Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería en Ciencias y Sistemas

OPTIMIZACIÓN E IMPLEMENTACIÓN DE ESTRUCTURAS INTERNAS, MÓDULOS ACTUALES Y NUEVOS PARA LOS DISTINTOS ROLES DE USUARIOS EN EL SISTEMA DE DESARROLLO DE TRANSFERENCIA TECNOLÓGICA (DTT) DE LA ESCUELA DE INGENIERÍA EN CIENCIAS Y SISTEMAS, FACULTAD DE INGENIERÍA, UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

Daniel Alexander Cos Pirir

Asesorado por el Ing. Sergio Arnaldo Méndez Aguilar

Guatemala, marzo de 2018

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

OPTIMIZACIÓN E IMPLEMENTACIÓN DE ESTRUCTURAS INTERNAS, MÓDULOS ACTUALES Y NUEVOS PARA LOS DISTINTOS ROLES DE USUARIOS EN EL SISTEMA DE DESARROLLO DE TRANSFERENCIA TECNOLÓGICA (DTT) DE LA ESCUELA DE INGENIERÍA EN CIENCIAS Y SISTEMAS, FACULTAD DE INGENIERÍA, UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

TRABAJO DE GRADUACIÓN

PRESENTADO A LA JUNTA DIRECTIVA DE LA
FACULTAD DE INGENIERÍA
POR

DANIEL ALEXANDER COS PIRIR
ASESORADO POR EL ING. SERGIO ARNALDO MÉNDEZ AGUILAR

AL CONFERÍRSELE EL TÍTULO DE

INGENIERO EN CIENCIAS Y SISTEMAS

GUATEMALA, MARZO DE 2018

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

NÓMINA DE JUNTA DIRECTIVA

DECANO	Ing. Pedro Antonio Aguilar Polanco
VOCAL I	Ing. Angel Roberto Sic García
VOCAL II	Ing. Pablo Christian de León Rodríguez
VOCAL III	Ing. José Milton de León Bran
VOCAL IV	Br. Oscar Humberto Galicia Nuñez
VOCAL V	Br. Carlos Enrique Gómez Donis
SECRETARIA	Inga. Lesbia Magalí Herrera López

TRIBUNAL QUE PRACTICÓ EL EXAMEN GENERAL PRIVADO

DECANO	Ing. Pedro Antonio Aguilar Polanco
EXAMINADOR	Ing. Marlon Antonio Pérez Türk
EXAMINADOR	Ing. Sergio Leonel Gómez Bravo
EXAMINADORA	Inga. Floriza Felipa Ávila Pesquera
SECRETARIA	Inga. Lesbia Magalí Herrera López

HONORABLE TRIBUNAL EXAMINADOR

En cumplimiento con los preceptos que establece la ley de la Universidad de San Carlos de Guatemala, presento a su consideración mi trabajo de graduación titulado:

OPTIMIZACIÓN E IMPLEMENTACIÓN DE ESTRUCTURAS INTERNAS, MÓDULOS ACTUALES Y NUEVOS PARA LOS DISTINTOS ROLES DE USUARIOS EN EL SISTEMA DE DESARROLLO DE TRANSFERENCIA TECNOLÓGICA (DTT) DE LA ESCUELA DE INGENIERÍA EN CIENCIAS Y SISTEMAS, FACULTAD DE INGENIERÍA, UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

Tema que me fuera asignado por la Dirección de la Escuela de Ingeniería en Ciencias y Sistemas, con fecha 18 de febrero de 2017.

Daniel Alexander Cos Pirir

ACTO QUE DEDICO A:

Dios	Mi todo y que por su infinita misericordia y amor me permite la vida, salud, tiempo, intelecto y todos los recursos para alcanzar esta meta.
Mis padres	Máximo Daniel Cos y Lidia Elizabeth Pirir, por ser mi motor, mi inspiración y mi ejemplo. Definitivamente no alcanzaría esta meta si no fuera por ustedes.
Mis hermanas y hermanos	Thelma, Brenda (q. e. p. d.), Ruth, Elda, Giovanni y Byron Cos Pirir, por haberme acompañado y apoyado siempre en el transcurso de mi carrera, por tenerme la paciencia y comprensión en los momentos en que no pude compartir con ustedes.
Mis cuñados	Oscar Arana y Alex Samayoa por el apoyo que me brindaron en el transcurso de mi carrera.
Mis amigos	Por las experiencias vividas en el transcurso de los años que pasamos en la Universidad.

AGRADECIMIENTOS A:

Universidad de San Carlos de Guatemala	Por ser mi casa de estudios y por ser el medio de formación de profesionales para el desarrollo del país.
Facultad de Ingeniería	Por abrir las puertas y darme la oportunidad de realizarme como profesional de tan magna Facultad.
Ing. Sergio Méndez e Ing. Miguel Marín	Por su valiosa colaboración en la asesoría durante el desarrollo de mi proyecto.
Inga. Floriza Ávila e Inga. Gladys Aceituno	Por su valioso tiempo en orientarme y acompañarme para definir, enfocar y mejorar mi EPS.

ÍNDICE GENERAL

ÍNDICE DE ILUSTRACIONES

FIGURAS

TABLAS

LISTA DE SÍMBOLOS

Símbolo	Significado
MB	Megabyte
Q	Quetzal guatemalteco

GLOSARIO

Backup	Copia de seguridad de datos de forma parcial o total.
DAL	*Database Abstraction Layer* (Capa de abstracción de base de datos).
DTT	Desarrollo de transferencia tecnológica.
Framework	Conjunto de herramientas y lineamientos prediseñados para facilitar la implementación de ciertas tecnologías.
NoSQL	No solo SQL.
Query	Consulta hacia una base de datos.
SQL	*Structured Query Language* (Lenguaje de consulta estructurada).
Trigger	Objeto de la base de datos que se asocia a una tabla.
USAC	Universidad de San Carlos de Guatemala.

RESUMEN

Debido al crecimiento que tuvo el sistema DTT en los últimos años, surgieron varios inconvenientes, tal fue el caso de la verificación de roles para las distintas funciones. En esta verificación se realizaban múltiples consultas a la base de datos SQL, repercutiendo en el rendimiento del sistema cuando este era accedido por muchos usuarios simultáneamente. Tras esto, se rediseño la verificación de roles utilizando una base de datos NoSQL cargada en memoria, enfocada en la rapidez de repuesta.

El segundo inconveniente radicó en la generación de reportes en el rol de administrador. Estos reportes realizaban consultas innecesarias a la base de datos SQL y dejaban en código la lógica de los mismos, dando como resultado un desempeño pobre al momento de ser generados. Se optimizaron las consultas de estos reportes que minimizaron las operaciones en código.

La tercer parte del proyecto consistió en mostrar el número de correos no leídos a cada usuario. Inicialmente este número era calculado cada vez que se cargaba el menú en una página web, esto provocaba una navegación excesivamente lenta. Para solucionar esto se utilizó una base de datos cargada en memoria en donde se guardaron todos los números de notificaciones, de tal forma que no fuera necesario ser calculado a cada momento.

Por último el sistema no contaba con un módulo de copias de seguridad (*backups*) para tener un respaldo de la información de la base de datos SQL. Por

lo cual se desarrolló dicho módulo con la finalidad de aumentar la disponibilidad de los datos y la protección ante posibles caídas del sistema DTT.

OBJETIVOS

General

Optimizar los tiempos de respuesta y aumentar la disponibilidad de los datos en el sistema de desarrollo tecnológico (DTT) de la Escuela de Ingeniería en Ciencias y Sistemas en un lapso de 6 meses.

Específicos

1. Agilizar el proceso de verificación de roles en las diferentes funciones del sistema DTT para todos los roles de usuario.

2. Generación de reportes de manera eficiente para el rol administrador.

3. Indicar el número de correos sin leer a cada usuario sin afectar el rendimiento del sistema.

4. Mantener la integridad y aumentar la disponibilidad de los datos ante posibles caídas del sistema.

INTRODUCCIÓN

En cualquier sistema de gestión como lo es DTT es fundamental el manejo de sesiones. Los roles que pueden acceder a cada función deben ser verificados correctamente, ya que la seguridad es un punto clave. Restringiendo de esta manera las acciones que un usuario pueda realizar dependiendo su rol. Pero esta verificación de roles puede volverse una tarea que afecte el rendimiento, debido a la constante lectura de los mismos, ya que habitualmente se almacenan en una base de datos SQL. De esta manera surge la necesidad de tener un sistema de verificación de roles eficiente.

Otro aspecto importante son los reportes para los administradores, ya que son indispensables para conocer el estado de los estudiantes en el sistema DTT. Dado el hecho que son datos de cantidad considerable, generarlos de una manera óptima se hace fundamental.

Dentro del sistema DTT se requiere que el estudiante sepa cuantos correos tiene sin leer en su bandeja de entrada. Este número debe aumentar o disminuir dependiendo de los envíos o la lectura de los correos, respectivamente. Debe estar previamente calculado para no afectar el rendimiento del sistema.

Fuera del contexto de rendimiento existe otro aspecto fundamental, tal es el caso de las copias de seguridad de la base de datos SQL. Debido a que el sistema DTT guarda información como las notas de cursos, información del estudiante, entre otros, es importante tener respaldo de estos datos ante cualquier eventualidad que podría poner en riesgo la pérdida de la información.

1. FASE DE INVESTIGACIÓN

Esta fase comprende el contexto de la institución que solicitó el proyecto y cuáles son las necesidades que se cubren con su realización. Entendiendo cuales son los problemas que requieren mayor atención y cómo se abordarán a lo largo del proyecto.

1.1. Antecedentes de la empresa

El proyecto DTT surge de la necesidad de comunicación, control y apoyo entre los estudiantes, tutores académicos y catedráticos de la Escuela de Ingeniería en Ciencias y Sistemas. Este sistema fue desarrollado en 3 fases, la primera fase consistió en construir e implementar los cimientos el sistema, los cuales se enfocaron en la automatización del proceso de prácticas finales. Los estudiantes que cursaban dicha práctica final fueron asignados como partícipes del programa, durante la segunda fase se implementaron nuevos módulos, y con ellos se logró un sistema que permitió llevar un seguimiento eficiente y eficaz de las actividades de los cursos, logrando así transparencia en el desarrollo de los mismos, durante la tercera fase se mitigaron varios errores y fallas del sistema, también se implementó un sistema de control de versiones para la gestión del código fuente con que trabajan los distintos estudiantes de EPS que apoyan en el proyecto.

El proyecto DTT en la actualidad es ya una plataforma virtual, en la cual interactúan más de 1 500 estudiantes inscritos que cursan la carrera de Ingeniería en Ciencias y Sistemas de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, catedráticos en conjunto con los tutores académicos,

llevan un mejor control de notas tanto de tareas, proyectos, actividades varias así como de conferencias, reportes y artículos.

1.1.1. Reseña histórica

La carrera de Ingeniería en Ciencias y Sistemas fue fundada en 1971, pasando por diversos cambios y renovaciones en el transcurso de su existencia. Uno de estos cambios es el surgimiento del departamento de transferencia tecnológica, cuyo principal objetivo nació al observar como profesionales que fueron partícipes en la reforma curricular 2011-2012, contribuyeron con ideas de manera entusiasta y con ánimos de mejorar la calidad educativa. El proyecto consistió en continuar con lo que se inició en 2012, con un conjunto de conferencias con temas de gran interés para los estudiantes y docentes de la carrera de Ingeniería en Ciencias y Sistemas. La metodología que fue utilizada para el proyecto DTT, fue hacer partícipes de la actualización curricular a los estudiantes que estaban realizando sus prácticas finales, como asistentes de cátedra (posteriormente llamados tutores académicos) en el proyecto DTT en el período de junio a noviembre de 2012. De esta manera los encargados de organizar y coordinar las conferencias y llevarlas a cabo eran los estudiantes.

1.1.2. Misión

Al estudiante otorgar las competencias acertadas que garanticen el éxito en la búsqueda del conocimiento por medio de los distintos estilos de aprendizaje y fomentando la investigación de manera permanente que le permita una mejor continuidad en su calidad de vida. Tomando en cuenta las opciones que el país ofrece a las distintas áreas del mercado actual (logística, administración, información tecnología, finanzas, contabilidad, comercial, entre otros.), tomando

en cuenta el ámbito internacional debido a la alta competencia que se maneja en estos tiempos.

Proporcionar información sobre los diferentes cambios y actualizaciones que se tiene a nivel mundial para estar enterados de los nuevos sistemas y aplicaciones que se están trabajando[1].

1.1.3. Visión

Reconocer al estudiante de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala como un profesional de alto nivel, con base en los saberes incorporados en el pensum de estudios que permitan formar al estudiante de manera integral para el ejercicio profesional otorgándole los instrumentos adecuados para su desarrollo ocupacional[2].

1.1.4. Servicio que realiza

La Escuela de Ingeniería en Ciencias y Sistemas se encarga de gestionar y planificar actividades que estén relacionadas con el crecimiento académico de los estudiantes. También es el ente que supervisa y evalúa las técnicas utilizadas en la enseñanza, lleva el control de las actividades realizadas por los catedráticos, tutores académicos y estudiantes.

En los diferentes cursos de la carrera tiene el control de los programas en los ciclos académicos, con el propósito de cumplir con los requerimientos y evaluaciones internas y externas del currículo.

---

[1] Escuela de Ciencias y Sistemas (ECYS-FIUSAC). [en línea]. <https://wikiversidad.wikispaces.com/Escuela+de+Ciencias+y+Sistemas+(ECYS-FIUSAC)>. [Consulta: 20 de junio de 2017]

[2] Ibíd.

Lleva a cabo la administración del área académica y cuenta con oficinas donde se encuentra el personal que dirige a esta organización.

1.2. Descripción de las necesidades

En esta sección se describen las necesidades de las cuales surge el proyecto de EPS.

1.2.1. Reestructuración del manejo de sesiones y verificación de roles

La necesidad de esta reestructuración surgió debido a problemas de rendimiento ante una alta concurrencia de usuarios. Se realizó un análisis en el cual se concluyó que el sistema realizaba excesivas consultas a la base de datos SQL. Entre las consultas realizadas se observó que había una en específico que se repetía constantemente, la cual era procesada para obtener los roles al que un usuario pertenecía, con el fin de comparar si el rol necesario para acceder a una función estaba contenido en estos. Este proceso se realizaba cada vez que el usuario accedía a una función del sistema.

Debido a lo anterior se determinó que la manera de verificación de roles del sistema no era óptima y se propuso disminuir el número de consultas a la base de datos SQL en este proceso. De allí surge la idea de utilizar otro método de verificación de roles con la ayuda de las bases de datos NoSQL, pero esto conlleva a que el manejo de sesiones también sea migrado a esta nueva base de datos y por ende una reestructuración completa de estos procesos.

1.2.2. Implementación de *queries* SQL en reportes principales del rol de administrador

Cada uno de estos reportes tenía un tiempo de respuesta alto al generarse, debido a que la lógica que se utilizó para realizarlos era ineficiente. Se realizaban consultas a la base de datos relacional, las cuales retornaban todo el contenido de las tablas, posteriormente se recorrían los datos con ciclos *for* y se filtraban con comparaciones *if*, este proceso era extremadamente lento ya que se realizaba desde código, y empeoraba con el tiempo ya que los datos guardados en las tabla aumentan cada periodo en que el sistema esté en funcionamiento.

Tomando en cuenta este inconveniente surge la necesidad de implementar *queries* complejos los cuales realicen la lógica de estos desde la base de datos SQL y que retornen los datos ya procesados.

1.2.3. Reestructuración del cálculo del número de correos no leídos en los roles de usuario universitario, alumno y catedrático

Debido a que los correos almacenados han tenido un crecimiento alto en los periodos en que el sistema DTT ha estado en funcionamiento, el cálculo del número de correos no leídos tuvo un impacto negativo debido a la lógica que se utilizaba para generarlo. Este cálculo era realizado cada vez que se cargaba el menú en una página del sistema DTT y se mostraba en la opción de bandeja de entrada.

El procedimiento utilizado para tener este número de correos no leídos era bastante lento, ya que se ejecutaba una consulta a la base de datos relacional, la cual tenía que cotejar dos tablas, una donde se encuentran todos los correos

con su detalle y otra que almacena los correos que ya han sido leídos. Posteriormente se realizaba un conteo de los correos que no se encontraban dentro de la tabla de correos leídos, para obtener de esta manera el número de los no leídos. Esta es una tarea que consume bastante recursos y que debido a la magnitud de datos, la complejidad aumenta y los tiempos de respuesta son altos.

Debido a estas complicaciones se necesitaba un método eficaz para poder mostrar al usuario el número de correos no leídos. Nuevamente se propuso utilizar una base de datos NoSQL que ayudara con este proceso.

1.2.4. Módulo de administración de copias de seguridad automáticas para la base de datos SQL

El sistema DTT no contaba con un módulo de esta índole. Normalmente las copias de seguridad se generaban utilizando los comandos que la base de datos brinda, pero para realizarlos se necesitaba una persona experta en este campo. Estas copias de seguridad no tenían un periodo establecido para realizarse, resultando en un riesgo de disponibilidad e integridad de datos.

Ante estos problemas se hizo necesario contar con un módulo en el cual se pudiera programar la realización de copias de seguridad automáticas y brindar una plataforma sencilla para que la persona responsable de administrarlos, no necesariamente fuera experta en esos temas, disminuyendo así el riesgo de pérdida de datos y manteniendo la integridad. Con esto se aseguraría una plataforma con un respaldo ante fallos en los servidores donde se encuentra instalado.

1.3. Priorización de las necesidades

En la tabla I se muestra la importancia de cada uno de los aspectos que contiene este proyecto.

Se define la prioridad de la manera siguiente:

- 1 al 3: prioridad baja, no urgente
- 4 al 7: prioridad media, urgente pero puede esperar
- 8 al 10: prioridad alta, urgente y no puede esperar

Tabla I. Prioridad e impacto de las necesidades

Prioridad	Impacto	Necesidad
10	Alto	Reestructuración del manejo de sesiones y verificación de roles.
8	Alto	Módulo de administración de copias de seguridad automáticas para la base de datos SQL.
6	Medio	Implementación de *queries* SQL en reportes principales del rol administrador.
5	Medio	Reestructuración del cálculo del número de correos no leídos en los roles de usuario universitario, alumno y catedrático.

Fuente: elaboración propia.

2. FASE TÉCNICO PROFESIONAL

En este capítulo se describe como fue abordada la solución para cada necesidad presentada en el capítulo anterior.

2.1. Descripción del proyecto

En esta sección se especificarán los procesos que se utilizaran para realizar cada aspecto del proyecto.

2.1.1. Reestructuración del manejo de sesiones y verificación de roles

Para esta reestructuración se usará una base de datos cargada en memoria para almacenar las sesiones. Estas sesiones tendrán un identificador único, el cual estará almacenado en la base de datos junto al listado de roles al que el usuario pertenezca.

Posterior a esto se enviará una *cookie* al usuario con el identificador de la sesión encriptado, de manera que no sea legible y al momento de ser devuelta al sistema sea capaz de verificar si la sesión esta activa y si el usuario pertenece al rol correspondiente para realizar alguna acción si así fuera requerido.

Esta sesión deberá tener un tiempo de expiración, el cual dependerá de las necesidades del negocio. Este tiempo transcurrirá a partir de que el usuario ya no realice acciones dentro del sistema y llegado al final de este tiempo la sesión expirará automáticamente.

2.1.2. Implementación de *queries* SQL en reportes principales del rol de administrador

En esta parte se implementarán *queries* SQL que sustituyan las consultas realizadas por medio de la DAL del *framework*. Asimismo se creará una librería que contenga los *queries* SQL, de manera que estén centralizados en una sola parte del código y no dispersos en cada uno de los métodos.

Los reportes a optimizar son los siguientes:

- Reporte por estado: este presenta el estado de los reportes de los tutores académicos.
- Anomalías por periodo: presenta las anomalías registradas en el sistema en un periodo determinado.
- Resumen de semestre: es el resumen del resultado de las actividades realizadas en todos los proyectos registrados dentro del sistema.
- Gestión de estudiantes: presenta el histórico de acciones realizadas en datos de estudiantes.
- Gestión de asignaciones con métrica: presenta el histórico de las gestiones de asignaciones con métrica.
- Gestión de notas: presenta el histórico de las gestiones en las notas de las actividades de los proyectos dentro del sistema.
- Gestión de actividades con métrica: presenta el histórico de las actividades que contienen métrica de los proyectos en el sistema.
- Gestión de solicitudes de cambios de actividades con métrica: presenta el histórico de solicitudes de cambios en actividades de proyectos.

2.1.3. Reestructuración del cálculo del número de correos no leídos en los roles de usuario universitario, alumno y catedrático

Para esta reestructuración se utilizará nuevamente la base de datos en memoria. Como primer paso se deberá realizar un método que calcule el número de correos no leídos para cada uno de los usuarios que pertenezcan a los roles universitario, alumno y catedrático. Este cálculo será hará una sola vez para cada usuario y se almacenará en la base de datos.

Posteriormente se deberá modificar el proceso de envío de correos, para que cuando se genere uno nuevo, el número de correos no leídos, aumente para los usuarios destinatarios. Asimismo se tendrá que modificar el proceso de lectura de correos, para que cuando un usuario ingrese a su bandeja de entrada y revise sus correos el número de correos no leídos disminuya.

Por último se deberá presentar el número de correos no leídos al usuario, en la parte derecha de la opción del menú bandeja de entrada.

2.1.4. Módulo de administración de copias de seguridad automáticas para la base de datos SQL

En este último punto se creará un módulo nuevo dentro del sistema DTT que permitirá la configuración de copias de seguridad por parte del administrador del sitio. En este módulo se podrá configurar varios aspectos, tales como la activación o desactivación del proceso, el tipo de copia de seguridad a crear, el servidor donde se almacenará, entre otros.

El módulo llevará un registro de las copias de seguridad realizadas el cual mostrará el estado del proceso de creación, así como también la ruta y el servidor donde se encuentran almacenados. Por último el proceso creara un *script* de recuperación el cual servirá para la restauración de la base de datos a partir de las copias de seguridad creadas en un tiempo determinado.

2.2. Investigación preliminar para la solución del proyecto

En esta sección se describen las ventajas que brindan las tecnologías que se utilizan en cada parte del proyecto.

2.2.1. Reestructuración del manejo de sesiones y verificación de roles

Para este primer aspecto, se investigó sobre motores de base de datos que optimizaran la lectura de datos para poder obtener rápidamente los registros almacenados.

2.2.1.1. Base de datos

Se optó por utilizar la base de datos redis, ya que brinda un esquema clave-valor el cual se acopla muy bien al manejo de sesiones. Además es una base de datos muy rápida, dado a que carga los datos en memoria, dando como resultado respuestas casi inmediatas.

Pero esto no significa que los datos no sean persistentes, ya que cada cierto tiempo los datos cargados a memoria se guardan en disco, por ende aunque se reinicie el servidor los datos quedarán guardados y al momento de que vuelva a iniciar el servidor, estos serán cargados nuevamente a memoria.

Otro aspecto importante acerca de redis es que se puede colocar un tiempo de expiración a una clave determinada, dándonos la facilidad de manejar esta funcionalidad de las sesiones directamente en la base de datos y no en código.

2.2.1.2. Métodos de inicio de sesión y verificación de roles

Para los métodos de inicio de sesión se inspeccionó detalladamente el código de la librería encargada de esto. Se detectaron los métodos involucrados en el inicio de sesión y cuál era el flujo de ejecución de dichos métodos.

Luego para la verificación de roles se tenían dos alternativas: la primera consistía en modificar cada una de las funciones del sistema DTT con el objetivo de que la verificación se hiciera a través de la clave almacenada en redis. Pero esto llevaba a una exhaustiva modificación de todo el sistema y por consiguiente la verificación de cada una de las funciones, dando como resultado un tiempo muy largo en la fase de validaciones y pruebas.

La siguiente alternativa surgió al detectar que todas las funciones ya contenían un método verificador de roles, resultado de la manera en que esto se realizaba. Este método leía los roles de un usuario en la base de datos SQL y se encargaba de verificar dichos roles. Dado este hecho se procedió a proponer la modificación del método y cambiar la verificación hacia una base de datos en memoria, en este caso redis.

2.2.1.3. Uso de *cookies* para validar la sesión

Teniendo en cuenta que es necesario que cada petición que se realice al servidor sea identificada, ubicando de qué usuario proviene y si este ya ha iniciado sesión anteriormente en el sistema, se hace necesaria una manera de verificarla. Es allí donde surge la necesidad de enviar *cookies* al navegador del usuario. Una *cookie* como bien es conocido son pequeños datos que se envían al usuario, con la función de que estas al ser devueltas al servidor cuando se realiza alguna petición, este sea capaz de tomar esta información, realizar alguna acción y devolver un resultado especifico dependiendo del contenido de la *cookie*.

De igual manera se envía al usuario una *cookie* con el identificador único de su sesión de manera que cuando esta sea devuelta al servidor, éste pueda verificar si existen claves en redis que contengan ese valor y de esta forma determinar que el usuario anteriormente ha iniciado sesión en el sistema. Pero esa no será la única función de la *cookie*, además de la verificación de la sesión, también servirá para verificar los roles a los que pertenece el usuario.

2.2.2. Implementación de *queries* SQL en reportes principales del rol de administrador

Para esta parte se investigó sobre la capa de abstracción de datos del *framework* para utilizarla de manera óptima o usar otra alternativa.

2.2.2.1. La DAL (database abstraction layer)

La capa de abstracción de base de datos es una API que asocia objetos de bases de datos SQL con objetos en el lenguaje de programación. Debido a que

toda la programación del sistema se basa en ella, los reportes no eran la excepción.

Se analizó cómo trabajaba la DAL y la sintaxis de la misma para poder trasladar estas consultas a *queries* SQL. Se investigó si existía un método en la DAL que nos diera la facilidad de ejecutar *queries*, para no tener que instanciar una nueva conexión hacia la base de datos y aprovechar la ya existente. Se determinó que existe un método el cual realiza precisamente la ejecución de un *query* SQL.

Teniendo en cuenta que los *queries* estarían ahora en una cadena, surgió la necesidad de centralizarlos de manera que no estuviera por todas las partes del código, sino que siguieran un estándar, por lo tanto, crear una librería que devolviera los *queries* solicitados era indispensable.

2.2.2.2. Análisis de reportes a modificar

Se realizó un análisis detallado de cada parte de los reportes para detectar cuáles eran los que más problemas tenían en la parte de programación y que también fueran claves para el administrador. Con lo cual se seleccionaron 8 reportes:

- Reporte por estado: este reporte resultó ser el más crítico debido a que su ejecución duraba aproximadamente 4 minutos. Y la información que genera es indispensable para el administrador del sistema DTT.
- Anomalías por periodo: este reporte tenía problemas de consistencia de datos, por ende no solo había que optimizarlo sino que también volver a generar de manera correcta las consultas. La información brindada por este reporte es importante para el administrador del sistema.

- Resumen de semestre: al igual que el anterior, este reporte carecía de la consistencia de datos. También era necesario optimizarlo y generar nuevamente las consultas. La información generada por el reporte es importante para el administrador del sistema.
- Gestión de estudiantes
- Gestión de asignaciones con métrica
- Gestión de notas
- Gestión de actividades con métrica
- Gestión de solicitudes de cambios de actividades con métrica

Para los reportes de gestión solo era necesario pasar los *queries* utilizados hacia la librería para estandarizar el uso de los mismos.

2.2.3. Reestructuración del cálculo del número de correos no leídos en los roles de usuario universitario, alumno y catedrático

En esta sección se describen aspectos importantes de las herramientas seleccionadas para poder reestructurar el cálculo de los correos no leídos en cada uno de los roles dentro del sistema DTT.

2.2.3.1. Base de datos

Al igual que en el caso del manejo de sesiones para el número de correos no leídos se requería una base de datos rápida. Y redis nuevamente cumplió con estos requisitos debido a que nos da la facilidad de tener un máximo de 16 bases de datos, brindándonos la opción de manejar el número de correos no leídos en una base distinta a la del manejo de sesiones.

2.2.3.2. Calculo de correos no leídos

Para esta reestructuración se realizó una inspección de la forma anterior de calcular el número de correos no leídos, ubicando el procedimiento dentro de la creación del menú principal del sistema.

Este procedimiento realizaba varias consultas a la base de datos usando la DAL y eran utilizadas de manera ineficiente. Estas consultas debían cambiarse utilizando *queries* SQL y de ser posible el procedimiento de cálculo debía realizarse una única vez.

2.2.3.3. Aumento y disminución de número de correos no leídos

Para este caso se tenían dos opciones: la primera consistía en modificar cada uno de los métodos de envío y lectura de correos para ir aumentando o disminuyendo el número de correos no leídos de los destinatarios.

El problema consistía en ubicar cada función dentro del sistema que tuviera relación con el envío o lectura de correos, ya que algunos se envían de manera automática y no existía una manera centralizada y estandarizada de realizar dicho envío. Además de este inconveniente se tenía el problema de que en futuras implementaciones dentro del sistema se utilizarán envíos de correos, dando como resultado que el alcance de esta modificación no cubriría los nuevos desarrollos.

La segunda opción consistía en realizar un *trigger* en la base de datos SQL, el cual al momento de existir una inserción dentro de la tabla que almacena los correos enviados, ejecutará un método que aumentará el número de

notificaciones para el usuario destinatario. Asimismo un *trigger* para la tabla que almacena los correos leídos, el cual ejecutara un método que disminuyera el número de notificaciones del usuario en cuestión.

2.2.4. Módulo de administración de copias de seguridad automáticas para la base de datos SQL

En esta sección se detallan diferentes aspectos que conciernen a las copias de seguridad y que ayudaron a definir un módulo eficiente y aceptado por el usuario final.

2.2.4.1. Tipos de copias de seguridad

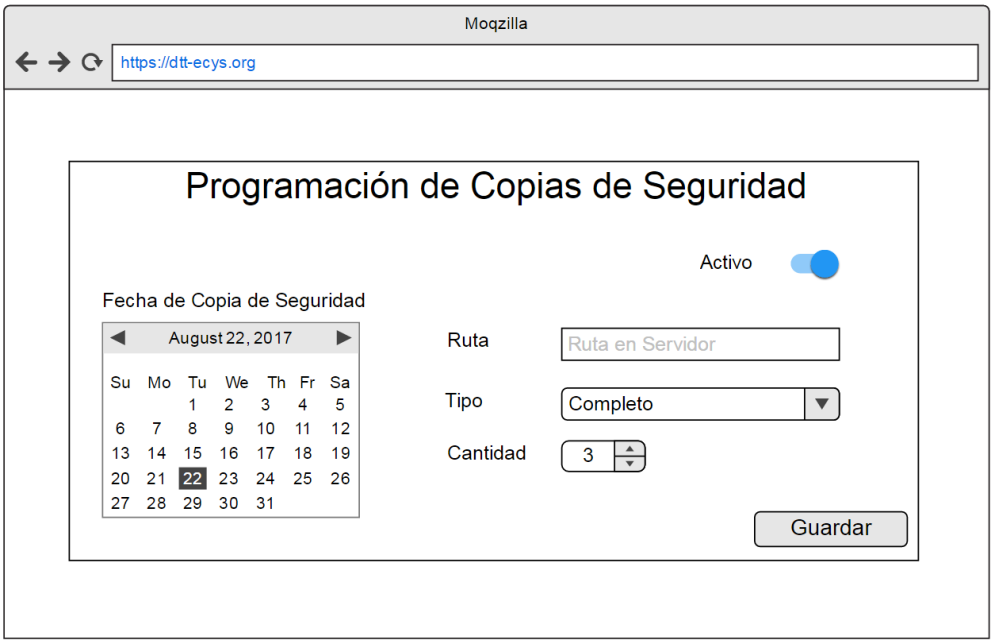
Se investigó acerca de los tipos de copias de seguridad que podrían realizarse desde la base de datos SQL, determinando los siguientes:

- Completo: este tipo de copia de seguridad contiene la totalidad de datos que existan. El inconveniente de este radica en el tiempo que tarda en realizarse y el espacio en disco que ocupa, ya que se trata de una copia total. La ventaja de una copia de seguridad completa es que al momento de restaurar se tendrá el juego completo de datos y no se necesita realizar operaciones complejas para restaurarlo.
- Incremental: este tipo de copia de seguridad contendrá los datos nuevos, que cambien o las eliminaciones a partir de la última copia de seguridad completa realizada. A pesar de que estas copias son más pequeñas y utilizan de mejor manera el espacio en disco, tienen el inconveniente de que al momento de restaurarlas requieren varios procesos, lo cual repercute en el tiempo de restauración.

2.2.4.2. Interfaz del módulo

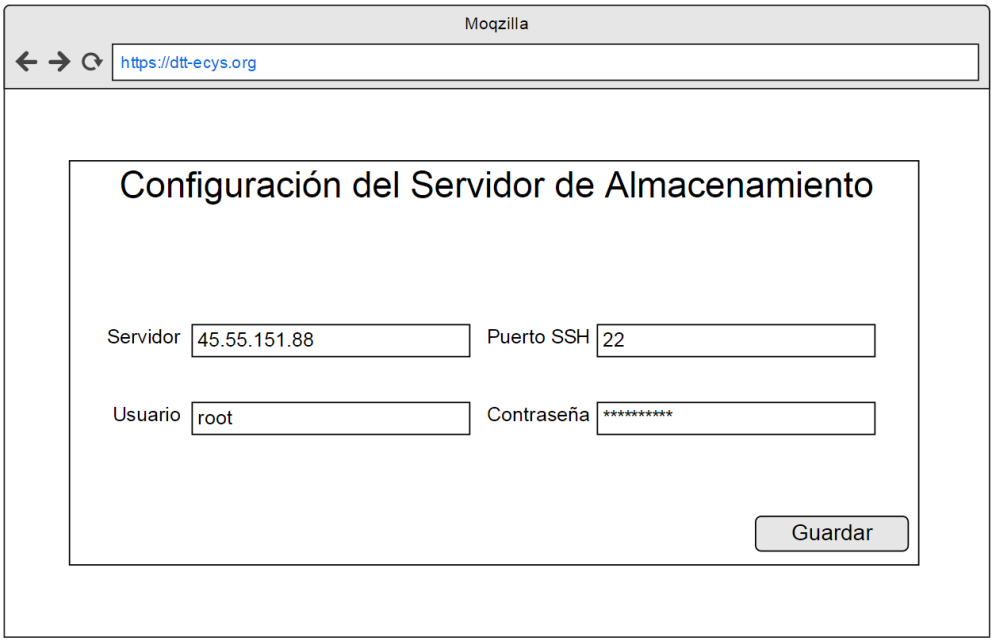
Para la interfaz del módulo no existía otra posibilidad más que realizarla dentro del mismo marco en que está desarrollado el sistema DTT. Este está orientado a la generación de aplicaciones web de manera segura, escalables y con el enfoque en base de datos.

Figura 1. **Prototipo pantalla de programación de copias de seguridad**



Fuente: elaboración propia.

Figura 2. **Prototipo pantalla de configuración de servidor de almacenamiento**



Fuente: elaboración propia.

Figura 3. **Prototipo histórico de copias de seguridad**

Fuente: elaboración propia.

2.2.4.3. Proceso periódico de creación de copias de seguridad

Se hizo necesario un proceso a nivel de sistema operativo el cual ejecutara comandos para la creación de las copias de seguridad y a su vez actualizara la próxima fecha de realización. Este proceso debía ser escrito en un lenguaje nativo del sistema operativo. Se utilizaría el intérprete de Linux *bash* para ejecutar un *script* programado en lenguaje *shell* el cual se encargaría de generar la copia de seguridad y trasladarla a un servidor de almacenamiento.

Este proceso, dependiendo la configuración, se ejecutaría una única vez o bien periódicamente.

2.2.4.4. Herramienta de creación de copias de seguridad

Para la realización de las copias de seguridad se investigó herramientas las cuales tuvieran la opción de realizar copias de seguridad incrementales.

Dado lo anterior se decidió utilizar percona *xtrabackup*, la cual brinda la opción de crear copias de seguridad en caliente, dando como resultado que el sistema no deje de funcionar durante el proceso de generación.

2.3. Presentación de la solución al proyecto

A continuación se describe detalladamente cual fue la solución que fue implementada de acuerdo con los requerimientos de cada módulo, para llegar a un resultado satisfactorio y óptimo.

2.3.1. Reestructuración del manejo de sesiones y verificación de roles

En esta sección se define como se realizó la reestructuración tanto en el manejo de sesiones como en la verificación de los roles, explicando el proceso de cada función.

2.3.1.1. Base de datos

La base de datos utilizada para esta reestructuración fue redis, la cúal es un motor de base de datos que almacena los datos en memoria, utiliza un esquema clave-valor de tablas *hash* y debido a que su contenido radica en memoria es altamente rápida para la obtención de los datos almacenados.

Debido al esquema que utiliza no soporta consultas SQL, sino que únicamente utilizará la clave como referencia para retornar el valor que esta almacene. Redis contiene un máximo de 16 bases de datos. Los datos almacenados son de tipo cadena de texto.

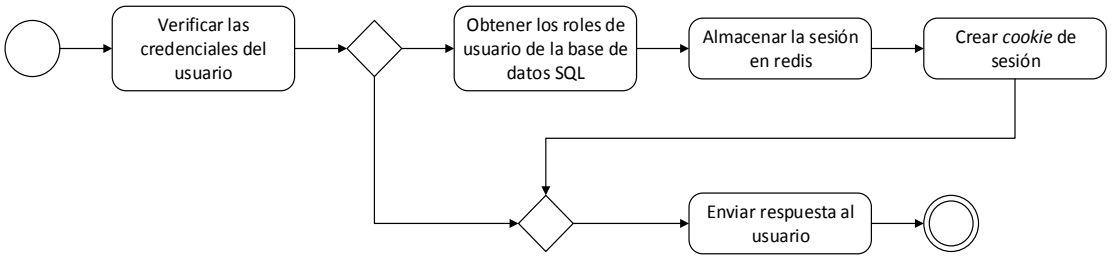
2.3.1.2. Proceso de creación y almacenamiento de sesiones

Inicialmente el usuario envía al servidor la petición de inicio de sesión y el sistema corrobora en la base de datos SQL que el usuario existe y su contraseña sea la correcta. Posteriormente se da comienzo al proceso de almacenamiento de las sesiones en redis. Como primer paso se obtienen de la base de datos SQL los roles al que el usuario pertenece.

Luego se crea un identificador único que distinguirá a la sesión. Para el almacenamiento en redis, se le concatena a cada uno de los roles, el identificador generado anteriormente y se almacena cada uno de estos. De esta manera se tendrán varias claves en redis para esta sesión, cada una de estas pertenecientes a un rol que el usuario desempeñe dentro del sistema.

Una vez almacenados los roles de la sesión se genera una *cookie* cuya función será almacenar el identificador único que se creó para la sesión. Esta *cookie* será fundamental para validar que la sesión sea correcta y posteriormente para la verificación de roles.

Figura 4. **Diagrama de actividades de creación y almacenamiento de sesiones**



Fuente: elaboración propia, empleando Visio.

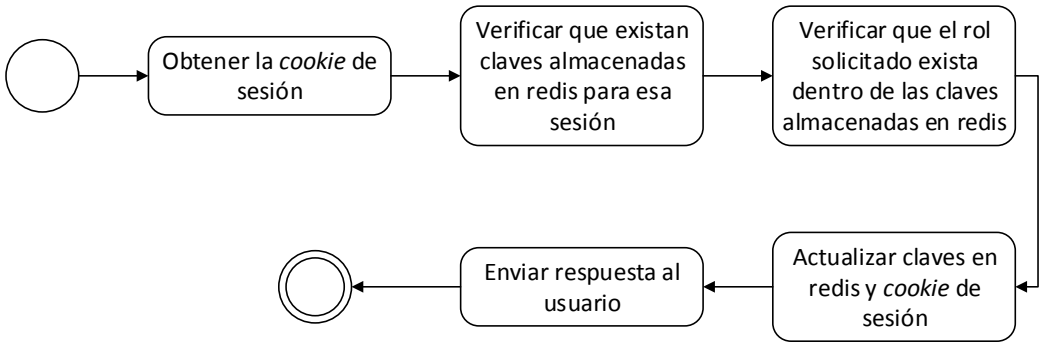
2.3.1.3. Proceso de verificación de roles

Al momento en que un usuario desea acceder a una función en el sistema, éste envía una solicitud al servidor. En la solicitud viaja la *cookie* de sesión la cual es parte fundamental de la verificación de roles. Una vez el sistema obtiene la *cookie* de sesión, extrae su contenido que es el identificador único que se generó cuando el usuario inicio su sesión.

Posteriormente se verifica que existan claves en redis que contengan el identificador. Si se retornan valores se da por válida la sesión y en caso contrario, se redirecciona al usuario a la página de inicio de sesión. Si la sesión es válida se obtiene el rol al que le es permitido acceder a la función solicitada y se le concatena el identificador único proveniente de la *cookie* de sesión.

Por último se verifica que este valor exista como clave en redis. Si la clave existe se brinda acceso a la función solicitada y si no existe, se finaliza la sesión del usuario y se redirecciona a la página de inicio de sesión.

Figura 5. **Diagrama de actividades de verificación de roles**



Fuente: elaboración propia, empleando Visio.

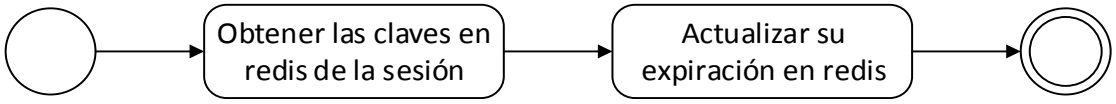
2.3.1.4. Proceso de actualización expiración de sesiones

Las sesiones necesitan tener un tiempo de expiración para evitar que se saturen las claves almacenadas. Este tiempo de expiración depende de las necesidades del negocio y afectarán únicamente a aquellas sesiones que se detecten como inactivas en un determinado periodo de tiempo.

Este proceso se hace a través de la base de datos redis, la cual provee una manera de colocar un tiempo de expiración a una clave determinada. Al momento en que llega al límite del tiempo de expiración que se ha colocado a una clave, esta es eliminada de la base de datos.

En el sistema DTT cada vez que un usuario accede a una función, los tiempos de expiración de las claves pertenecientes a esa sesión son actualizados. De esta manera cuando el usuario no realice ninguna acción dentro del sistema en un determinado periodo de tiempo, las claves de la sesión se eliminaran automáticamente de redis.

Figura 6. **Diagrama de actividades de actualización de expiración de sesiones**



Fuente: elaboración propia, empleando Visio

2.3.1.5. Proceso de eliminación de sesiones

Para este proceso únicamente se expiran todas las claves en Redis relacionadas con el identificador único de la sesión y conjuntamente se expira la *cookie* de sesión.

2.3.2. Implementación de *queries* SQL en reportes principales del rol de administrador

Para iniciar con esta implementación se creó una librería de *queries*, la cual quedó alojada en los módulos del proyecto de web2py. Posteriormente se importó en cada parte del código donde se requería utilizar algún *query*, mandando a llamar al método que retornaba la cadena con el *query* deseado y ejecutándolo con la ayuda del método de la DAL execute_sql().

Reporte por estado: para este reporte fue necesario la creación del *query* correspondiente. Este *query* reemplazo a la codificación realizada con consultas de DAL. Fue necesario hacerle modificaciones a la vista ya que estaba adecuada a la manera de consultas de la DAL y no hacia los resultados del *query*. También se modificaron las vistas que pertenecían a los detalles de los resultados que este reporte mostraba, ya que estaban adecuadas a la manera antigua de generación del reporte.

Anomalías por periodo: en este reporte se reemplazó nuevamente las consultas de DAL por consultas SQL. Se creó desde cero el *query* necesario para la generación de la lógica del reporte ya que las consultas de DAL contenían errores y no mostraban la información correcta. También la vista de este reporte sufrió modificaciones, ya que se adecuó a los resultados devueltos por la ejecución del *query* SQL.

Resumen de semestre: para este reporte fue necesario crear un *query* con la lógica del reporte, ya que las consultas de DAL contenían errores y mostraban información incorrecta. Posteriormente se modificó la vista para adecuarla a los resultados del *query*. También se categorizó la forma de mostrar los datos, ya que inicialmente se mostraba un reporte general que contemplaba todas las áreas del sistema. Se colocó la opción de elegir el área a mostrar.

Para los reportes de:

- Gestión de estudiantes
- Gestión de asignaciones con métrica
- Gestión de notas
- Gestión de actividades con métrica
- Gestión de solicitudes de cambios de actividades con métrica

Se trasladaron los *queries* que implementaban a la librería y luego se realizó la llamada correspondiente al *query* que se requería en cada aspecto de la lógica del reporte.

2.3.3. Reestructuración del cálculo del número de correos no leídos en los roles de usuario universitario, alumno y catedrático

Se inició generando un *query* que contara el número de correos no leídos para un usuario en específico. Posteriormente se realizó un método, el cual ejecutara la consulta con el *query* generado anteriormente y obteniendo de esta manera el número de correos no leídos para un usuario en específico.

Posteriormente este número se almacenó en redis, en la base de datos número 1, teniendo como clave el *username* del usuario y de valor el número de notificaciones obtenido con el método.

Para el aumento y disminución de los valores de dichos números en redis, se creó un UDF (u*ser-defined function*) y se añadió a la base de datos SQL. Esta función recibe como parámetro el *username* del usuario y un parámetro que indica si se aumenta o disminuye el número de notificaciones, posterior realiza la conexión a redis y ejecuta el comando SET con el nuevo valor de notificaciones para el *username* enviado.

Esta función UDF se llama a través de varios *triggers*. Estos *triggers* se activan cuando se realiza una inserción en las tablas *notification_log* y *read_mail*. Ya teniendo el manejo de este número se procede a mostrarlo en el menú al lado derecho de la opción bandeja de entrada. La obtención del número se realiza a través de una conexión hacia la base de datos redis y con la ejecución del comando GET.

2.3.4. Módulo de administración de copias de seguridad automáticas para la base de datos SQL

Como primer paso se instaló la herramienta percona *xtrabackup*. Posteriormente se creó en la base de datos SQL una tabla la cual sería la encargada de guardar la configuración de las copias de seguridad y otra que serviría para el histórico.

Para la proceso de creación de copias de seguridad se realizó un *script shell*, el cual contiene toda la lógica necesaria para cargar la configuración y realizar la copia de seguridad, trasladándola a un servidor de almacenamiento.

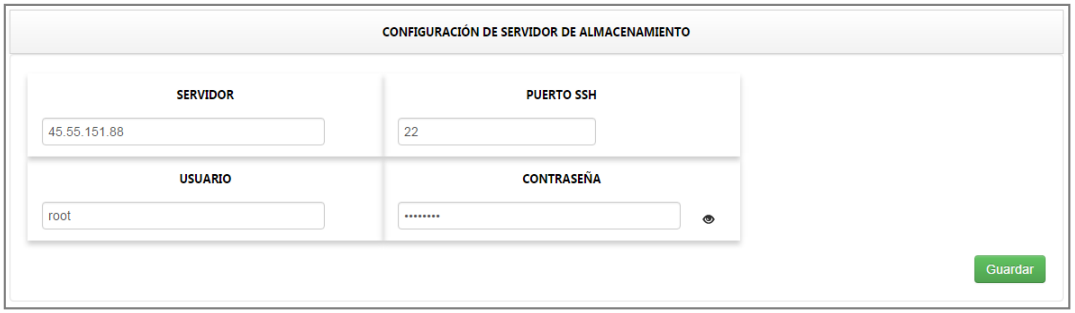
Paralelamente este proceso se encarga de crear un *script* de recuperación el cual puede utilizarse posteriormente para restaurar la base de datos SQL.

Por último se realizó la interfaz gráfica de este módulo dentro del sistema DTT. La cual únicamente muestra los campos necesarios para la configuración de las copias de seguridad, como también del servidor de almacenamiento. En un apartado muestra el histórico de creación de copias de seguridad.

Figura 7. **Programación de copias de seguridad**

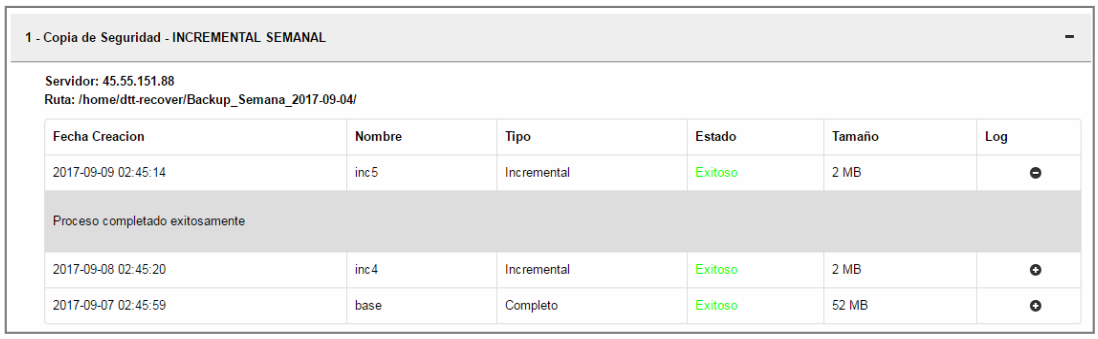
Fuente: elaboración propia, empleando captura de pantalla.

Figura 8. **Configuración de servidor de almacenamiento**



Fuente: elaboración propia, empleando captura de pantalla.

Figura 9. **Histórico de copias de seguridad**



Fuente: elaboración propia, empleando captura de pantalla.

2.4. Costos del proyecto

Tabla II. Costos del proyecto

Recursos	Cantidad	Costo Unitario	Subtotal
Transporte	120 días	Q25,00	Q3 000,00
Energía Eléctrica	120 días	Q20,00	Q2 400,00
Consultora institución	1 persona durante 20 días	Q600 el día	Q12 000,00
Consultor escuela.	1 persona durante 40 días	Q600 el día	Q24 000,00
Desarrollo	1 persona durante 180 días	Q300 el día	Q54 000,00
Impresiones	300 impresiones	Q0,50	Q150,00
Materiales de oficina		Q500,00	Q500,00
		Total	**Q96 050,0**

Fuente: elaboración propia.

2.5. Beneficios del proyecto

- Mejores tiempos de respuesta al verificar permisos en cada función del sistema DTT.
- Usuarios satisfechos con el rendimiento del sistema DTT y por ende lo utilizarán más para llevar sus controles de cursos.
- Generación de reportes para los administradores de forma rápida y eficiente.
- Presentación del número de notificaciones al usuario sin repercutir en el desempeño del sistema.
- Brindar una manera sencilla de configurar copias de seguridad para el resguardo de la información del sistema.

3. FASE DE ENSEÑANZA APRENDIZAJE

En esta sección se abarcará el proceso en el cual se han capacitado a las diferentes partes interesadas en el proyecto.

3.1. Proceso de capacitación

Se realizaron una serie de reuniones que incluían al asesor de proyecto EPS como al asesor de la institución. Así también hubo reuniones separadas para cada uno.

3.1.1. Capacitación presencial al asesor de proyecto

Se realizaron 5 reuniones en las cuales se presentaron avances de cada una de las fases de este proyecto de EPS. En estas reuniones se capacitó al asesor en cómo utilizar las mejoras realizadas, y en cuál era el proceso para que se instalarán en el ambiente de producción, así como también aspectos técnicos que conllevaban cada uno de los avances presentados. En cada reunión se obtuvo retroalimentación del asesor para mejorar funciones y formas de realizar algunos procesos, siempre y cuando estos aplicarán y no salieran del alcance del proyecto. Existieron casos en los que se debatió acerca de algunas configuraciones, tal fue el caso del tiempo límite que duraría una sesión al momento de detectar que el usuario estaba inactivo en el sistema.

Cada una de estas reuniones está respaldada por una minuta de avance, donde se dejó constancia por escrito de cada avance y de cuál era el porcentaje de progreso en cada una de las actividades realizadas.

3.1.2. Capacitación presencial al asesor de institución

Estas reuniones se realizaron cada 2 semanas, los días sábados. En estas reuniones se dialogaba con el asesor de institución el cual era el encargado de aceptar cada avance que se hiciera del proyecto. Se mostraba lo hecho hasta el momento y se le capacitaba en la utilización de las mejoras realizadas. No se llegaba a aspectos técnicos, solamente a aspectos funcionales. Cada vez que se terminaba alguna fase del proyecto se le presentaba en un ambiente de desarrollo y se dialogaba sobre las fechas para que esto pudiera llegar a producción. Si todo lo anterior se terminaba correctamente, él daba el visto bueno y firmaba una minuta de avance.

3.2. Material elaborado

Se elaboraron los respectivos manuales técnicos para cada fase del proyecto y los manuales de usuario en las fases que así lo requerían. Cada manual técnico detalla que archivos son los que fueron modificados, eliminados o agregados, así como la guía de cómo instalar algunos componentes necesarios para el buen funcionamiento de las mejoras realizadas.

CONCLUSIONES

1. El uso de una base de datos cargada en memoria para la verificación de permisos dentro de un sistema es más eficiente que con una base de datos relacional.

2. El uso de *queries* SQL para la generación de los datos a mostrar en un reporte es más óptimo que procesarlos en código.

3. El número de correos no leídos debe ser previamente calculado para únicamente ser mostrado al usuario.

4. Con la creación de copias de seguridad se minimiza el riesgo de pérdida de datos y entre más corto sea el periodo de realización de una a otra, menos información estará en riesgo.

RECOMENDACIONES

1. Implementar bases de datos cargadas en memoria para mejorar tiempos de respuesta procesos de verificaciones de roles y permisos.

2. Utilizar las herramientas que nos brindan las bases de datos para procesar información ya que ellas están optimizadas para eso.

3. Documentar los nuevos desarrollos para tener retroalimentación ante alguna falla de algún módulo dentro del sistema DTT.

4. Establecer un estándar de distribución de las fuentes y desarrollo de código dentro del sistema.

BIBLIOGRAFÍA

1. ALEGSA, Leandro. *Definición de Query* [en línea]. <http://www.alegsa.com.ar/Dic/query.php>. [Consulta: 15 de agosto de 2017].

2. DI PIERRO, Massimo. *Chapter 6: The database abstraction layer.* [en línea]. <http://www.web2py.com/books/default/chapter/29/06/the-database-abstraction-layer>. [Consulta: 1 de agosto de 2017].

3. Escuela de Ciencias y Sistemas (ECYS-FIUSAC). [en línea]. <https://wikiversidad.wikispaces.com/Escuela+de+Ciencias+y+Sistemas+(ECYS-FIUSAC)>. [Consulta: 20 de junio de 2017].

4. GARZAS, Javier. *¿Bases de datos NoSQL o Bases de datos SQL? ¿Tiene sentido en "nuestro mundo" usar bases de datos NoSQL?* [en línea]. <http://www.javiergarzas.com/2013/06/bases-de-datos-nosql.html>. [Consulta: 5 de agosto de 2017].

5. ROUSE, Margaret. *Framework*. [en línea]. <http://searchdatacenter.techtarget.com/es/definicion/Framework>. [Consulta: 15 de julio de 2017].

6. Venemedia. *Definiciòn de Backup.* [en línea]. <http://conceptodefinicion.de/backup/>. [Consulta: 15 de julio de 2017].

APÉNDICES

Apéndice 1. **Tabla de configuración de copias de seguridad**

Campo	Tipo	Descripción
id	int	Identificador único para cada registro.
fecha_proximo	date	Fecha en que se realiza la próxima copia de seguridad.
tipo	char	Tipo de copia de seguridad que se realiza. Puede ser: • U (Completo único) • I (Incremental semanal)
estado	char	Si la configuración está activa o inactiva. Puede ser: • A (Activa) • I (Inactiva)
ruta	varchar	Ruta donde se almacenan las copias de seguridad.
cantidad	int	Número de copias de seguridad que se almacenan.
servidor	varchar	Ip del servidor donde se almacenan las copias de seguridad.
puerto	int	Puerto del servidor de almacenamiento en el cual está habilitado SSH.

Continuación del apéndice 1.

usuario	varchar	Usuario del servidor de almacenamiento para iniciar sesión vía SSH.
clave	varchar	Contraseña del usuario del servidor de almacenamiento.

Fuente: elaboración propia.

Apéndice 2. **Tabla de histórico de copias de seguridad**

Campo	Tipo	Descripción
id	int	Identificador único para cada registro.
fecha	date	Fecha en que se realiza la copia de seguridad.
tipo	char	Tipo de copia de seguridad. Puede ser: • C (Completo) • I (Incremental)
tipoconf	char	Tipo de configuración de copia de seguridad. Puede ser: • U (Completo único) • I (Incremental semanal)
servidor	varchar	Ip del servidor donde se encuentra almacenada la copia de seguridad.

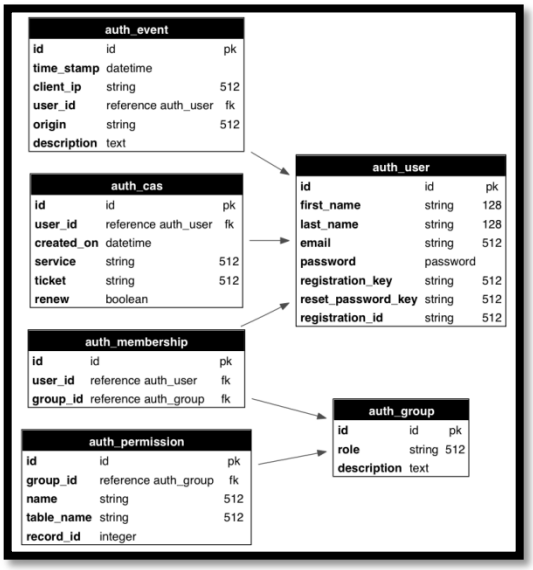
Continuación del apéndice 2.

puerto	int	Puerto de conexión SSH del servidor donde se encuentra almacenada la copia de seguridad.
usuario	varchar	Usuario del servidor que almacena la copia de seguridad para iniciar sesión vía SSH.
clave	varchar	Contraseña del usuario del servidor de almacenamiento.
ruta	varchar	Ruta donde se encuentra almacenada la copia de seguridad en el servidor de almacenamiento.
nombre	varchar	Nombre de la copia de seguridad.
estado	char	Si el proceso de creación de *backup* fue exitoso o no. Puede ser: • E (Exitoso) • F (Fallido)
tamano	varchar	Tamaño en bytes que ocupa la copia de seguridad.
procesolog	varchar	Log del proceso de creación de copias de seguridad.

Fuente: elaboración propia.

ANEXO

Anexo 1. **Diagrama entidad relación, sistema de autorización de web2py**



Fuente: Web2py. Diagrama ER Auth.
http://web2py.com/books/default/image/29/schema_auth.png. Consulta: 10 de agosto de 2017.