12.000 | $ 12.000 |
| Papelería y útiles |  | $ 500 | $ 500 | $ 500 | $ 500 | $ 500 | $ 500 |
| Teléfono y Celulares |  | $ 500 | $ 500 | $ 500 | $ 500 | $ 500 | $ 500 |
| Movilidad y Viáticos |  | $ 500 | $ 500 | $ 500 | $ 500 | $ 500 | $ 500 |
| Seguros (1% de Activos) |  | $ 792 | $ 1.302 | $ 1.302 | $ 1.302 | $ 1.302 | $ 1.302 |
| Pago Interés Préstamo |  | $ - | $ 30.846 | $ 30.683 | $ 30.515 | $ 30.344 | $ 30.169 |
| Total de Egresos | **$ 79.200** | **$ 52.700** | **$ 169.346** | **$ 124.417** | **$ 124.249** | **$ 100.179** | **$ 100.004** |

## Totales

|  |  |
| --- | --- |
|  | Total del proyecto |
| Recursos Humanos | $ 373.037 |
| Equipamiento | $ 130.200 |
| Hosting Servidores | $ 6.000 |
| Alquiler de Oficina | $ 72.000 |
| Papelería y útiles | $ 3.000 |
| Teléfono y Celulares | $ 3.000 |
| Movilidad y Viáticos | $ 3.000 |
| Seguros (1% de Activos) | $ 7.302 |
| Pago Interés Préstamo | $ 152.557 |
| Total de Egresos | **$ 750.096** |

# Entregas del proyecto

El proyecto se realizara en 3 entregas parciales y una final

El Proyecto Llevara un Total de 6 Meses y Medio, dividiéndose Principalmente en las siguientes etapas.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Comienzo | Termina de configurar ambiente Interno y Arquitectura | Primera Entrega | Segunda Entrega | Tercera Entrega | Entrega Final, Puesta en Marcha | Capacitación |

Midiéndolo en días, si el proyecto comenzara el 6/5, se haría de la siguiente forma

El proyecto comienza **el 6/5 y termina el 16/11**

|  |  |
| --- | --- |
| 6/5 | Inicio configuración interna. Instalación de Ambiente de desarrollo, y arquitectura |
| 6/6 | Se termina la configuración. Se comienza a programar |
| 28/6 | Primera Entrega:  Principales Características Entregadas: Interfaz General, Modulo de Consulta, ABM Consultas, visualización de consultas |
| 16/08 | Segunda Entrega:  Gestión de consultas, Agenda, información de encuestas, gestión de perfiles de usuario. Portal web para el ciudadano con chat y consultas externas. Trazabilidad y registro de uso |
| 13/9 | Tercera Entrega:  Rellamada, gestiones sobre la TSI, programación de citas, uso de documentos. Extracción de listados y estadísticas. Seguimiento de patologías. Mantenimientos varios. Módulos Administrativos |
| 18/10 | Entrega Final:  Puesta en marcha. Sistema completo |
| 2/11 | Comienzo de capacitación a empleados |
| 16/11 | fin proyecto total |



## Detalle en Las Entregas

|  |  |  |
| --- | --- | --- |
| **Detalle de los Casos de uso incluidos en cada entrega** | | |
| **Fecha** | **Número de entrega** | **Casos de Uso incluidos** |
| **28/06/10** | **1° Entrega** | * Integración con el Sistema STARS. * Módulo de seguridad que oculta información del llamado. * (GEN01) Identificación y autenticación. * (USR04) Consulta. * (ISC01) Transferir información comunicación a STARS. * ABM Consultas * ABM Usuarios |
| **16/08/10** | **2° Entrega** | * (GEN02) Gestión General de Registros * (USR02) Agenda * (USR06) Gestionar Consultas Pendientes * (USR12) Encuestas * (USR13) Buscar Consultas * (USR15) Cerrar consulta y dar respuesta al ciudadano * (USR16) Visualizar Consulta * (USR17) Crear/Transferir consulta * (ADM08) Mantenimiento de encuestas * (ADM10) Creación de formularios de encuesta y seguimiento * (ADM11) Mantenimiento de usuarios, perfiles y permisos * (SIS01) Trazabilidad y registro de uso * (ISC02) Iniciar llamada desde STARS * (ISC03) Información encuesta * Protección de acceso * ABM cuestionarios * Interfaz general * Interfaz mantenimiento usuarios y perfiles * Listado de usuarios y perfiles * Interface con sistema de comunicaciones (ISC01, ISC02, ISC03) * Web para Consultas de ciudadanos + chat |
| **13/09/10** | **3° Entrega** | * (USR01) Determinación del servicio de atención * (USR03) Rellamada * (USR05) Consejo Salud * (USR07) Programación de cita previa * (USR08) Gestiones sobre la TSI * (USR09) Servicio de Traducción * (USR11) Seguimiento de patologías * (USR14) Hacer uso de documentos * (ADM01) Mantenimiento del entorno, clasificaciones y prioridades * (ADM02) Extracción de estadísticas y listados * (ADM04) Mantenimiento de recursos * (ADM05) Mantenimiento de guías de interrogatorio * (ADM06) Mantenimiento de fichas temáticas * (ADM07) Mantenimiento de relaciones sobre guías de interrogatorio y fichas temáticas * (ADM09) Mantenimiento de seguimiento de patologías * ABM Fichas/guías/recursos * Interfaz para usuario avanzado |
| **18/10/10** | **Entrega Final** | * (USR10) Información sanitaria general * (ADM03) Mantenimiento de documentos * (ADM12) Listado de acciones proactivas * (ADM13) Listado de documentos * (ADM14) Listado sobre consultas * (ADM15) Mantener asignaciones TSI * (ADM16) Asignar documentos a acciones * (INT01) Direcciones de farmacias, CAP y otra información breve * Interfaz de listados para ADM02 * Interfaz Direcciones INT01 * ABM enfermedades/centros de salud |

# Costo y Formas de Pago

El proyecto tendrá un costo total de $750000, repartido en 3 pagos, cada uno realizado luego de las entregas (parciales y final) según la siguiente tabla

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Periodo | Proyecto | | | | | |
|  | **1** | **2** | **3** | **4** | **5** | **6** |
| Desarrollo de la Solución | $ 0 | $ 300.000 | $ 0 | $ 250.000 | $ 0 | $ 200.000 |
| Total Ingresos | **$ 0** | **$ 300.000** | **$ 0** | **$ 250.000** | **$ 0** | **$ 200.000** |

Los mismos serán pagados al día, mediante transferencia bancaria

# Ganancias Esperadas

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | Proyecto | | | | | |
| Concepto | **Start-up** | **1** | **2** | **3** | **4** | **5** | **6** |
| Ingresos | $ 133.224 | $ 0 | $ 300.000 | $ 0 | $ 250.000 | $ 0 | $ 200.000 |
| Egresos | **$ 79.200** | $ 52.700 | $ 169.346 | $ 124.417 | $ 124.249 | $ 100.179 | $ 100.004 |
| Resultado Bruto Acumulado | **$ 54.024** | **$ 1.324** | **$ 131.979** | **$ 7.561** | **$ 133.312** | **$ 33.132** | **$ 133.128** |
| Impuesto a los Ingresos Brutos | **3,00%** | $ 0 | $ 9.000 | $ 0 | $ 7.500 | $ 0 | $ 6.000 |
| Impuesto a los débitos y créditos bancarios | **0,600%** | $ 316 | $ 2.816 | $ 747 | $ 2.245 | $ 601 | $ 1.800 |
| Resultado Neto antes de impuestos e intereses | **$ 54.024** | **$ 1.008** | **$ 120.162** | **$ 6.815** | **$ 123.566** | **$ 32.531** | **$ 125.328** |
| Gastos financieros | **4,00%** | $ 0 | $ 0 | $ 0 | $ 0 | $ 0 | $ 0 |
| Amortizacones no desembolsables |  |  |  |  |  |  |  |
| Resultado Neto antes de impuestos |  | **$ 1.008** | **$ 120.162** | **$ 6.815** | **$ 123.566** | **$ 32.531** | **$ 125.328** |
| Impuesto a las ganancias | **35,00%** | $ 353 | $ 42.057 | $ 2.385 | $ 43.248 | $ 11.386 | $ 43.865 |
| Resultado Neto Acumulado |  | **$ 655** | **$ 78.106** | **$ 4.429** | **$ 80.318** | **$ 21.145** | **$ 81.463** |

El total del proyecto se financiara con crédito bancario, no necsitando un capital incial para solventar los gastos.

Luego de terminado el mismo y pagado el préstamo solicitado, el estado de resultado nos muestra una ganancia de $67886,14 Calculada con un VAN del 20% anual

|  |  |  |
| --- | --- | --- |
| Valor Neto Actual | $ 67.886,14 | 20,00% |