UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Desarrollo de un Sistema de Costos Web

INFORME DE ACTIVIDADES PROFESIONALES

Que para obtener el título de

Ingeniero en Computación

P R E S E N T A

Marco Antonio de Alba Lugo

ASESOR DE INFORME

M.I. Norma Elva Chávez Rodríguez

Ciudad Universitaria, Cd. Mx., Septiembre 2016

Agradecimientos

A mi padre, **Marco,** por haberme forjado como la persona que soy hoy en día. Por inculcarme todos los valores y hábitos que me definen como ser humano .Por el gran esfuerzo de darme una educación digna. Por nunca dejarme bajar los brazos en los momentos difíciles que atravesé como estudiante, como hijo y como profesionista. Por su incalculable apoyo en todos los momentos de mi vida. Por su amor, comprensión y por confiar en mí y en mis sueños. Pero sobre todo por acompañarme en mi camino con el inmenso esfuerzo de asumir el rol de padre y madre por todos estos años.

A mi madre, **Maru**, mi ángel guardián, que me dio todo su amor apoyo y fuerza para seguir adelante. Por ser mi confidente y mi motor en los momentos más difíciles. Por ese recuerdo y ese ejemplo de lucha, perseverancia, carácter, éxito y amor que sembró en mí. Y por estar a mi lado en todos los días de mi vida.

A mi hermana, **Maru**, por ser mi compañera de vida, por que juntos hemos librado todos los obstáculos que se nos han puesto en el camino. Por motivarme a ser un mejor ser humano.

A mis tíos, **Arturo**, **Maricela**, **Carolina**, **Ix**, por el incalculable apoyo que me han brindado como ser humano y como estudiante. Por ser mi inspiración como profesionista. Por todo su amor, consejos y ánimo para conseguir este título.

A **mis primos**, mis personas favoritas. Por ser una inspiración más en mi vida, porque sus logros siempre me impulsaron para seguir adelante. Por ser como mis hermanos y enseñarme a perseguir los sueños.

A **UNICA**, a mi tan odiada y querida **FI**, a mi amada **UNAM**, por el enorme esfuerzo de crear profesionales competentes y seres humanos con una misión y visión ética, humanista y de excelencia. Por abrirme sus puertas y darme ese orgullo de ser parte de la mejor universidad de México.

GRACIAS.

Índice general.

Contenido

Índice de figuras

INTRODUCCIÓN

Las actividades que se pueden desarrollar en un área dedicada a la innovación e investigación tecnológica dentro de los sistemas de computación son bastas. Con el apremiante cambio en el hardware y software, una empresa que desee mantenerse a la vanguardia requiere de cambios continuos en su infraestructura. Sin embargo estos cambios involucran costos importantes, por lo que es necesario conocer el mercado y saber cuál proveedor puede ofrecernos las mejores herramientas con el costo más accesible. En una empresa con tanto contenido digital los proveedores de servicios de CDN, storage y cloud computing son muy importantes para mantener un esquema de negocio rentable.

Dentro de las áreas más importantes en este ámbito se encuentra Azteca Internet que provee de contenido a todos los sitios web de Azteca, en donde converge todo el contenido digital. La demanda de contenido en los sitios es tan alta que es imposible contar con una infraestructura propia capaz de soportar y abastecer todos los servicios. Se requeriría de un centro de datos muy grande, lo cual implica un espacio físico muy amplio, costos de mantenimiento, una renta de ancho de banda y servicios de seguridad adicionales. Es por ello que se optó por contratar los servicios de Kaltura, una plataforma de software libre, pionera en ofrecer servicios de ingesta de video. Kaltura provee de una infraestructura capaz de abastecer la demanda de todos los sitios de Azteca, manejando, organizando y distribuyendo el contenido. Desde luego, dicho servicio conlleva un costo y nosotros como proveedores de dicho servicio requerimos de una herramienta capaz de distribuir el costo entre todos nuestros clientes, las producciones.

Mediante el API que expone Kaltura es posible calcular los gastos que genera cada uno de los videos que se reproducen en los sitios web y de esta forma conocer el monto que gasta cada una de las producciones.

CAPÍTULO I

Descripción de la empresa.

1.1 Historia de TV AZTECA

TV Azteca, es uno de los dos mayores productores de contenido para televisión en español en el mundo. Opera dos redes de televisión con cobertura nacional en México: Azteca 13, que se orienta a toda la familia; Azteca 7, que se enfoca en audiencias jóvenes de ingresos medios y altos, cubriendo 97% y 95% de los telehogares mexicanos, respectivamente. De igual forma opera Proyecto 40, un canal que reúne las personalidades más notables de México en los ámbitos cultural, social, económico y político. Azteca es propietaria también de Azteca América, cadena de televisión enfocada en el mercado hispano de los Estados Unidos con presencia en más de 70 plazas que abarcan más de 90% de los hogares hispanoamericanos en EE.UU., (Grupo Salinas)

La compañía es propietaria de Monarcas y Atlas, equipos de primera división de la Federación Mexicana de Futbol y opera Azteca Internet, un portal de internet que ofrece diversos contenidos de éxito.

1.2 Misión y visión

Las compañías de Grupo Salinas tienen una visión, misión, y valores en común. Dichos valores son sólidos compromisos con los empleados, accionistas, clientes, socios y las comunidades donde operamos. Son ideales que representan quienes somos, lo que hacemos y guían nuestro comportamiento como una corporación responsable. (Salinas)

VALORES:

- Honestidad
- Ejecución
- Inteligencia
- Excelencia
- Generosidad
- Enfoque en el cliente
- Trabajo en equipo
- Aprendizaje
- Rápido y simple

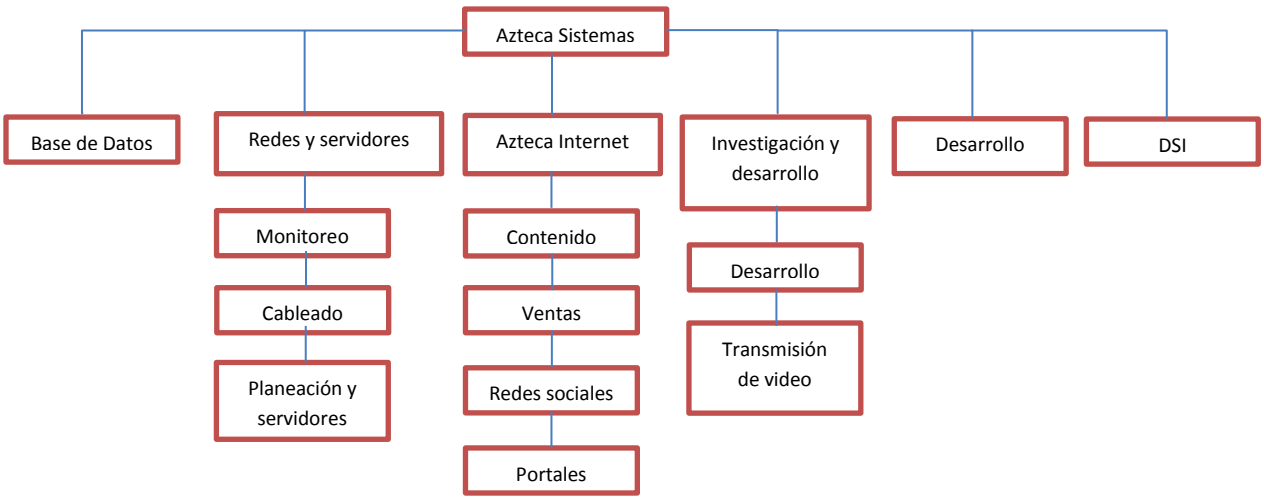
MISIÓN:

Crear valor para nuestros accionistas al producir y distribuir el mejor contenido para televisión en español en el mundo.

VISIÓN:

Ser la mejor televisión de habla hispana del mundo dedicada a entretener, formar e informar a la sociedad, sustentada en nuestro código de valores.

1.3 Organigrama



CAPÍTULO II

Puesto de trabajo

2.1 Investigación y desarrollo tecnológico

Mi participación dentro de Tv Azteca fue dentro del área de Ingeniería en sistemas, específicamente en el departamento de investigación y desarrollo tecnológico, implementando, desarrollando y manteniendo servicios que daban soporte a aplicaciones y sitios pertenecientes a Azteca Web, Azteca Deportes y Azteca Internet.

Tv Azteca se encuentra en una etapa de cambios en los que se realiza el mayor esfuerzo por mantenerse a la vanguardia tecnológica, con estos cambios las actividades que desarrollaba en mi puesto fueron amplias. Como objetivo principal, la investigación e implementación de nuevas tecnologías y su interconexión con los sitios, bases de datos y CMS, *Content Manager System* existentes.

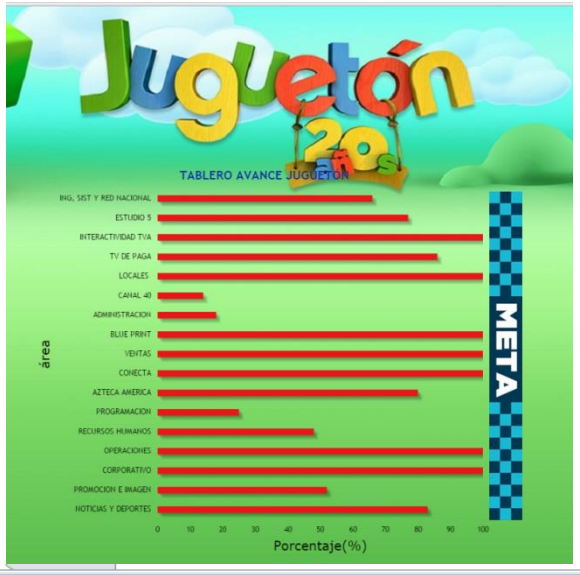
Fue necesario el desarrollo de sistemas para personal interno con el fin de que todas las producciones (Azteca Trece, Azteca Siete, Azteca Deportes, P40, etc.) tuvieran control sobre sus gastos de transmisión de video por internet, el manejo e implementación de algoritmos para el correcto y oportuno borrado de contenido de los sitios. Los mayores retos convergieron en la investigación y evaluación de proveedores con servicios IaaS, *Infrastructure as a Service* y PaaS, *Plataform as a Service* en la nube para llevar a todo el mundo el contenido de Azteca de una forma más rápida con mayor eficiencia y seguridad. Trabajé con la ingesta de video en los sitios de la empresa así como su almacenamiento y distribución.

Gran parte de mi trabajo como empleado radicó en la interacción con usuarios, que al final del día son los que nos nutren como empresa televisiva, fue muy importante para la televisora mantenerse en contacto con toda la comunidad de nuestros televidentes, con el completo auge de las redes sociales (Twitter y Facebook) tuve la necesidad de desarrollar aplicaciones basadas en el API, *Application Programming Interface* de dichas redes generando una gran experiencia de retroalimentación con todos nuestros cibernautas y televidentes. Se desarrollaron

aplicativos que mejoraron la experiencia del televidente, interactuando directamente con la producción desde cabina de Azteca Deportes y Box Azteca para el correcto conteo de golpes acertados por los contrincantes en una pelea de Box.

Figura 1. Recuento de votos utilizando Twitter API

Para el Juguetón (Campaña de Fundación Azteca que convoca a la sociedad civil, empresas e instituciones para que donen juguetes para niños en condiciones vulnerabilidad) se desarrolló dentro del portal interno una pantalla mediante la cual fue posible conocer el avance diario de juguetes donados por cada una de las áreas. Se motivó la competencia y mantuvimos informados a todos los empleados de su aportación como equipo de trabajo a esta noble causa. Desarrollado en java mediante la lectura y escritura de archivos Excel.



	A	B	C	D	E	F	G	H	I	J	K	L	M
1	ÁREAS	META	09-dic	10-dic	11-dic	12-dic	15-dic	16-dic	17-dic	18-dic	19-dic	20-dic	21-dic
2	PROGRAMACIÓN	291	4	4	4	4	35	53	55	75			
3	AZTECA AMÉRICA	100	0	0	0	0	0	0	0	80			
4	CORPORATIVO	100	0	0	0	0	0	0	0	190			
5	VENTAS	200	0	0	0	0	0	0	0	300			
6	NOTICIAS Y DEPORTES	300	0	0	0	0	38	66	66	249			
7	ING, SIST Y RED NACIONAL	800	0	0	20	20	59	244	244	530			
8	LOCALES	96	0	0	0	0	0	167	167	223			
9	CANAL 40	1000	0	0	0	0	0	129	129	140			
10	TV DE PAGA	107	0	0	0	0	0	50	50	93			
11	ADMINISTRACIÓN	100	0	0	0	0	2	3	3	18			
12	INTERACTIVIDAD TVA	40	0	0	0	0	10	32	32	41			
13	CONECTA	60	0	0	0	0	0	0	0	62			
14	PROMOCION E IMAGEN	40	0	0	0	0	0	0	0	21			
15	OPERACIONES	500	0	0	0	0	100	365	365	1600			
16	RECURSOS HUMANOS	100	0	2	2	2	15	22	22	48			
17	ESTUDIO 5	132	0	0	102	102	102	102	102	102			
18	BLUE PRINT	20	0	0	0	0	0	0	0	35			
19													

Figura 2. Aplicación Juguetón

Enfocadas en el desarrollo web, fue indispensable el conocimiento de tecnologías para el desarrollo de páginas web estáticas y dinámicas dedicadas a la venta de servicios por Internet. Tecnologías como Java, CSS, HTML 5, PHP, JQuery, JavaScript, JSON , XML, Bootstrap, etc.

Figura 3. Página HTML estática

Para el desarrollo de los aplicativos se utilizó el manejo de distintos lenguajes de programación y *frameworks* como Java, JSP, Spring, PrimeFaces, JSF, PHP, Python y la instalación de servidores Linux con distribuciones como Centos, Fedora y Ubuntu. El levantamiento de servicios como Tomcat, Apache y Glassfish. Y el manejo de DBMS entre los cuales se encuentran Oracle y MySql.

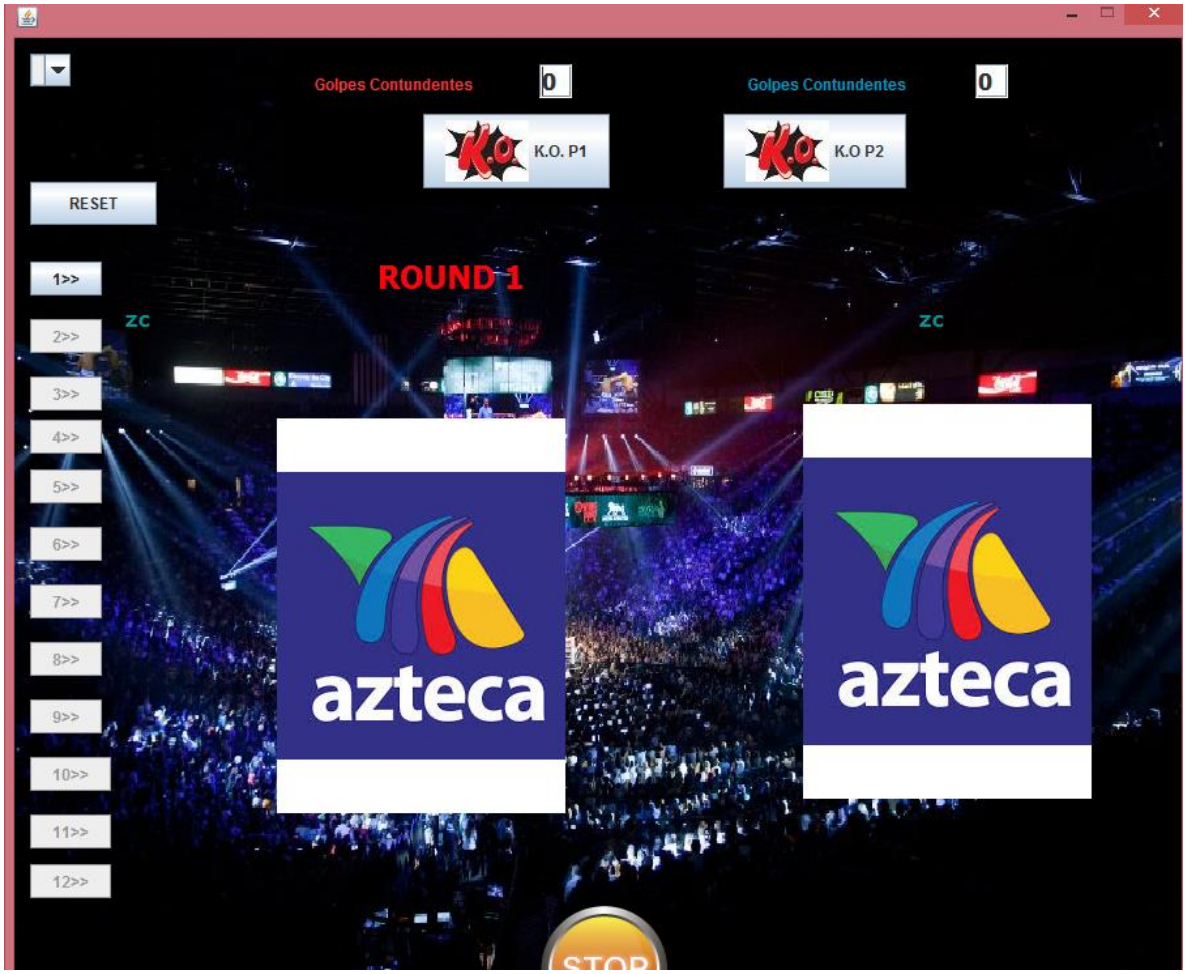


Figura 4. Aplicación Java. Conteo de golpes box con JoyStick analógico

Entre las mejores tecnologías que se lograron integrar al flujo de trabajo con desarrollo en la nube se encuentran proveedores de servicios como Kaltura, Amazon AWS, Adobe Experience Manager, Docker, OwnCloud, Twitter API, Facebook API, Microsft Azure, etc.

Un aspecto importante de mis aportaciones como profesionista fue el poder crear y liderar equipos de trabajo. Fue mi deber y responsabilidad reclutar y evaluar personal capaz y comprometido con los objetivos de Tv Azreca, integrarlos para construir un equipo eficiente y eficaz.

CAPÍTULO III

Proyecto desarrollado

3.1 SCWEB Sistema de Costos Web

La empresa genera contenido diario para todos sus sitios web y el contenido multimedia (imágenes y videos) es el más importante para mantener los sitios con un alto margen de visitantes. Existen equipos de trabajo dedicados a obtener, generar y cargar contenido para el funcionamiento de los sitios. Personal dedicado a la edición de video y audio e ingenieros en computación dedicados a la correcta carga y distribución del material. Para poder almacenar y distribuir dicho contenido a lo largo de todo el mundo se buscó el apoyo en una plataforma de video de código abierto llamada Kaltura.

Kaltura es una compañía de software que desarrolló la primera plataforma de código abierto que permite la gestión de video. Es líder en los mercados de OTT TV (*Over The Top TV*), OVP (*Online Video Plataform*), EdVP (*Education Video Plataform*) y EVP (*Enterprise Video Plataform*). Kaltura es utilizado alrededor del mundo por miles de compañías dedicadas a la distribución de contenido, proveedores de servicios e instituciones educativas, llegando así a millones de espectadores (Kaltura).

El desarrollo de este sistema surge a partir de la necesidad de poder vender este servicio a las distintas producciones contando con una herramienta capaz de distribuir los costos de Kaltura en partes proporcionales a su consumo, ya que anteriormente se dividía el costo total por partes iguales resultando un gasto excesivo en algunas de las producciones. Se cuentan con seis cuentas distintas de Kaltura el total de los videos asciende a más de 80 mil. Los costos se reportan mensualmente y es necesario conocer el gasto generado por cada uno de los videos, cada video corresponde a un programa, cada programa a una fábrica de negocio y cada fábrica a una unidad de negocio.

Para realizar métricas de consumo sobre el servicio de Kaltura, se tomaron en cuenta indicadores que permitieron hacer el cálculo del costo por cada uno de los

videos y con ello el gasto de cada producción. Los indicadores que ayudaron en este proceso fueron: la duración del video en segundos, el tamaño o peso del video en GB, el tiempo visto, que se refiere al tiempo que se vio el video en Internet durante un periodo de tiempo. Así conociendo el total de segundos vistos se puede obtener un precio por ancho de banda consumido y con el tamaño del video se puede saber cuánto almacenamiento está ocupando en la nube y por lo tanto cuanto es el precio que se debía de pagar.

Kaltura ofrece a todos sus clientes una plataforma gráfica en la cual se puede observar, manipular, editar, clasificar, pre visualizar, borrar, cargar, activar y desactivar todo el contenido que se coloca en la cuenta. Ofrece graficas en tiempo real del consumo de ancho de banda, minutos vistos, el número de veces que se reprodujeron los videos, cuantos videos existen en la cuenta, cuantos se han borrado, etc. Debido a la gran cantidad de contenido generado diariamente en la empresa, esta herramienta gráfica no resultó útil. Resultaría demasiado tardado hacer el recuento de toda la información video por video, juntarla y dispersar los costos seria poco práctico y demasiado tardado.

La solución más efectiva fue utilizar el API de Kaltura, el cual ofrece conexión a todos sus servicios con los lenguajes de programación PHP, .NET, Ruby, Adobe Flex, Javascript, Python y Java. Debido a la experiencia trabajando con Java, se eligió este API como motor de conexión a los servicios de Kaltura.

3.2 Kaltura

El API de Kaltura es código abierto, se puede encontrar y consultar por cualquier persona desde su página web (http://www.kaltura.com/api_v3 /testmeDoc/index.php), no obstante, para poder hacer uso es necesario contar con una suscripción al servicio. Una vez teniendo acceso a su plataforma, Kaltura expone las credenciales necesarias para poderse conectar a sus servicios desde Java

Los elementos necesarios que solicita el API para lograr una autenticación exitosa son:

Partner ID, un dato de tipo entero que se refiere a un identificador que le permite saber a Kaltura cuál es la cuenta a la que se está intentando accesar.

Admin_secret, una cadena de caracteres que representa una llave privada y nos ayuda a generar tokens de autenticación para crear sesiones. Se expone desde el KMC, *Kaltura Management Console*. Permite autenticarse como administradores de la cuenta y tener acceso a ejecutar todos los comandos del API.

User_secret, al igual que el Admin_secret, esta cadena de caracteres representa una llave privada, con la diferencia que limita algunos de los comandos, en su mayoría de los casos esta llave es suficiente para poder ejecutar los comandos más indispensables del API.

Con las credenciales listas, el siguiente paso consistió en crear un cliente, mediante el cual pude accesar a los métodos que necesitamos.

```
// Deportes
private static final String USER_SECRET = "o9ujjnhh89jsnaueuoajdnd984y474yy4";
private static final String ADMIN_SECRET = "P009ijjdu74hai398aiu487usgu7656g";
private static final int PARTNER_ID = 7645431;
```

Figura 5.Codigo Java. Credenciales Kaltura

```
private KalturaClient getKalturaClient(int partnerId, String secret, boolean isAdmin) throws KalturaApiException
    // set a new configuration object
    KalturaConfiguration config = new KalturaConfiguration();
    config.setPartnerId(partnerId);
    config.setEndpoint("http://www.kaltura.com");


    // get a new client object using our configuration
    KalturaClient client = new KalturaClient(config);

    // start a new session (admin/user) and recieve the ks (kaltura session) string
    String userId = "user's name";
    KalturaSessionType sessionType = (isAdmin ? KalturaSessionType.ADMIN : KalturaSessionType.USER);
    String ks = client.getSessionService().start(secret, userId, sessionType);
    // System.out.print(ks);
    // set the kaltura client to use the recieved ks as default for all future operations
    client.setSessionId(ks);

    // return the configured client
    return client;
}
```

Figura 6. Código Java. Generar cliente Kaltura

3.3 Proceso de recopilación de datos (Scheduler)

Para hacer un sistema eficiente se separó la parte visual de la parte operativa. Así tenemos un sistema web que únicamente se encarga de mostrar información al usuario y un sistema automático, independiente a la vista, que se ejecuta en el servidor, mediante el cual se hace la conexión a Kaltura, extrae la información y enriquece la base de datos.

Este proceso se llamó Scheduler, debido a que se comporta como un planificador, el código trabaja con el Quartz de Java y es el encargado de hacer toda la recopilación de datos. Se compone de hilos en paralelo que ejecutan clases llamadas Jobs. La clase Scheduler de Java permitió programar cronológicamente los Jobs con la creación de crones, cada cron invoca a un job y cada uno realiza una tarea específica. La ventaja de utilizar crones es que permitió establecer la fecha y hora en la que se debían ejecutar los Jobs, pudiendo tener un proceso mensual automático.

Al arrancar, el sistema toma de la base de datos las credenciales de todas las cuentas y de esta forma tiene acceso al contenido de todas ellas, posteriormente ejecuta los Jobs en la siguiente secuencia:

1- Actualizar Categorías

La estructura dentro de Kaltura está basada en categorías, así todos los videos que pertenecen a un programa pueden ser alojados en una categoría especifica que lleva el nombre de dicho programa.

Figura 7. Categorías

En ocasiones es necesario crear categorías nuevas para alojar contenido. Para que no hubiese categorías sin cargar en el sistema, este proceso se encarga de ir a Kaltura a verificar si todas las categorías existen en la base de datos, de no ser así descarga su nombre y su id para alojarlos en la base de datos.

2- Categorizar videos

Existe gente responsable de categorizar los videos al pasar por el CMS, sin embargo esto no siempre sucedía. Por lo cual quedaban videos sin categoría. El conflicto que generaba esto es que no había producción a la cual se le pudiera cargar el costo de ese video.

El API de Kaltura ofrece el servicio de categorización de videos.

Este proceso busca en Kaltura todos los videos que no tengan ninguna categoría asignada y los asigna en una categoría llamada

"SIN_CATEGORÍA", así pudimos identificar los videos y solicitar que se categorizaran.

3- Actualizar entrys

La base de datos se va cargando de información cada vez que el proceso Scheduler es ejecutado, lo cual ocurre una vez al mes. La tarea de este proceso es identificar los nuevos videos, aquellos que se hayan cargado durante el mes, de esta forma se mantuvo la base de datos actualizada.

4- Consumo Azteca, Noticias, Deportes y P40

Es el proceso más largo debido a la gran cantidad de contenido que se aloja en las cuentas.

Su tarea es recorrer toda la tabla 'kaltura_entry' la cual tiene el identificador de cada video. Toda esta información se descarga en una lista y elemento por elemento se consulta en Kaltura cuanto fue la cantidad de segundos que se consumió el video en el mes actual, tras obtener la información se guarda en la tabla 'kaltura_reporte'.

5- Crear subreporte

El propósito de este proceso es realizar las consultas del SCWEB de una forma más rápida. Al ser tanta la información que se expone, se requirió de un proceso que realizara las consultas más tardadas, para que el sistema tuviera un tiempo de respuesta mas corto.

6- Borrado

Para el ahorro en el costo de Kaltura es indispensable reducir al máximo el número de videos almacenados en la plataforma, teniendo en mente este propósito se desarrolló un algoritmo de borrado que permitió reducir el número de videos poco productivos, conservando sólo los que reditúen su gasto.

El algoritmo consta de 3 filtros constituidos a partir de los distintos comportamientos que presentan los videos a lo largo del tiempo. Para que este algoritmo funcione se requiere un mínimo de 6 muestras por video. Las cuales se obtienen mensualmente.

FILTRO 1.

Todos los videos de los cuales la suma de sus n>6 muestras sean igual a cero, serán borrados.

Esto quiere decir que si un video no fue visto ni una sola vez en el periodo de n>6 muestras (m), no es eficiente y será eliminado de la plataforma.

$$\sum m = 0 \quad \text{BORRADO}$$

FILTRO 2.

Densidad a la derecha. Se encuentra el pico más alto del muestreo. Si a partir del pico hay una cantidad del 60% de ceros al lado derecho del pico, el video será eliminado.

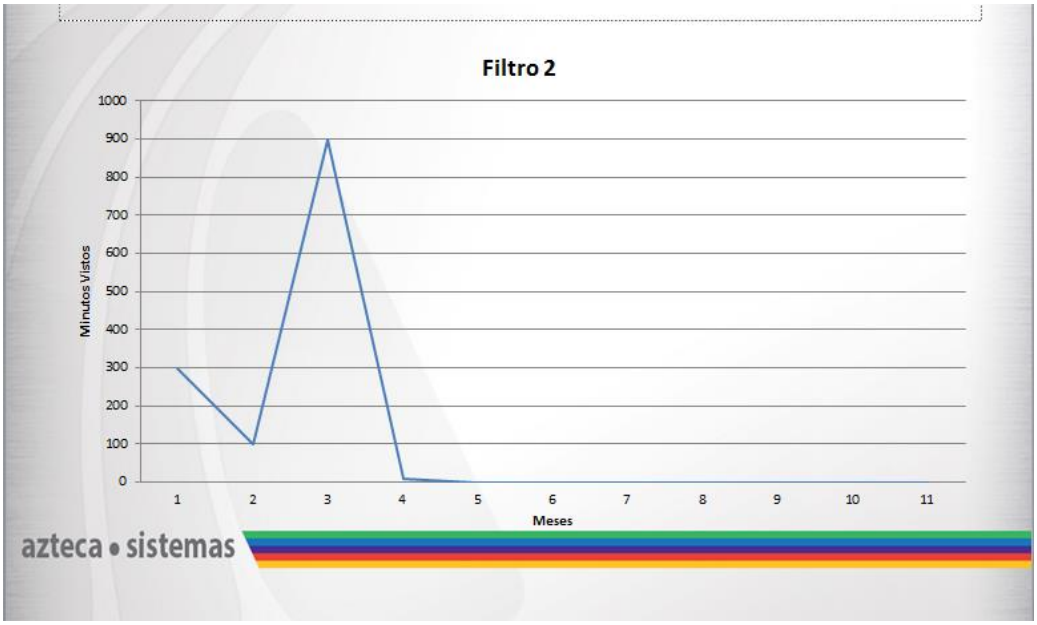


Figura 8. Gráfica Filtro de borrado 2

FILTRO 3.

Amortiguamiento. Se busca el pico más alto en el muestreo de datos, si la cantidad de caídas a partir del pico son mayores a (n-1)/4 donde n = número de muestras, se busca un sub pico y se realiza la misma fórmula hasta encontrar una amortiguación menor al 60% de área bajo la curva entre picos. En este caso el video será eliminado. Por el contrario, se tiene un numero de subidas mayores a (n-1)/4 que tiendan a la derecha del periodo, significa que el video se recuperó.

Figura 9. Gráfica de borrado Filtro 3

Para la solución de dicho problema se utilizó estructuras de datos como listas, mapas o diccionarios y pilas.

3.4 Módulos del sistema web

Para el desarrollo del sistema se buscó trabajar con herramientas de fácil acceso, una sencilla implementación y sobre todo que fueran de código abierto.

La base del desarrollo se basa en el esquema Modelo-Vista-Controlador, dicho modelo permite separar los datos de la lógica del negocio y gestionar con mayor facilidad la interfaz del usuario.

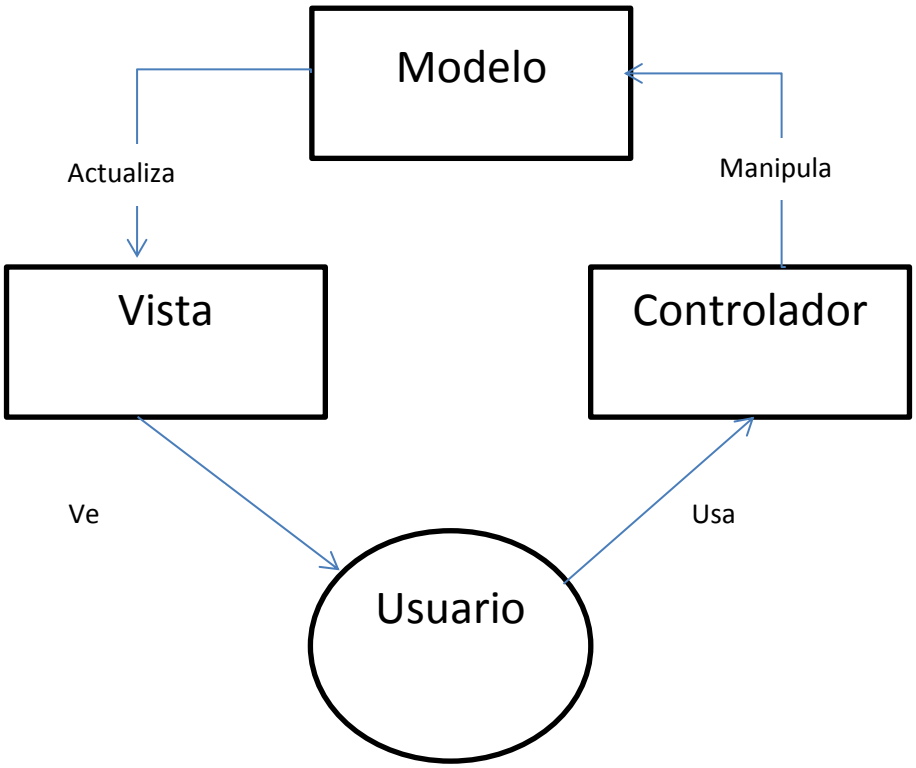


Figura 10. Modelo Vista Controlador

El uso de Frameworks facilita este proceso. Spring aparece en los tres componentes del MVC proporcionando control y administración sobre los diferentes módulos del sistema, generando modelos de datos, ayudando a incluir capas de seguridad e interactuando con la vista mediante controles de acceso.

Con la integración de JSF y el componente de Prime Faces la programación de la vista, como lo son tablas, menú, cajas de texto, botones, etc se torna más amigable, estos frameworks proporcionan una amplia librería de etiquetas personalizadas para dibujar componentes UI dentro de una página HTML, ofrecen amplia documentación y una extensa gama de ejemplos en sus páginas oficiales.

Es posible lograr la integración de todos estos frameworks mediante Netbeans un IDE, *Integrated Development Environment* que soporta el desarrollo en Java y ofrece una amplia variedad de módulos para desarrollos tanto móviles, de plataforma y web, contando con la capacidad de levantar servicios por ejemplo a bases de datos, servicios web, servidores e incluso servicios de integración con Maven y ambientes cloud.

Todo este conjunto de herramientas permiten modular la vista del usuario final, controlar los procesos que al usuario no le interesan pero son vitales para el correcto funcionamiento del sistema, proporcionan seguridad de información, versionar para revertir cambios, respaldado con una de los sistemas manejadores de bases de datos más robustos y utilizados y de código libre como lo es MySql. Es por ello que el sistema aquí descrito se realizó bajo estas tecnologías con la confianza de la estabilidad, seguridad, funcionamiento y escalabilidad.

A continuación muestro la estructura de los proyectos involucrados para el funcionamiento de este sistema donde se puede ver el modelo MVC:

- **kalturaScheduler**: El Scheduler, es el encargado de realizar los Jobs y todas las consultas hacia el API de Kaltura, en la estructura del proyecto se puede ver el paquete “com.azteca” donde se guardan todas las clases relacionadas al control, modelado, jobs y servicios relacionados a la aplicación. También se observa el paquete de clases “com.kaltura”, aquí se contienen todas las clases que expone el API de Kaltura para realizar las consultas a sus servicios.

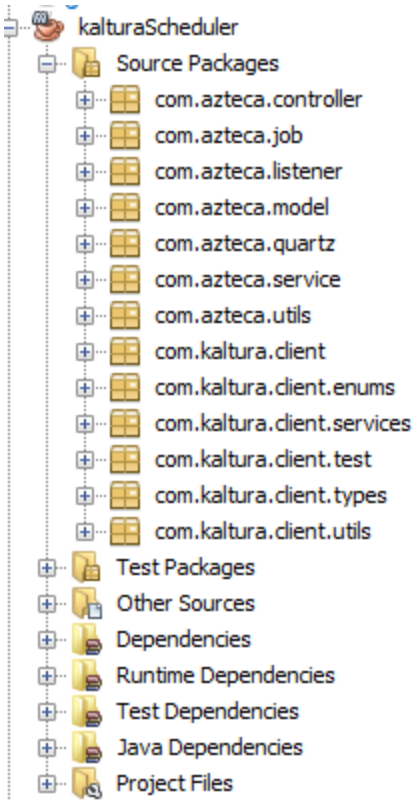


Figura 11. Estructura kalturaScheduler

- **Persistance:** Se llama "persistencia" de los objetos a su capacidad para guardarse y recuperarse desde un medio de almacenamiento. La función de este proyecto como lo indica la descripción es ingresar y extraer los datos de la base de datos. La estructura de este proyecto muestra dos paquetes importantes: "com.azteca.persistance.entities" y "com.azteca.persistance.repository".

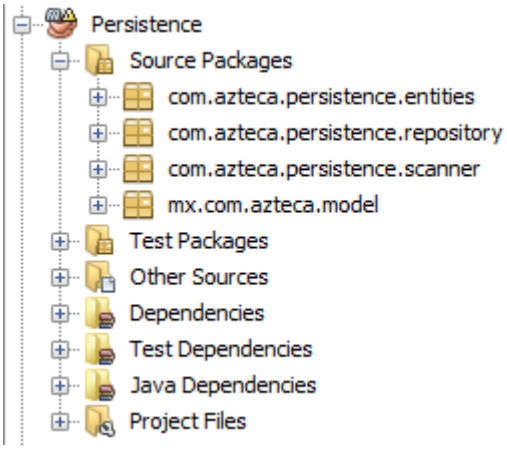


Figura 12. Estructura Persistance

El primero contiene el mapeo de datos de todas las entidades de la base de datos, de esta forma pude trabajar con los datos como si fueran objetos de Java. Tanto Hibernate como JPA ayudaron a realizar la persistencia de datos. Un ejemplo de codificación de una entidad en Java se muestra a continuación:

```
@Entity
@Table(name = "KALTURA_ENTRY")
public class KalturaEntry implements Serializable{

    @Id
    @NotNull
    @Column(name = "ENTRY_ID")
    private String entryId;

    @Basic(optional = false)
    @NotNull
    @Size(min = 1, max = 45)
    @Column(name = "NOMBRE")
    private String nombre;

    @Basic(optional = false)
    @NotNull
    @Column(name = "FECHA_CREACION")
    @Temporal(javax.persistence.TemporalType.DATE)
    private Date fechaCreacion;

    @Basic(optional = false)
    @NotNull
    @Column(name = "TAMANIO")
    private Integer tamanio;

    @Basic(optional = false)
    @NotNull
    @Column(name = "DURACION")
    private Integer duracion;
```

Figura 13. Clase Kaltura_Entry

El segundo paquete contiene las interfaces Java que se implementan en el "Scheduler" y en el "SCWEB", cuando se requiere hacer consultas a la base de datos. A continuación se muestra parte de código de la codificación de la interface "KalturaEntryRepository" que hace uso de la entidad descrita anteriormente "KalturaEntry".

```
public interface KalturaEntryRepository extends PagingAndSortingRepository<KalturaEntry, Integer> {

    List<KalturaEntry> findAll();

    List<KalturaEntry> findByNombre(String nombre);

    List<KalturaEntry> findByEntryId(String nombre);

    @Query("select u from  KalturaEntry u where (u.fechaCreacion between :primera and  :segunda) AND u.kalturaPrograma=:tercera ")
    List<KalturaEntry> findCortesByFecha(@Param("primera") Date fechaInicial, @Param("segunda") Date fechaFinal, @Param("tercera")

//    @Query(" select k  from KalturaEntry k  JOIN k.kalturaPrograma p JOIN p.kalturaFabrica f JOIN f.kalturaUnidad u JOIN u.kaltur
//    List<KalturaEntry> findByNombreCuenta(@Param("primera") int nombreCuenta);

    @Query(" select k  from KalturaEntry k where k.kalturaPrograma=:primera and k.borrado= false")
    List<KalturaEntry> findByProgramaAndBorrado(@Param("primera") KalturaPrograma kalturaPrograma);

    @Query(" select k  from KalturaEntry k where  k.borrado= false and k.pendiente= false")
    List<KalturaEntry> findByNoborrado();

    @Query(" select k  from KalturaEntry k where k.kalturaPrograma=:primera")
    List<KalturaEntry> findByPrograma(@Param("primera") KalturaPrograma kalturaPrograma);
}
```

Figura 14. Clase KalturaEntryRepository

- **KalturaWebApp:** Es la parte visual de todo el sistema. Son las pantallas que visualiza el usuario final, la estructura del proyecto se compone de carpetas importantes:

    - **WEB-INF:** Este directorio contiene todos los recursos relacionados con la aplicación web que no deben servirse al cliente. Este directorio no forma parte del documento público, por lo que ninguno de los ficheros que contenga va a poder ser enviado directamente a través del servidor web. En este directorio se coloca el archivo web.xml, donde se establece la configuración de la aplicación web. También contiene los archivos "applicationContext.xml" y "securityContext.xml" dos archivos indispensables para la configuración de Spring.

- **Source Packages:** Contiene las clases Java referentes al control, servicios e implementación de la aplicación.

- **Project Files:** El archivo más importante que guarda este directorio es el "POM.xml", *Project Object Model* archivo utilizado por Maven para describir el proyecto a construir, sus dependencias de otros módulos y componentes externos, así como el orden de construcción de los elementos. Cumple con objetivos predefinidos para realizar tareas como la compilación del código y su empaquetado.

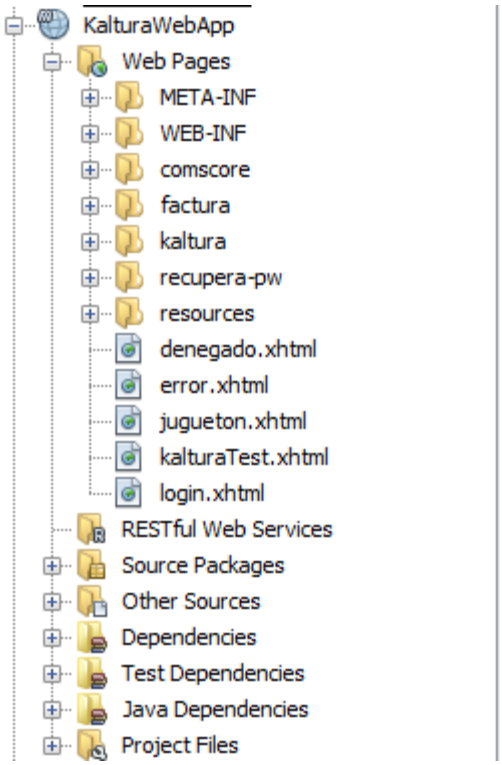


Figura 15. Estructura KalturaWebApp

3.5 Flujo de pantallas

Para garantizar que el contenido mostrado en el sistema fuera únicamente accesible por usuarios permitidos se desarrolló un módulo de *login*, mediante el cual el usuario introduce su nombre y contraseña. Este módulo, así como la navegación en el sitio está controlado con un módulo de Spring llamado 'Spring Security'.

Spring Security provee servicios de seguridad para aplicaciones Java, siendo el control de acceso (Autenticación y Autorización), el área de mayor importancia. Con el proceso de autenticación valide que el usuario sea quien dice ser, si existe en la base de datos y si proporciona la contraseña adecuada. La Autorización se refiere al proceso de conocer si algún usuario tiene permiso de realizar ciertas acciones dentro del sistema.

Figura 16. Módulo Login SCWEB

```
<security:http auto-config="true" use-expressions="true">
    <security:intercept-url pattern="/service/**" access="permitAll()" />
    <security:intercept-url pattern="/kalturaTest.xhtml" access="permitAll()" />
    <security:intercept-url pattern="/repositorio/**" access="isAuthenticated()" />
    <security:intercept-url pattern="/factura/**" access="hasRole('Administrador')" />
    <security:intercept-url pattern="/recupera-pw/index.xhtml" access="permitAll()" />
    <security:intercept-url pattern="/kaltura/index.xhtml" access="isAuthenticated()" />
<!--        <security:intercept-url pattern="/comscore/index.xhtml" access="permitAll()" />-->
    <security:intercept-url pattern="/kalturaTest.xhtml" access="permitAll()" />
    <security:form-login login-page="/login.xhtml" default-target-url="/kaltura/index.xhtml" authentication-failure-url=
    <security:logout invalidate-session="true"/>
    <security:access-denied-handler error-page="/denegado.xhtml"/>
    <security:session-management>
        <security:concurrency-control max-sessions="1" expired-url="/login.xhtml?faces-redirect=true"/>
    </security:session-management>
</security:http>
```

Figura 17. Spring SecurityContext

Una vez autenticado se tiene acceso a la pantalla que permite al usuario generar reportes de consumo mensual. Consta de un filtro por fecha, en la cual se especifica la fecha inicial y final del reporte, un check box por si deseamos ver los videos que no se consumieron, dos cajas de texto en las cuales se especifica el costo en dólares por segundo y por GB y un módulo en donde podemos decidir si queremos ver un reporte en tablas o en gráficas.

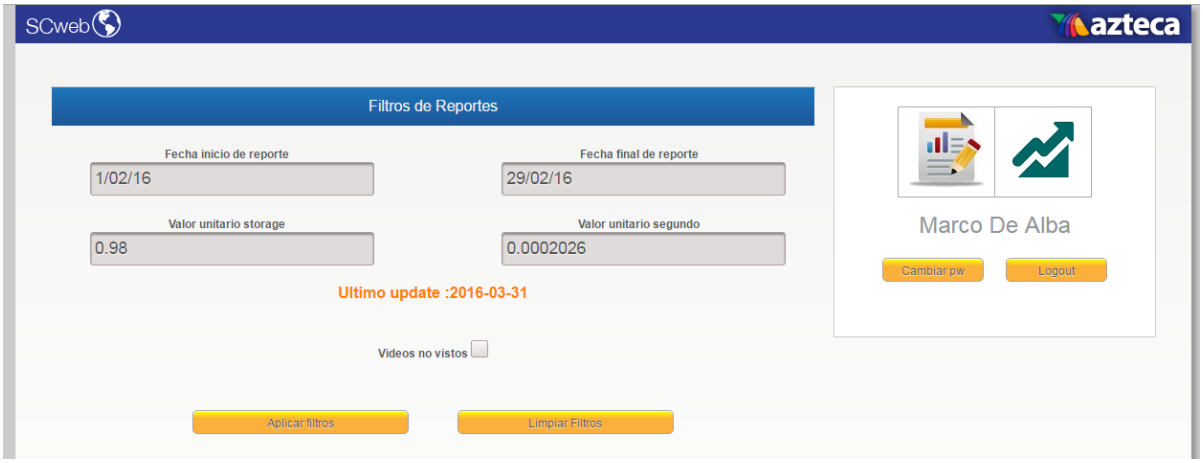


Figura 18. SCWEB. Módulo de reportes

El reporte en tablas nos arroja la siguiente información:

- Un reporte a nivel global en donde podemos ver el total de videos que hay en todas las cuentas de Kaltura, cuánto tiempo se consumieron todos estos videos y el peso total en GB.

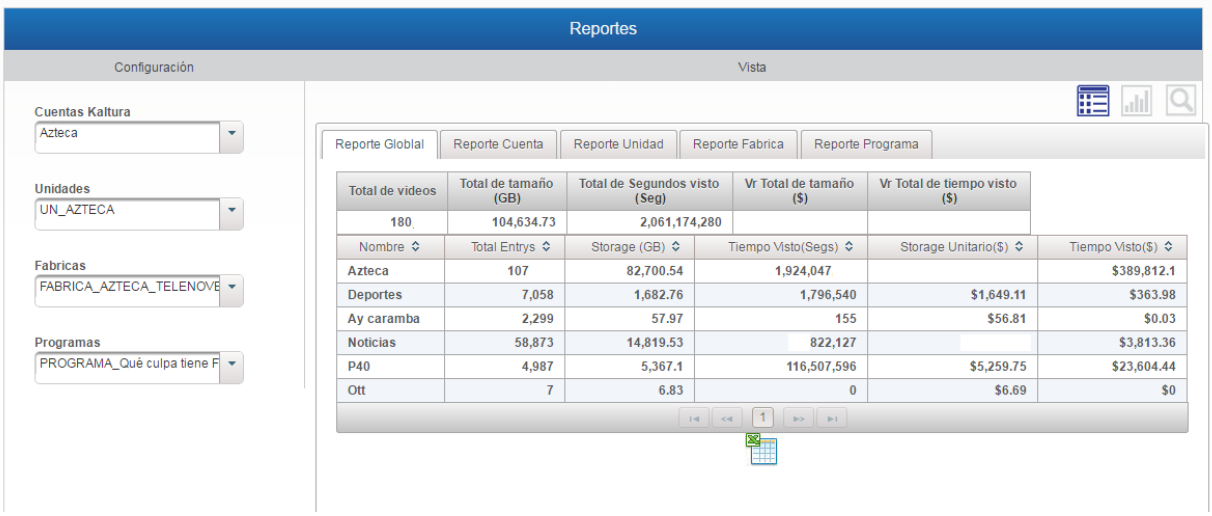


Figura 19. SCWEB. Módulo Reporte Global

- Una tabla más detallada del consumo individual por cuenta.
- Y cuatro select box mediante las cuales podemos seleccionar cual contenido deseamos consultar.

La granularidad del contenido se basa en la siguiente jerarquía:

Cuenta → Unidad de Negocio → Fábrica → Programa

Los select box que vemos en la imagen anterior nos permiten navegar a través de este esquema, dando la capacidad de obtener reportes en todos los niveles granulares.

Al seleccionar la pestaña “Reporte unidad” podemos obtener un detalle del consumo de todas las fábricas que conforman la Unidad. Así como una suma total.

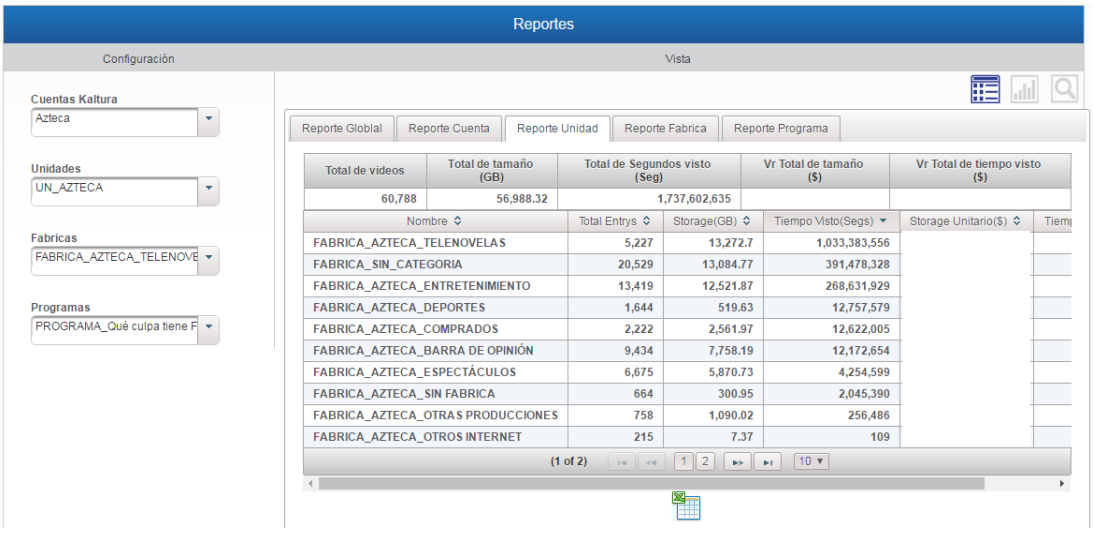


Figura 20. SCWEB. Reporte por unidad

De igual manera, para conocer a detalle todos los videos que conforman un programa en especial, basta con seleccionar la pestaña “Reporte Programa”. Este es el nivel más granular del reporte, se muestran todos los videos con nombre, tamaño, duración, fecha de creación, segundos vistos en el mes y el costo que generó cada uno de ellos.

Figura 21. SCWEB. Reporte por programa

Con la implementación de Prime Faces es posible utilizar el componente "*lineChart"* para generar gráficas. Esto me dio la posibilidad de generar un módulo de reportes gráfico que expone la misma información que el módulo de reportes por tablas, pero aquí la información se grafica para tener una idea más clara del comportamiento del consumo de videos.

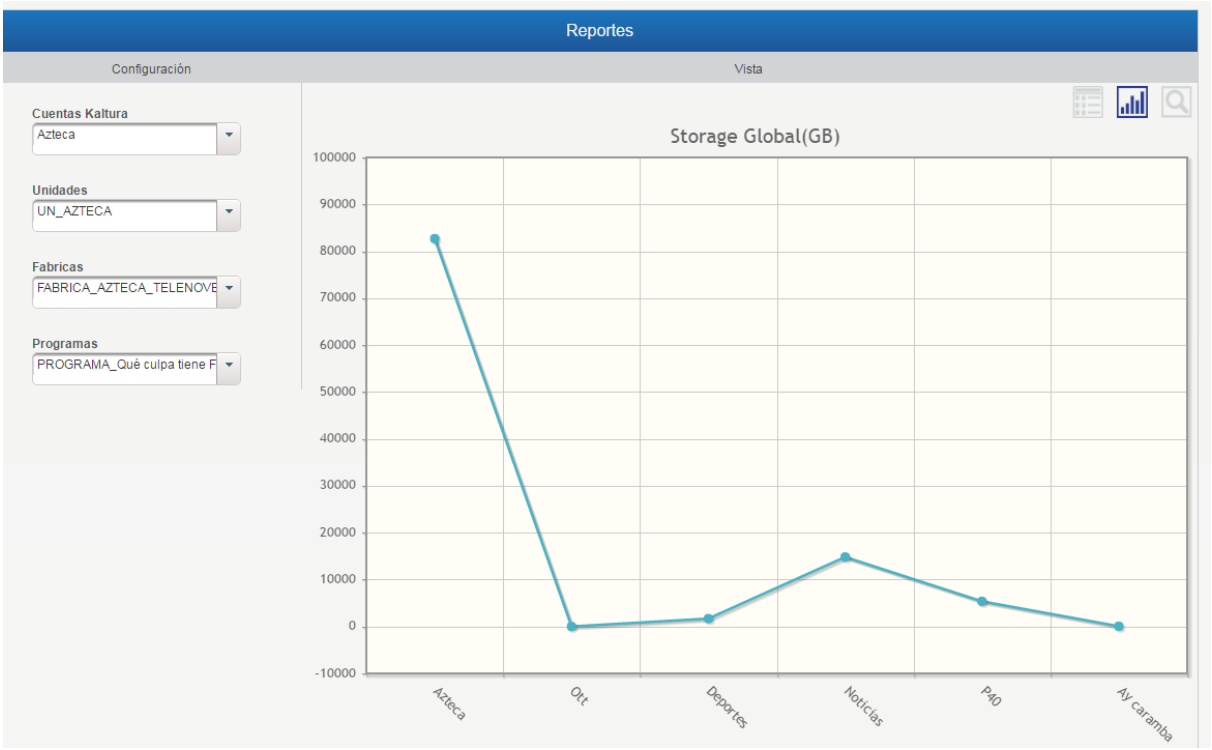


Figura 22. SCWEB. Gráfica Storage

3.6 Gestor de Bases de Datos MySQL.

Un gestor de bases de datos (DBMS), integra los componentes como un lenguaje de definición de datos (DDL), un lenguaje de manipulación de datos (DML) y un lenguaje de consulta (QL), además puede incluir una interfaz gráfica para el usuario (GUI) con el fin de realizar las operaciones más recurrentes hacia la base de datos.

En la actualidad existen una gran cantidad de DBMS, algunos gratuitos y muchos más con algún costo y respaldados por grandes compañías. Azteca aloja algunos de sus sistemas en servidores respaldados por el DBMS de MySQL siendo esta razón la principal para que nuestro sistema se base en esta herramienta.

El servidor MySQL es el servicio mysqld, que puede recibir solicitudes de clientes locales o remotos a través TCP/IP, sockets o pipes en forma de ficheros locales a la máquina en que se está ejecutando. En la distribución se incluye un cliente llamado mysql-client. (Rafael Camps Paré, 2005)

La administración y seguridad de MySQL está diseñada sobre un esquema de usuarios y privilegios. Los usuarios deben ser creados por el administrador con sus respectivos privilegios y restricciones. Es el administrador quien decide si los nombres de los usuarios de MySQL se corresponden o no a los del sistema operativo. (Rafael Camps Paré, 2005)

Un aspecto muy destacable es su condición de open source, que hace que su utilización sea gratuita e incluso se pueda modificar con total libertad, pudiendo descargar su código fuente. Esto ha favorecido muy positivamente en su desarrollo y continuas actualizaciones, para hacer de MySQL una de las herramientas más utilizadas por los programadores orientados a Internet.

A continuación se muestra el esquema Entidad-Relación, omitiendo los campos en las tablas por privacidad de la información.

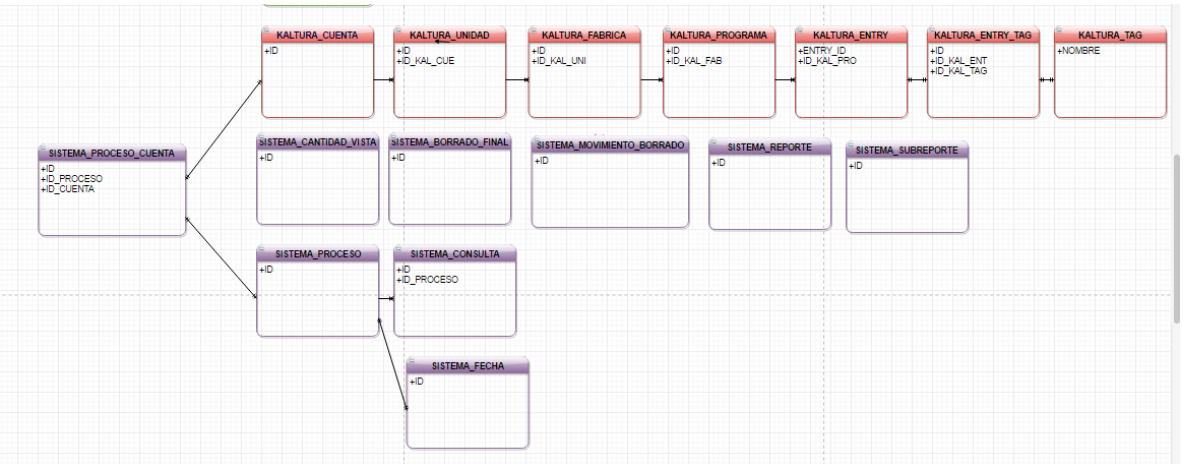


Figura 23. Diagrama ER

La base de datos se integró con 16 tablas, algunas relacionadas entre sí mediante llaves foráneas. Las consultas ocupadas utilizan el Lenguaje de Manipulación de datos:

- SELECT: Consulta de elementos.
- INSERT: Inserción de nuevos elementos.
- DELETE: Borrado de elementos existentes.
- UPDATE: Actualización de elementos existentes.

Mensualmente se agregan al sistema más de 200,000 registros por lo cual el uso e implementación de índices resulto crucial. Un índice contiene claves generadas a partir de una o varias columnas de la tabla o la vista. Dichas claves están almacenadas en una estructura (árbol b) que permite que MySQL busque de forma rápida y eficiente la fila o filas asociadas a los valores de cada clave. Sin un índice, MySQL tiene que iniciar con el primer registro y leer a través de toda la

tabla para encontrar los registros relevantes. Aún en tablas pequeñas, de unos 1000 registros, es por lo menos 100 veces más rápido leer los datos usando un índice, que haciendo una lectura secuencial.

Herramientas y requerimientos para el desarrollo:

Lenguaje de Programación:	Java 7
Web Server:	Tomcat 7
DBMS:	MySQL
IDE:	NetBeans 7.3
FrameWorks:	Spring, JSF, PrimeFaces, Hibernate, Maven
SO	Linux, Windows

Herramientas y requerimientos del servidor para la implementación:

Distribución Linux:	Centos 6
Espacio en disco:	8GB RAM
Java 7	
Servidor DB	MySQL
Puertos activos	80, 8080, 22, 21
Web Server:	Tomcat 7

3.7 Instalación y despliegue de la aplicación

Para realizar la instalación de la aplicación SCWEB en un servidor Centos fue necesario tener previamente instalados la máquina virtual de Java y el servidor de aplicaciones Apache Tomcat. Para realizar la instalación de la aplicación debió generarse el archivo "WAR" que a su vez es requerido para realizar el despliegue.

A continuación se dividirá la instalación en tres pasos los cuales constan de la instalación de la máquina virtual Java, la instalación del servidor de aplicaciones Apache Tomcat y la instalación y configuración del aplicativo SCWEB.

3.7.1 Instalación de la máquina virtual Java

La instalación de la máquina virtual Java varía de acuerdo al sistema operativo que se maneja en el servidor, sin embargo, para este caso la versión que se utilizó fue CentOS por lo que los siguientes pasos son exclusivos del mismo.

El primer paso para instalar la máquina virtual de Java es conocer las versiones disponibles que existen y que han sido liberadas por SUN, para ello es necesario ejecutar el siguiente comando :

 # yum search java | grep -i --color JDK

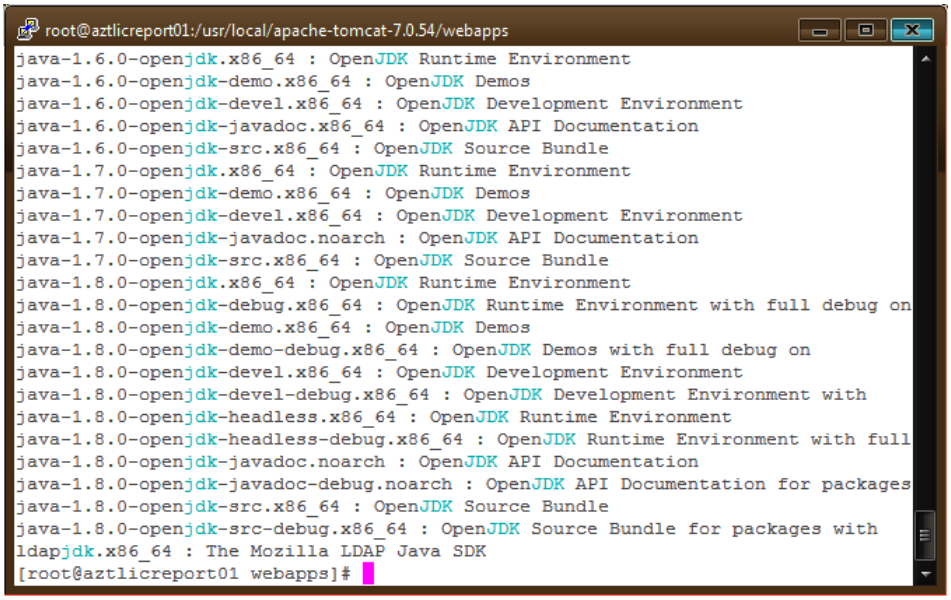


Figura 24. Salida del comando yum search java | grep -i --color JDK

Para esta aplicación utilice la versión 1.7

Una vez seleccionada la versión a utilizar se ejecuta el siguiente comando:

 # yum install java-1.7.0-openjdk java-1.7.0-openjdk-devel

Ahora se debe de buscar la carpeta de instalación para configurar las variables de entorno mediante el siguiente comando:

 # ls -l /usr/lib/jvm/

Al ejecutarlo aparecerá entre otra información la versión del JDK que se está utilizando, en este caso utilizaremos la parte que nos interesa es la versión del JDK.

```
lrwxrwxrwx. 1 root root   38 Apr 19 13:06 jre-1.7.0-openjdk.x86_64 -> java-1.7.0
-openjdk-1.7.0.99.x86_64/jre
```

Figura 25. Comando ls -l /usr/lib/jvm/

Como último paso se configuró la variable de entorno JAVA_HOME para poder realizar la ejecución de comandos propios de Java sin la necesidad de ir a la carpeta de instalación.

 # export JAVA_HOME="usr/lib/jvm/jre-1.7.0-openjdk.x86_64"

Al final del comando en la última diagonal utilizamos el resultado del paso anterior.

3.7.2 Instalación del servidor de aplicaciones Apache Tomcat

Para realizar la instalación del servidor de aplicaciones Apache Tomcat es necesario tener ya instalada la máquina virtual de Java. Lo primero que se debe de realizar es situarse en una carpeta donde se pueda realizar la descarga de la aplicación, como ejemplo podemos utilizar la siguiente carpeta :

 # cd /tmp

El siguiente paso consta de realizar la descarga de la versión del servidor Apache Tomcat de acuerdo a la versión de Java que se haya instalado, en la siguiente página se puede obtener mayor información al respecto:

http://tomcat.apache.org/

Para realizar la descarga se utilizó el siguiente comando:

#wgethttp://archive.apache.org/dist/tomcat/tomcat7/v7.0.68/bin/apache-tomcat-7.0.68.tar.gz

Una vez completada la descarga se deberá realizar la descompresión del archivo mediante el siguiente comando :

 # tar xzf apache-tomcat-7.0.68.tar.gz

Una vez realizado lo anterior deberemos reubicar la carpeta generada en el directorio donde deseamos que resida el servidor, a manera de ejemplo se utiliza el siguiente comando que contiene un directorio por defecto :

 # mv apache-tomcat-7.0.68 /user/local/tomcat7

3.7.3 Instalación y configuración de la aplicación SCWEB

Para realizar la instalación de la aplicación SCWEB se generó un archivo tipo WAR y se realizó el despliegue del mismo en el servidor de aplicaciones.

Generación del archivo tipo WAR

En el IDE de desarrollo que utilicé (NetBeans) se da clic derecho sobre la aplicación y se selecciona la opción "Clean and Build".

Una vez terminado el proceso nos indicará la ruta donde se generó el archivo tipo WAR el cual utilizaremos para la instalación.

```
Installing E:\Users\60048080\Documents\NetBeansProjects\KalturaSchedulerWeb3.2\scweb\target\KalturaWebApp~3.2.war
Writing tracking file E:\Users\60048080\.m2\repository\com\azteca\KalturaWebApp\3.2\_remote.repositories
Installing E:\Users\60048080\Documents\NetBeansProjects\KalturaSchedulerWeb3.2\scweb\pom.xml to E:\Users\60048080
Writing tracking file E:\Users\60048080\.m2\repository\com\azteca\KalturaWebApp\3.2\_remote.repositories
Installing com.azteca:KalturaWebApp/maven-metadata.xml to E:\Users\60048080\.m2\repository\com\azteca\KalturaWebA
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time: 02:38 min
Finished at: 2016-05-27T13:11:48-06:00
Final Memory: 33M/132M
------------------------------------------------------------------------
```

Figura 26. Clean and Build

Se copió el archivo WAR en el servidor donde se instaló la máquina virtual de Java y el servidor de aplicaciones Apache Tomcat. Dependiendo del sistema operativo en que se esté trabajando será la forma en que se realizará la transferencia del archivo, para sistemas operativos Windows se cuenta con aplicaciones con ambiente gráfico como lo es FileZilla donde se configuran las credenciales del servidor de destino y a partir de este se localizan las carpetas de origen y a la que se desea enviar.

En mi caso el SO utilizado fue Linux, el cual cuenta con comandos como lo son SCP para realizar el copiado de la información. A continuación se muestra un ejemplo:

scp -r SCWeb.war usuario@255.255.255.255:/tmp

Una vez se tenga el archivo en el servidor de destino es necesario copiarlo en la carpeta "webapps" que se encuentra dentro de los archivos donde está el servidor de aplicaciones, para ello utilizaremos el siguiente comando :

 # mv nombreArchivo.war rutaCarpetaWebapps
 # mv SCWeb.war /usr/local/ apache-tomcat-7.0.68/ webapps

Para finalizar se deberá de levantar (iniciar) el servidor de aplicaciones, en este caso es necesario dirigirse a la carpeta bin y ejecutar el comando " startup.sh".

#cd /usr/local/ apache-tomcat-7.0.68/bin
#./ startup.sh

Al ejecutar este comando se realizará del despliegue de la aplicación, y una vez terminado el mismo se podrá tener acceso al mismo escribiendo en el navegador la dirección IP del servidor, el puerto (en este caso por defecto es 8080), y el nombre de nuestra aplicación.

http://255.255.255.255:8080/SCWeb/

Configuración de la aplicación SCWeb

En caso de que se requiera que la URL de la aplicación en vez de mostrar el nombre de la aplicación simplemente muestra la dirección IP del servidor (http://255.255.255.255:8080/) se deberá de respaldar y reemplazar el archivo ROOT.war del servidor de aplicaciones que se encuentra en la carpeta "webapps" y renombrar la aplicación SCWeb.war por el nombre de ROOT.war, a continuación se describe una serie de comandos como ejemplo:

 # cd /usr/local/ apache-tomcat-7.0.68/ webapps

 # mkdir Respaldo

 # mv ROOT.war Respaldo

 # mv SCWeb.war ROOT.war

Dichos comando deben ser ejecutados una vez que el servidor está detenido, para esto debemos ir a la carpeta "bin" del servidor de aplicaciones y ejecutar el comando "shutdown.sh ".

 # cd /usr/local/ apache-tomcat-7.0.68/ bin

 # ./ shutdown.sh

En caso de que se requiera enmascarar el puerto de la aplicación es necesario cambiar el mismo dentro del archivo de configuración "server.xml", a continuación se muestran los comando utilizados para realizarlo.

 # cd/usr/local/ apache-tomcat-7.0.68/conf

 # vi server.xml

En este caso se utilizó el editor de textos vi, la parte que se desea modificar es la que se muestra en la siguiente imagen:

Figura 27. Archivo conf

Donde se encuentra la etiqueta de conector como se muestra en la imagen se tiene el puerto 80, es aquí donde deberá reemplazarse el puerto 8080 por el antes mencionado. Estos pasos deberán realizarse mientras es servidor de aplicación se encuentra detenido, por lo que se deberán emplear las instrucciones que se enunciaron anteriormente.

Conclusiones

Con el desarrollo de este sistema se logró apoyar al personal de costos a tener una herramienta segura, rápida y eficiente mediante la cual pudieron hacer estimaciones mucho más precisas de los gastos generados por cada programa. Los gastos pueden ser distribuidos de una forma más clara, anulando así el problema de dividir costos en proporciones iguales, cuando evidentemente existen producciones con programas que son más vistos que otros y que debieran pagar un porcentaje mayor del total de la factura.

Adicionalmente la herramienta otorga información muy valiosa para la venta de programas, conociendo la cantidad de tiempo que se ve cada programa es más sencillo poder venderlo a alguno de los tantos patrocinadores. A su vez sirve como un medidor de rating porque nos indica si un programa está siendo visto lo cual quiere decir que a la gente le gusta y que podemos seguir invirtiendo en él sin riesgos de perdida de dinero. Se logró reducir el costo de la factura, con los procesos implementados y el algoritmo de borrado es mucho más sencillo identificar los videos que no son vistos y que por lo tanto solo generan gastos en lugar de ingresos.

Al tener toda la información almacenada en una base de datos es posible generar consultas diferentes a las estipuladas en el sistema, con el uso de querys directamente en la base de datos se pude conocer, por ejemplo, el número total de nuevos videos subidos en el mes, cuales videos han sido borrados y mediante que filtro, el consumo de video en una fecha específica o simplemente cuales fueron los diez videos más vistos durante el mes.

El uso de Java y la implementación de algunos de sus frameworks facilitó el trabajo y logró crear un sistema objetivo, eficiente e intuitivo con la finalidad de que fuera sencillo de utilizar para los usuarios. Actualmente tiene un aproximado de treinta usuarios concurrentes y sus procesos se ejecutan automáticamente una vez al mes.

Entre los módulos que tenemos proyectados desarrollar a futuro para escalar esta aplicación, se evaluará integrar un módulo de facturas, el cual las genere en base al gasto de

cada producción, un módulo de gasto de CDN, *Contentet Delivery Network* e integración con Google Analytics para conocer el número de visitantes que tienen nuestros sitios en internet.

Bibliografía

Grupo Salinas. (s.f.). Recuperado el 2016 de Agosto de 02, de http://www.gruposalinas.com.mx/es/azteca

Kaltura. (s.f.). *Kaltura*. Recuperado el 29 de Marzo de 2016, de http://corp.kaltura.com/About-Kaltura

Rafael Camps Paré, L. A. (1 de 5 de 2005). Bases de datos. Barcelona, España.

Salinas, G. (s.f.). Recuperado el 20 de Agosto de 2016, de http://www.gruposalinas.com.mx/es/valores