UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

DESARROLLO DE HERRAMIENTA DE MONITOREO AUTOMÁTICO PARA UNA APLICACIÓN DE APOYO AL APRENDIZAJE: SKETCHPAD

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL EN COMPUTACIÓN

ALFONSO JAVIER CORNEJO CASTILLO

PROFESOR GUÍA:
NELSON BALOIAN TATARYAN.

MIEMBROS DE LA COMISIÓN:
NELSON BALOIAN TATARYAN.
JORGE PÉREZ ROJAS.
JUAN ÁLVAREZ RUBIO.

SANTIAGO DE CHILE
2015

RESUMEN DE LA MEMORIA PARA OPTAR AL TÍTULO DE: Ingeniero Civil en Computación
POR: Alfonso Javier Cornejo Castillo
FECHA: 19/04/2015
PROFESOR GUÍA: Nelson Baloian Tataryan

DESARROLLO DE HERRAMIENTA DE MONITOREO AUTOMÁTICO PARA UNA APLICACIÓN DE APOYO AL APRENDIZAJE: SKETCHPAD

Sketchpad es una herramienta de apoyo a la educación presencial enfocada al trabajo colaborativo. Consiste en una aplicación web diseñada para ser usada en dispositivos tablet que permite a sus usuarios trabajar colaborativamente en diapositivas, dibujando y escribiendo texto sobre ellas de forma síncrona y en tiempo real.

Adicionalmente, Sketchpad tiene integrado un sistema de analíticas que muestra estadísticas de uso de la aplicación por sus usuarios y las relaciona con sus desempeños académicos con el propósito de entregar feedback respecto a situaciones que pudiesen estar correlacionadas con los resultados académicos. Estas estadísticas son presentadas con interfaces diseñadas para encontrar patrones de comportamiento de forma visual y rápida.

El objetivo del presente trabajo es describir el desarrollo e implementación de Sketchpad, explicando el propósito general de la aplicación y la motivación para construirla, las funcionalidades que ofrece, la arquitectura y diseño con la que fue implementada y la metodología de desarrollo usada.

Se concluye este trabajo con el feedback recibido al realizar una prueba de usuario con alumnos de tercer año de la carrera de Ingeniería Civil en Computación en la que se usó la aplicación para resolver gráficamente problemas de lógica.

La aplicación logra promover la interactividad entre usuarios y permite el desarrollo de clases colaborativas. Las herramientas analíticas desarrolladas entregan información valiosa a los usuarios respecto al contexto y el desarrollo de clases. Concluimos que Sketchpad se construyó exitosamente de acuerdo al propósito para el que fue diseñado.

Agradecimientos

A mis profesores guías por alentarme, a mi familia por siempre apoyarme y a mis amigos de TF2 por las infinitas noches de compañía.

Tabla de Contenido

I. Introducción

En la actualidad, el método de enseñanza usado más comunmente es el de las clases unilaterales, donde un profesor presenta material a sus alumnos siguiendo un orden secuencial y las interacciones entre los participantes son mínimas [1]. Esta baja interacción dificulta que el profesor sepa con certeza el grado de avance de los alumnos y que se adapte a lo que la clase requiere para que el aprendizaje sea óptimo [2].

Con la aparición y popularización de dispositivos electrónicos en la sala de clases se han desarrollado aplicaciones que apoyan a la docencia y el aprendizaje con mayor interactividad, utilizando estos dispositivos para implementar funcionalidades que introducen prácticas pedagógicas que favorecen la interacción alumno-profesor y alumno-alumno. Una de las teorías educativas más aceptadas hoy en día es la del constructivismo, la cuál entre otras cosas sostiene que el conocimiento se construye mejor de manera colaborativa. Luego el uso de dispositivos electrónicos y el apropiado software de apoyo a la docencia se alinean con la enseñanza constructivista [6].

Adicionalmente, el uso de software en la educación presencial gatilla una transición del manejo de la información de forma efímera a datos explícitos; mientras que escuchar una cátedra no deja huellas o registros físicos, cada click, cada interacción entre personas y cada vista de página en internet deja una marca, marcas que cada día más aplicaciones y dispositivos capturan. Estas huellas generadas por usuarios pueden dejar valiosa información que puede ayudar a entender qué es lo que realmente ocurre en el proceso de aprendizaje de un alumno, información que puede ser tomada por profesores o administrativos y puede ser usada para encontrar formas de mejorar el aprendizaje [3].

Recopilar esta información generada por alumnos para analizarla es lo que se conoce como Learning Analytics, concepto que según la Primera Conferencia Internacional de Learning Analytics and Knowledge se define como: “La medida, recolección, análisis y reporte de datos acerca de personas que aprenden y sus contextos, con el propósito de comprender y optimizar el aprendizaje y el entorno en el que ocurre” [4].

Por todo esto, en este trabajo de título se tomó el prototipo de una herramienta de apoyo a las clases presenciales llamada Sketchpad, el cual permite dibujar y diagramar contenido utilizando herramientas para dibujar trazos, texto e imágenes, y se completó y extendió para implementar interacción entre usuarios permitiendo aprender y trabajar colaborativamente.

El sistema desarrollado permite al profesor distribuir contenido a los alumnos sincronizadamente, con herramientas para modificarlo en tiempo real de modo de que todos los usuarios participantes de una clase ven las diapositivas y los cambios en ellas al mismo tiempo y pueden colaborar sobre ellas. Además, los alumnos pueden tomar diapositivas del profesor, crear copias y tomar apuntes encima de ellas, los que quedan guardados en una base de datos de la aplicación y quedan disponibles para ser vistos y utilizados posteriormente.

La aplicación se diseñó de modo que sus funcionalidades sean aprovechadas en dispositivos tablets. Particularmente, para nuestro contexto de trabajo la aplicación fue desarrollada para su uso en la Facultad de Economía y Negocios de la Universidad de Chile, en un curso piloto que se realizará en una sala dispuesta especialmente para este proyecto en donde se contará con un dispositivo iPad 3 para cada alumno (en una sala con capacidad de alrededor de 40 alumnos) y con 7 Televisores integrados con AppleTV para el uso de actividades grupales. Ambos dispositivos mencionados, iPads 3 y AppleTVs, soportan navegación web y el estándar HTML5, que es el estándar sobre el que está construido Sketchpad.

Adicionalmente, la aplicación registra todas las acciones que realizan los usuarios y ofrece interfaces para mostrar varios análisis del comportamiento de los usuarios relacionados con los niveles de actividad y colaboración. Entre los análisis desarrollados se encuentran, por ejemplo, la comparación del nivel de actividad de un alumno con el nivel de actividad promedio de su curso y el nivel de actividad de un alumno y su desempeño académico.

El objetivo buscado es el de permitir entregar información sobre un alumno en particular o sobre un curso completo de forma gráfica de modo que sea fácil identificar tendencias respecto a ciertos comportamientos o facilitar la detección temprana de situaciones que ayuden a los educadores o incluso a los mismos alumnos a tomar decisiones para mejorar su proceso educativo.

Se realizaron algunas pruebas de usuario cuando la aplicación aún estaba en desarrollo y los resultados obtenidos mostraron que la aplicación estimula e incentiva la participación colaborativa de los alumnos y ayuda a realizar prácticas constructivistas en la educación presencial.

1. Motivación

Un factor importante que puede afectar el aprendizaje de los alumnos es la atención. Los estudiantes que no se ven involucrados en las discusiones en clases, simplemente esperan que los otros estudiantes respondan las preguntas, lo que ocasiona que ellos pierdan la atención y con ello la oportunidad de razonar, concluir, equivocarse y analizar los errores propios como parte del proceso de aprendizaje [5]. Una forma de disminuir este fenómeno en los estudiantes a la hora de responder preguntas es utilizar un sistema que permita la participación colaborativa de usuarios a través de dispositivos digitales [6].

El modelo de aprendizaje colaborativo se basa en los supuestos constructivistas, en los cuales se visualiza el aprendizaje como una construcción del conocimiento que realiza el alumno al interactuar con el medio y en donde se le pone especial atención a las interacciones alumno-profesor y alumno-alumno para apoyar, estimular y evaluar el trabajo colaborativo del grupo. El uso de sistemas computacionales de apoyo a la enseñanza en la sala de clases ha mostrado ser beneficioso para los estudiantes dado que, entre otras cosas, aumenta la comunicación entre alumnos con el profesor y con sus pares [6] [7].

Sketchpad apoya el trabajo grupal y promueve la participación y colaboración entre miembros. De modo que los usuarios pueden aportar a las discusiones y los profesores pueden ayudar al aprendizaje estimulando, coordinando y sugiriendo caminos para el aprendizaje en tiempo real.

Por su parte, Learning Analytics [4] provee un nuevo modelo para mejorar la enseñanza, el aprendizaje, la eficiencia organizacional y las decisiones que se toman en la educación [3]. Basar decisiones en datos y evidencia parece ser el camino obvio a seguir y, de hecho, estudios indican que este tipo de decisiones logran significativas mejoras tanto organizacionalmente y en productividad [8] [11]. Se hacen innecesarias las teorías de comportamiento humano, no es importante entender por qué las personas hacen lo que hacen, lo importante es saber que lo hacen y que podemos saber exactamente qué es lo que hacen. Pero también, usar analytics requiere que se piense detenidamente sobre qué se necesita conocer y qué datos serán más probable que nos digan lo que queremos saber [9].

Sin duda, Learning Analytics tiene un rol significativo que jugar en el futuro de la educación superior. El crecimiento de técnicas de análisis y tecnologías en gobiernos y sectores empresarios reafirman esta tendencia. En la educación el valor de Analytics puede ser identificado claramente en cómo puede asistir a educadores en mejorar la

enseñanza y el aprendizaje, lo que permite mejorar la calidad y el valor de la experiencia del aprendizaje [3] [10].

2. Antecedentes

2.1. Colaboración

Estudios han demostrado que usar dispositivos digitales para tomar apuntes en clases puede ser usado efectivamente para promover la interacción, la evaluación y el feedback. Golub [18] por ejemplo propone que el buen uso de software para tomar y recopilar apuntes puede ayudar a obtener feedback de una clase de forma organizada y eficiente, tanto durante la clase como posterior a su término. También propone que usar esta metodología beneficia al proceso de evaluar el trabajo y las tareas de alumnos.

Al mismo tiempo, otros trabajos [12], [19] y [20] sostienen que el uso de dispositivos electrónicos en la sala de clases, como elemento activo de la clase misma, potencian y mejoran el nivel de atención y de interacción de los alumnos, fomentando la participación en clases.

Si consideramos la teoría del constructivismo, algunos estudios proponen que el uso de dispositivos móviles son ideales para practicar esta metodología educativa [6] [21]. Se ha concluido que estos dispositivos apoyan el trabajo colaborativo de grupos, promueven la interacción y la cooperación y además tienen la ventaja de que al ser portátiles permiten acercarse físicamente a las personas con las que se requiere interactuar.

2.2. Learning Analytics

Learning Analytics (LA) es la medición, recolección, análisis y reporte de datos acerca de estudiantes y sus contextos de enseñanza con el propósito de entender y optimizar el aprendizaje y el entorno en el que ocurre [22].

LA está fuertemente enfocado a mejorar los procesos de aprendizaje, sostiene la promesa de mejorar la eficiencia y efectividad de la enseñanza entregando a educadores, estudiantes y tomadores de decisiones con información relevante sobre actividades de clases y cursos [22]. El uso de LA en la educación es una tendencia que cada día más universidades e instituciones educacionales han ido adoptando.

En [10] se discuten enfoques que se pueden tomar para hacer Learning Analytics en distintos contextos y se compara cómo fueron presentados los resultados a los educadores y cómo estos valoraron dichos resultados. Se concluye que dependiendo de la forma o la visualización que se use para presentar los resultados, estos pueden ser percibidos por profesores con mayor o menor valor, pero que en general este tipo de análisis son apreciados por los educadores.

Debido a la masificación del uso de Learning Analytics, un grupo de investigadores de la Society for Learning Analytics Research propone en 2011 una plataforma abierta para integrar técnicas de Learning Analytics en distintos contextos [22]. Experimentos realizados que han seguido este estándar se reportan como exitosos casos de estudios en los que efectivamente se obtiene conocimiento sobre el proceso de aprendizaje, y reafirman el valor de Learning Analytics como una útil metodología para mejorar el aprendizaje [23] [24].

Aunque a primera vista pudiese parecer difícil recopilar datos relevantes para hacer un análisis para mejorar el proceso educativo en una aplicación en la que su proceso educativo consiste en dibujar trazos sobre una pizarra en blanco, estudios sugieren ciertos parámetros que pueden ser relevantes en la educación apoyada por dispositivos electrónicos.

En los estudios realizados en [15], [16] y [17] se proporcionó feedback a los educadores haciendo extensivos análisis de registros de datos presentados de forma visual. La herramienta utilizada (TADV) entregó importante feedback a los profesores tales como los peores desempeños o los errores más frecuentes. Este sistema registra extensivamente todas las actividades de los estudiantes en las actividades que se realizan día a día, detecta cuando los alumnos necesitan de asistencia y les informa a los profesores para que actúen al respecto.

Por su parte en otro trabajo [13] [14] se utilizó un enfoque en el que se presentó a los educadores representaciones visuales de estadísticas de uso de la herramienta utilizada CourseViz, permitiendo detectar patrones en sus alumnos sin usar algoritmos sofisticados de Data Mining. Este tipo de análisis ayudó a los educadores a explorar comportamientos cognitivos de los estudiantes usando esta herramienta digital.

Otra metodología estudiada [25] [26] realiza lo que se denomina como “Smart Learning Analytics”, donde en un proceso automatizado se presenta a los usuarios un ambiente de trabajo encapsulado en el IDE Eclipse [32] con una extensión que recolecta silenciosamente, es decir sin que el usuario lo note y sin necesidad de que intervenga, información del proceso de desarrollo de los usuarios (llamado EIDEE) y la envía a un

servidor especializado en la recolección y análisis de datos denominado Hackystats [27] [28]. Este servidor almacena en bruto toda la información recibida de los EIDEE clientes y luego se utilizan servicios que consumen estos datos para generar estadísticas que son visualizadas en una interfaz de usuario llamada MI-Dash. Esta interfaz está disponible para todos los usuarios del EIDEE y en ella se pueden ver gráficos con estadísticas relacionadas con el trabajo realizado y su efectividad al realizarlo.

2.3. Antecedentes del proyecto

El proyecto Sketchpad comenzó como una iniciativa de la Facultad de Economía y Negocios (FEN) de la Universidad de Chile en donde se buscó desarrollar una aplicación que aprovechara las ventajas de los navegadores web para desplegar en ellos una aplicación colaborativa de distribución de contenido.

Algunos de los requerimientos específicos de la aplicación que fueron definidos al comienzo del proyecto fueron los siguientes.

- Interfaz de usuario estar en inglés.
- Ser diseñada para ser usada en tablets iPad3 usando el navegador integrado Safari.
- Desarrollar la interfaz de usuario usando HTML5 y Javascript.
- Desarrollar el motor usando Java Empresarial (JEE) [33].

La metodología de desarrollo utilizada consistió en un enfoque en el que definiendo iteraciones cortas se fueron desarrollando partes específicas de la aplicación que mediante reuniones semanales con el cliente (FEN) eran evaluadas, discutidas y rediseñadas de ser necesario.

De esta forma, teniendo una comunicación frecuente con el cliente, se definieron objetivos semanales que aportaban al cumplimiento de la aplicación final.

3. Objetivos

3.1. Objetivo General

Desarrollar una aplicación web para uso en clases de apoyo al aprendizaje presencial que permita el trabajo colaborativo y la interacción entre usuarios.

Adicionalmente, se busca integrar el uso de prácticas de Learning Analytics en la aplicación para entregar información relevante al usuario respecto a su desempeño en clases y cómo éste pudiese estar relacionado con el resultado de sus evaluaciones.

3.2. Objetivos Específicos

- Extender y completar el prototipo de Sketchpad para que cumpla con las siguientes funcionalidades:
    1. Sincronización de diapositivas en tiempo real.
    2. Colaboración sobre diapositivas permitiendo a varios usuarios dibujar en una diapositiva al mismo tiempo.
    3. Implementación de trabajo en grupo durante la clase.
    4. Moderación de permisos para dibujar en diapositivas permitiendo al profesor que los alumnos interactúen con sus diapositivas de contenido durante el desarrollo de la clase.
- Implementar un Panel de instrumentos de Learning Analytics con estadísticas que permitan identificar rápida y visualmente situaciones que podrían afectar el desempeño de un estudiante en una asignatura. En particular este Dashboard debe ser capaz de mostrar a un usuario estadísticas tales cómo:
    1. Número de acciones realizadas en clases.
    2. Porcentaje relativo de acciones realizadas por el usuario comparado con sus pares.
    3. Mapa de compañeros con los que el usuario trabaja clase a clase.
    4. Y, relacionado con todo lo anterior, se espera poder relacionar estas estadísticas con el desempeño académico del usuario.

II. Descripción de la Aplicación

Sketchpad es una aplicación web creada para apoyar la educación presencial mediante la distribución de contenido en tiempo real y el trabajo colaborativo y grupal. Su funcionalidad principal consiste en la distribución de diapositivas sobre las que se puede dibujar usando herramientas predeterminadas y cuyos contenidos se mantienen sincronizados en todo momento con todos los usuarios participantes de una clase, de modo que si un usuario dibuja un nuevo trazo en la diapositiva, este nuevo contenido se distribuye a todos los usuarios permitiendo que todos vean el nuevo trazo en su propio dispositivo al mismo tiempo que el usuario autor del trazo. Esta funcionalidad de Sketchpad es la que se denomina "*Drawing App"*.

Además, la segunda funcionalidad de Sketchpad es la integración de un Analytics Dashboard que muestra estadísticas de uso de la aplicación por clase, tales como número de trazos dibujados, número de diapositivas creadas, número de interacciones con otros usuarios y otras estadísticas más complejas. Estas se presentan de forma visual en conjunto con los resultados de evaluaciones del curso con el propósito poder comparar fácilmente el desempeño académico de un estudiante con sus pares y encontrar tendencias respecto al uso de la aplicación y su correlación con los resultados académicos. Las visualizaciones del Analytics Dashboard junto con el proceso de extracción de los datos mostrados en él es lo que se denomina "*Learning Analytics*" (LA) en el proyecto.

El desarrollo de ambas funcionalidades son los principales objetivos de este trabajo de título. En esta sección se explicarán por separado ambas funcionalidades desde el punto de vista del usuario final.

1. Drawing App: Herramienta de aprendizaje colaborativo

Sketchpad permite crear diapositivas, dibujar en ellas a mano alzada o escribiendo texto con el teclado, y distribuirlas sincronizadamente y en tiempo real a todos los usuarios que la ven. Además, varios usuarios pueden dibujar en una diapositiva al mismo tiempo.

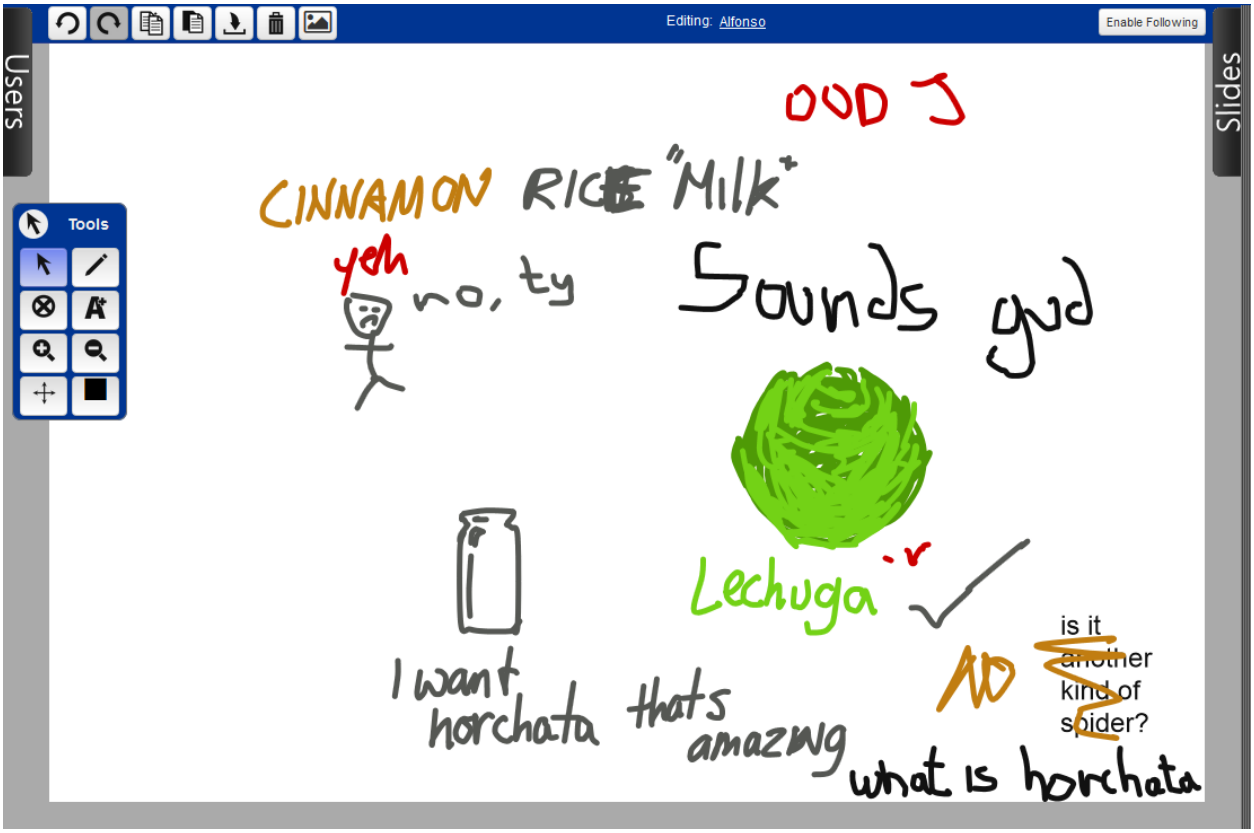


Figura 1. Ejemplo de diapositiva creada colaborativamente usando Sketchpad en la Drawing App.

1.1. Herramientas de Dibujo

Comenzaremos describiendo la funcionalidad central de Sketchpad, explicando la interfaz para dibujar en diapositivas.

Al crear una nueva diapositiva aparece completamente en blanco. Sobre una diapositiva se pueden dibujar trazos a mano alzada de distintos colores, escribir texto, o subir una imagen para que quede como fondo de la diapositiva. En la figura 2 se ven los botones de herramientas para dibujar sobre una diapositiva.

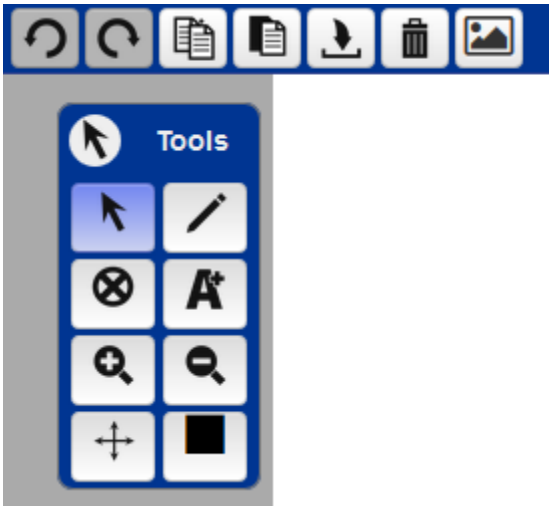


Figura 2. Herramientas de dibujo de Sketchpad.

1.1.1.  Herramienta de Dibujo

Dibuja trazos en la diapositiva. Arrastrar el puntero por la diapositiva dibuja una línea en ella.

1.1.2.  Herramienta de Texto

Escribir texto en la diapositiva. Haciendo click en la diapositiva hará aparecer un cuadro de texto en el que se puede escribir usando el teclado.

1.1.3.  Herramienta de Selección

Selecciona los elementos dibujados para hacer operaciones sobre ellos tales como mover, eliminar, copiar/pegar, cambiar color u otras.

1.1.4.  Eliminar

Elimina el o los elementos seleccionados de la diapositiva.

1.1.5.  Color

Selector de color. Cambia el color de el o los elementos seleccionados y define el color en el que se dibujaran los nuevos trazos y textos en la diapositiva.

1.1.6.  Copiar

Copia el o los elementos seleccionados a una variable temporal para luego ser duplicados en otra parte usando la función Pegar.

1.1.7.  Pegar

Pega el o los elementos que a los que se les había usado la función *Copiar* creando una copia de ellos en la diapositiva. Se pueden copiar-pegar elementos en una misma diapositiva o copiar elementos de una diapositiva a otra.

1.1.8.  Zoom-in

Acercar la vista de la diapositiva. Al hacer click en un punto en la diapositiva los elementos y fondo se verán ampliados y desplazados respecto al punto en el que se hizo click.

1.1.9.  Zoom-out

Aleja la vista de la diapositiva. Herramienta inversa del *Zoom-in*.

1.1.10.  Pan

Desplaza la vista de la diapositiva. Cuando la vista de la diapositiva se encuentra ampliada por usar la herramienta *Zoom-in*, esta herramienta permite desplazarse por la diapositiva arrastrando el cursor por la vista de la diapositiva.

1.1.11.  Deshacer

Deshace la última acción realizada en la diapositiva que haya afectado el estado global sincronizado del dibujo de la diapositiva, es decir, deshace todo tipo de acciones realizadas con las herramientas de dibujo excepto *Zoom-in/out* y *Pan*.

1.1.12.  Rehacer

Rehace la última acción que se deshizo con la herramienta *Deshacer*.

1.1.13.  Importar

Despliega un cuadro de diálogo para cargar al servidor una imagen que queda como fondo de la diapositiva al terminar de cargar.

1.1.14.  Descargar

Inicia una descarga de la diapositiva actual como una imagen en formato PNG.

1.1.15.  Eliminar Diapositiva

Elimina la diapositiva actual de la sesión.

Figura 3. Ejemplo de una diapositiva en la que se importó una imagen como fondo (Diagrama) y se usaron las herramientas de dibujo para destacar cosas con color rojo.

1.2. Usuarios y Grupos

En Sketchpad existen dos tipos de usuarios: Profesores y Estudiantes. Se considera que en una clase participan un profesor y una gran cantidad de estudiantes. Un usuario profesor tiene algunos privilegios sobre ciertos tipos de diapositivas y grupos con el propósito de que este pueda llevar un cierto control sobre el desarrollo de la clase. Estos privilegios serán explicados más adelante a medida que se mencionan las funcionalidades donde la distinción entre profesores y estudiantes entra en juego.

Además, se pueden crear grupos para trabajar específicamente con ciertos usuarios. Usuarios pertenecientes a un mismo grupo tienen acceso a una sección de diapositivas que sólo son accesibles a los integrantes del grupo y todos pueden trabajar en ellas concurrentemente.

Existen dos tipos de grupos: Grupos de Trabajo (Work Groups) y Grupos de Apuntes (Notes Groups). Los grupos de trabajo sólo pueden ser creados por el profesor y sus integrantes son designados por el profesor. Los grupos de apuntes pueden ser creados por cualquier tipo de usuario y cualquier integrante de un grupo de apuntes puede invitar a otros usuarios a unirse al grupo.

Por diseño, el profesor siempre es integrante de todos los grupos de trabajo a la vez. El propósito de esto es que el profesor le asigne tareas o actividades colaborativas a los grupos de trabajo y tenga acceso a ver el progreso del trabajo en todo momento.

Los grupos de apuntes por su parte son solo accesibles a los usuarios que son invitados a ellos y su propósito es incentivar a que los alumnos tomen apuntes grupales colaborativos durante la clase.

En la figura 4 se ve la pestaña *Users* de la aplicación, esta es una pestaña colapsable que se encuentra al costado izquierdo de la vista de la Drawing App.

A continución se describen las funcionalidades respectivas a usuarios y grupos.

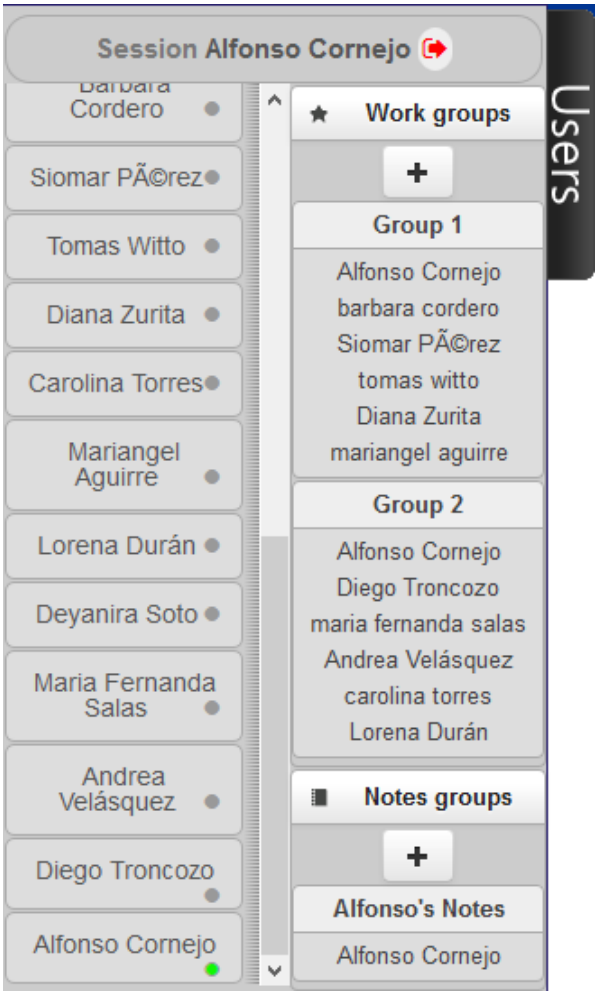


Figura 4. Pestaña *Users* con lista de usuarios y grupos de trabajo y de apuntes.

1.2.1. Crear grupos

Para crear grupos nuevos se utiliza el botón “+” en la sección correspondiente.

1.2.2. Agregar e invitar usuarios a grupos

Para agregar usuarios a grupos de trabajo, en el caso del profesor, o invitar usuarios a grupos de apuntes se deben arrastrar los nombres de los usuarios de lista de la izquierda al grupo correspondiente en la lista de la derecha. Las invitaciones a grupos de apuntes se mostrarán para los usuarios destino como una alerta que se despliega en esta pestaña.

1.2.3. Usuarios conectados y desconectados

En la lista de usuarios, los usuarios actualmente conectados aparecerán con un círculo color verde mientras que los usuarios desconectados tendrán un circulo gris.

1.3. Tipos de Diapositivas

Existen 3 tipos de diapositivas: Personales (Personal), Grupales (Groupal) y Pantalla (Screen). Las diapositivas personales son visibles solo para el usuario que las crea. Estas diapositivas pueden ser compartidas con otros usuarios copiándolas en una diapositiva grupal o de pantalla. Las diapositivas grupales son accesibles para los integrantes del grupo en el que se crea la diapositiva. Por su parte las diapositivas de pantalla son visibles para todos los usuarios, profesores y alumnos, pero, por defecto, sólo el profesor puede modificar su contenido. Los alumnos sin embargo pueden solicitar permiso al profesor para modificar una diapositiva de pantalla.

En la figura 5 se ve la pestaña *Slides* de la aplicación, esta también es una pestaña colapsable que se encuentra en el lado derecho de la vista de la Drawing App. Las secciones de los tipos de diapositivas son colapsables.

A continuación se describen las funcionalidades relacionadas con diapositivas.

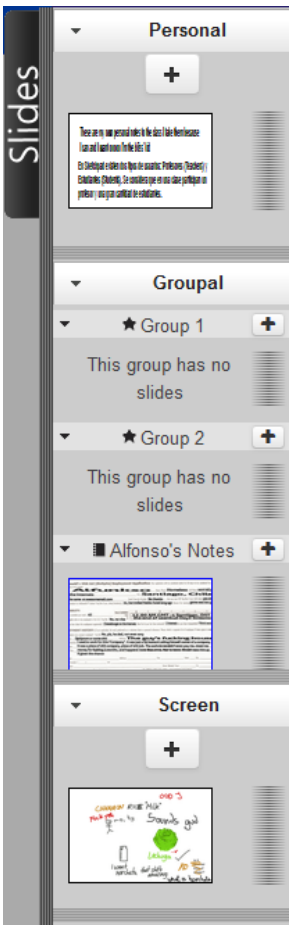


Figura 5. Pestaña *Slides* con diapositivas de distintos tipos creadas.

1.3.1. Crear diapositivas

Para crear una diapositiva nueve se debe usar uno de los botones “+” en la sección correspondiente siempre que se tenga permiso de crear diapositivas en esa sección.

1.3.2. Seleccionar diapositiva principal

Al hacer click en la mini-vista de una diapositiva la vista principal de diapositiva cambiará a la seleccionada y la diapositiva seleccionada quedará marcada con un borde azul que indica que aquella es la mostrada actualmente en la vista principal.

1.3.3. Copiar diapositivas

Al arrastrar la mini-vista de una diapositiva sobre la mini-vista de otra, se copiarán los contenidos de la diapositiva arrastrada sobre la diapositiva de destino, siempre que se cuenten con los permisos para modificar la diapositiva de destino.

1.4. Barra Superior

La barra superior de Sketchpad cumple la multifuncionalidad de mostrar información y de permitir ciertos comandos especiales que dependen del usuario y la diapositiva seleccionada actual.

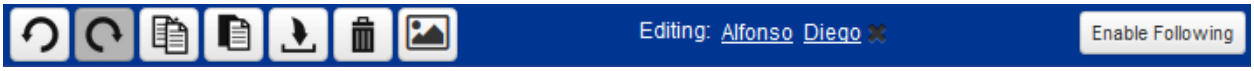


Figura 6. Barra superior mostrando 2 usuarios editando simultáneamente una diapositiva.

A la izquierda de la barra superior se encuentran algunas de las herramientas que fueron descritas previamente.

A continuación se describen las funcionalidades relacionadas con la barra superior.

1.4.1. Usuarios editando concurrentemente

En la parte central se muestra quiénes son los usuarios que están editando la diapositiva actual concurrentemente. En el caso de diapositivas grupales por ejemplo, si hay 2 integrantes de grupo con la misma diapositiva grupal seleccionada entonces el nombre de ambos integrantes se mostrará en la barra superior; pero si uno de los usuarios decide seleccionar una diapositiva diferente por algún motivo, luego su nombre desaparecerá de la barra superior hasta que vuelva a seleccionarla.

En el caso de la figura 6 se ve un ejemplo en el que el usuario Alfonso es el profesor y se encuentra una diapositiva de pantalla, el usuario Diego es un estudiante que acaba de pedir permiso para editar la diapositiva y Alfonso se lo ha concedido; luego ambos usuarios están editando la diapositiva concurrentemente y por ende ambos nombres aparecen en la barra superior.

1.4.2. Permisos de edición en diapositivas de pantalla

El profesor puede decidir quitarle el permiso de edición al estudiante haciendo click en la cruz al lado derecho del nombre del alumno en la barra superior.

En la parte derecha de la barra superior se dispondrán botones dependiendo del tipo de usuario y de la diapositiva seleccionada.

1.4.3. Seguimiento: “Enable Following”

En el caso del profesor se verá el botón “*Enable Following*” el cual al presionarse activará el modo de seguimiento para los estudiantes, el cuál no permite que los alumnos seleccionen otras diapositivas de pantalla que no sean la que el profesor se encuentra actualmente. Cuando este modo está activado los estudiantes cambiarán de

diapositiva automáticamente a la que se encuentra el profesor cuando el profesor cambie de diapositiva.

1.4.4. Permisos de Edición: “Request Edition”

Por otra parte para estudiantes en la parte derecha de la barra superior se mostrará el botón “Request Edition” cuando esté seleccionada una diapositiva de pantalla. Al hacer click en este botón al profesor le aparecerá una notificación en la pestaña de usuarios indicando el usuario que desea modificar la diapositiva y le dará la opción de concederle el permiso o ignorar la petición.

1.5. Selección de Asignatura y Actividad

La división lógica del modelo de Sketchpad consiste de asignaturas y actividades. Existen distintas asignaturas en las que distintos estudiantes están registrados y corresponden a distintos profesores. Luego cada asignatura tiene un conjunto de actividades, una actividad corresponde a una clase o una sesión de trabajo. Cada una de estas actividades toma lugar en la vista principal de Sketchpad, la que acabamos de describir, donde se crean diapositivas y grupos y los usuarios trabajan colaborativamente. Luego para comenzar a usar la aplicación y trabajar sobre diapositivas, lo primero que hay que hacer es seleccionar una asignatura y una actividad para unirse a la sesión de esa clase.

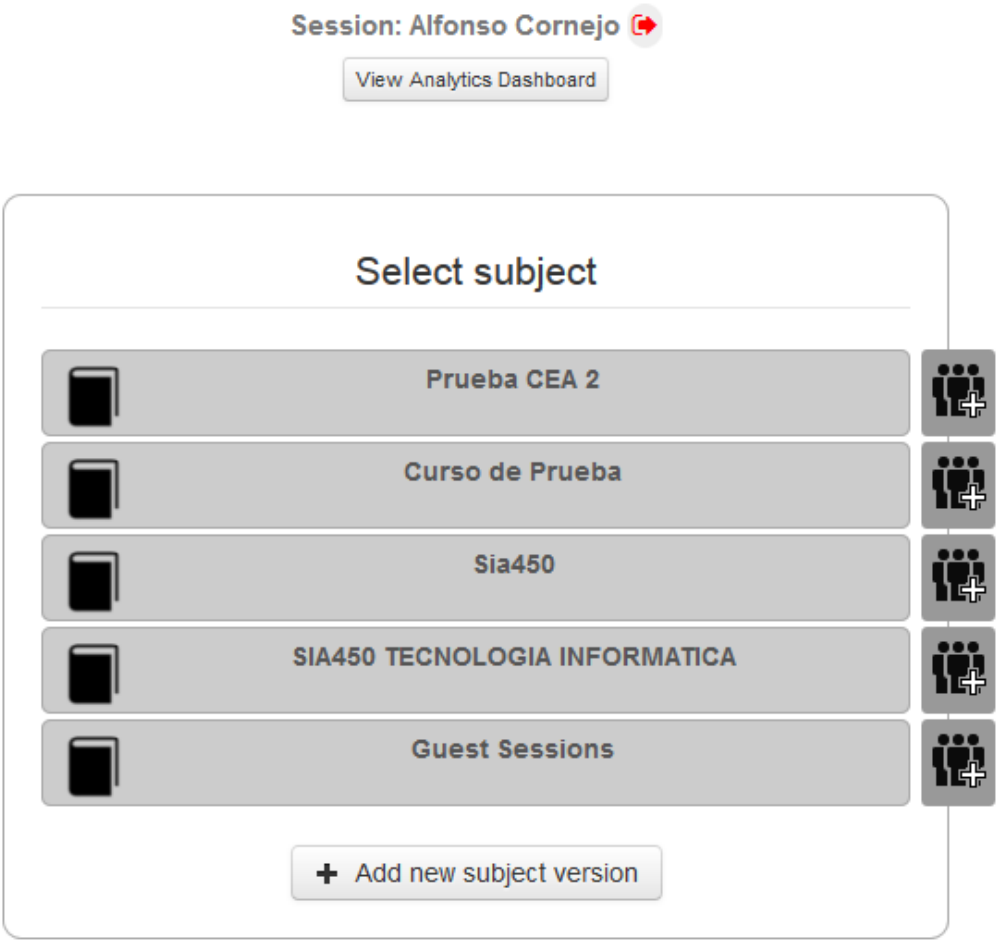


Figura 7. Pantalla de selección de asignatura.

En la figura 7 se encuentra la pantalla de selección de asignatura, al hacer click en una de ellas se entra a la pantalla de selección de actividad que se ve en la figura 8. Finalmente seleccionando una actividad se entra a la vista principal de trabajo en diapositivas colaborativamente (Figura 1).

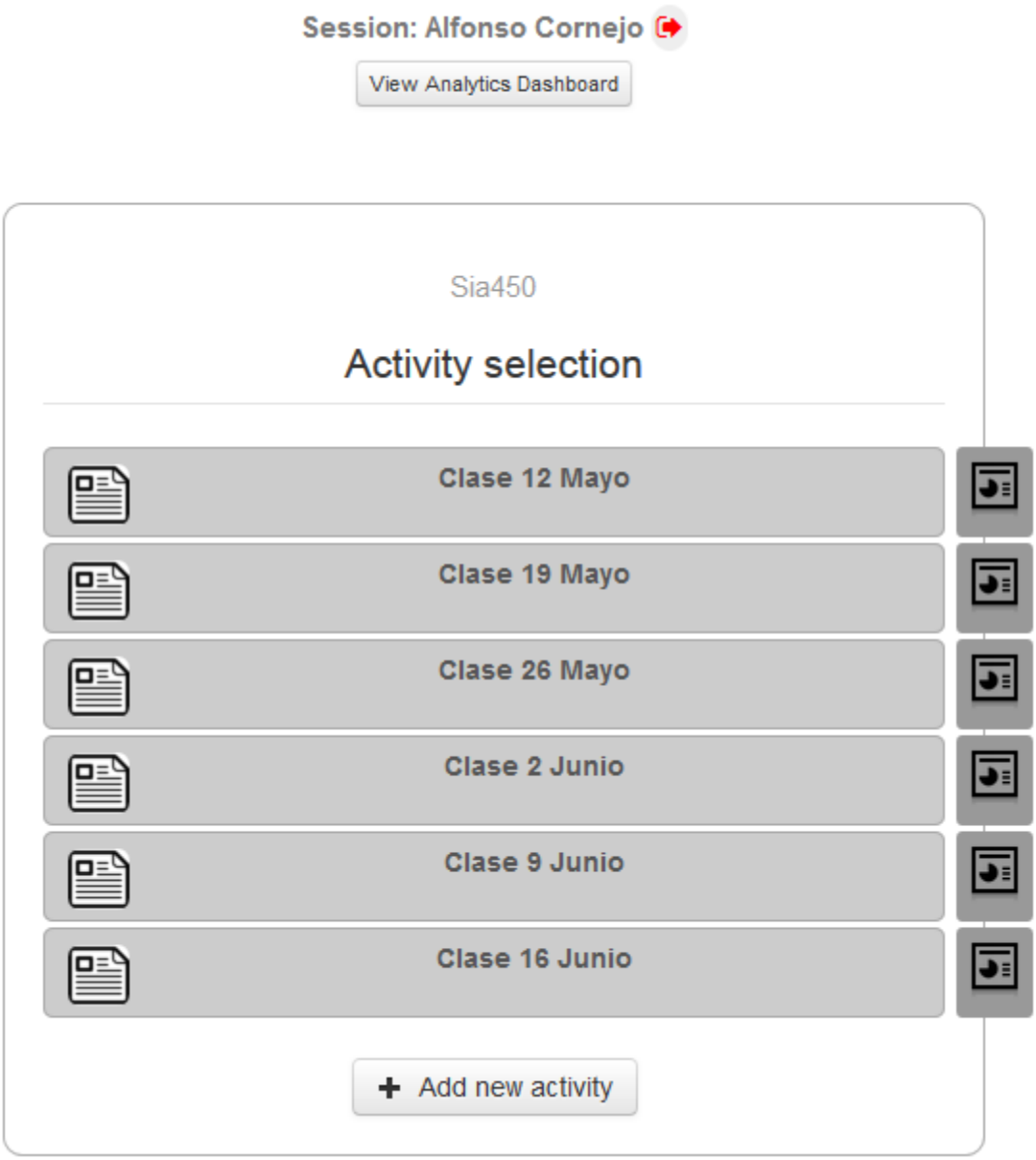


Figura 8. Pantalla de selección de actividad.

En las pantallas de selección de asignatura y de selección de actividad también se pueden crear nuevas instancias de asignaturas y actividades correspondientemente usando el botón destinado para ello al final de la página. Sólo el usuario profesor tiene el permiso de hacer esto.

1.6. Carga masiva de imágenes

Desde la vista de selección de actividad (Figura 8) se puede ingresar a la opción de cargar imágenes masivamente (Figura 9). Esta es una función que sólo el profesor tiene disponible y para ingresar a ella debe hacer click en ícono al lado derecho del nombre de la actividad.

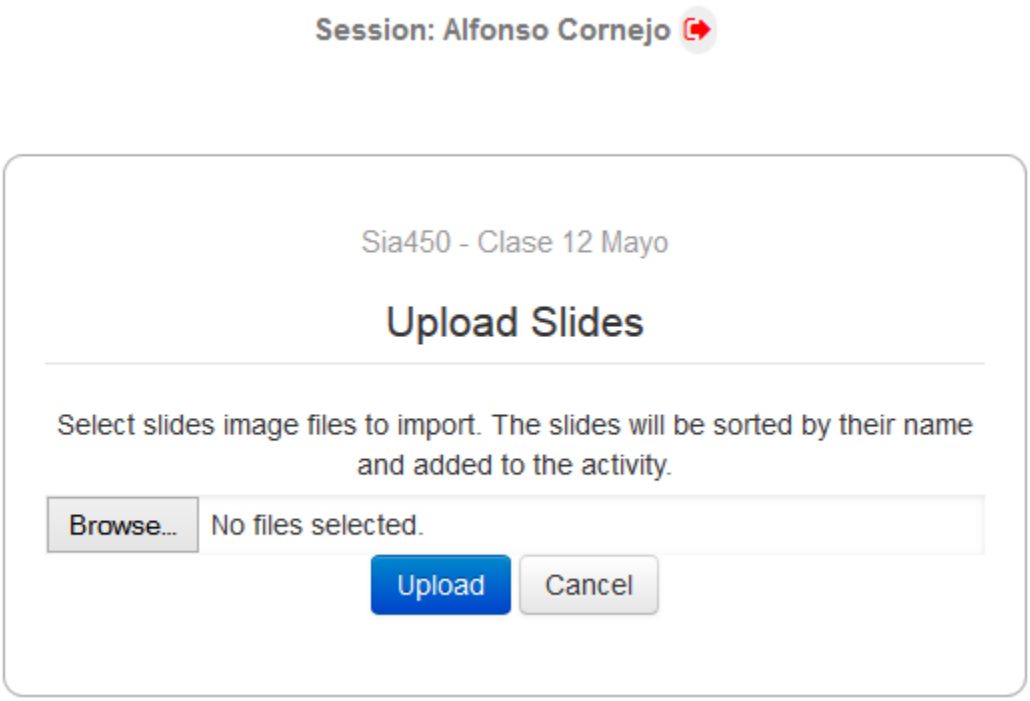


Figura 9. Carga masiva de imágenes.

Esta función permite seleccionar un conjunto de imágenes desde el dispositivo que el usuario esté usando las cuales son cargadas al servidor, ordenadas por nombre y agregadas como diapositivas a la actividad seleccionada. Esta función es útil para importar por ejemplo diapositivas exportadas de Power Point de modo que se pueda realizar una clase tal cómo se realizaría una presentación con diapositivas.

1.7. Registro de estudiantes en asignaturas

Según el diseño de Sketchpad, los estudiantes deben ser inscritos en las asignaturas para tener acceso a las actividades correspondientes. Este es un proceso que se debe realizar sólo una vez para cada asignatura y que sólo el profesor puede llevar a cabo. Para esto se debe hacer click en el ícono encontrado al lado derecho del nombre de la asignatura en la pantalla de selección de estas lo cuál lleva al usuario a la pantalla de la figura 10 en la cuál debe ingresar los correos electrónicos de los alumnos para quedar registrados en la asignatura.

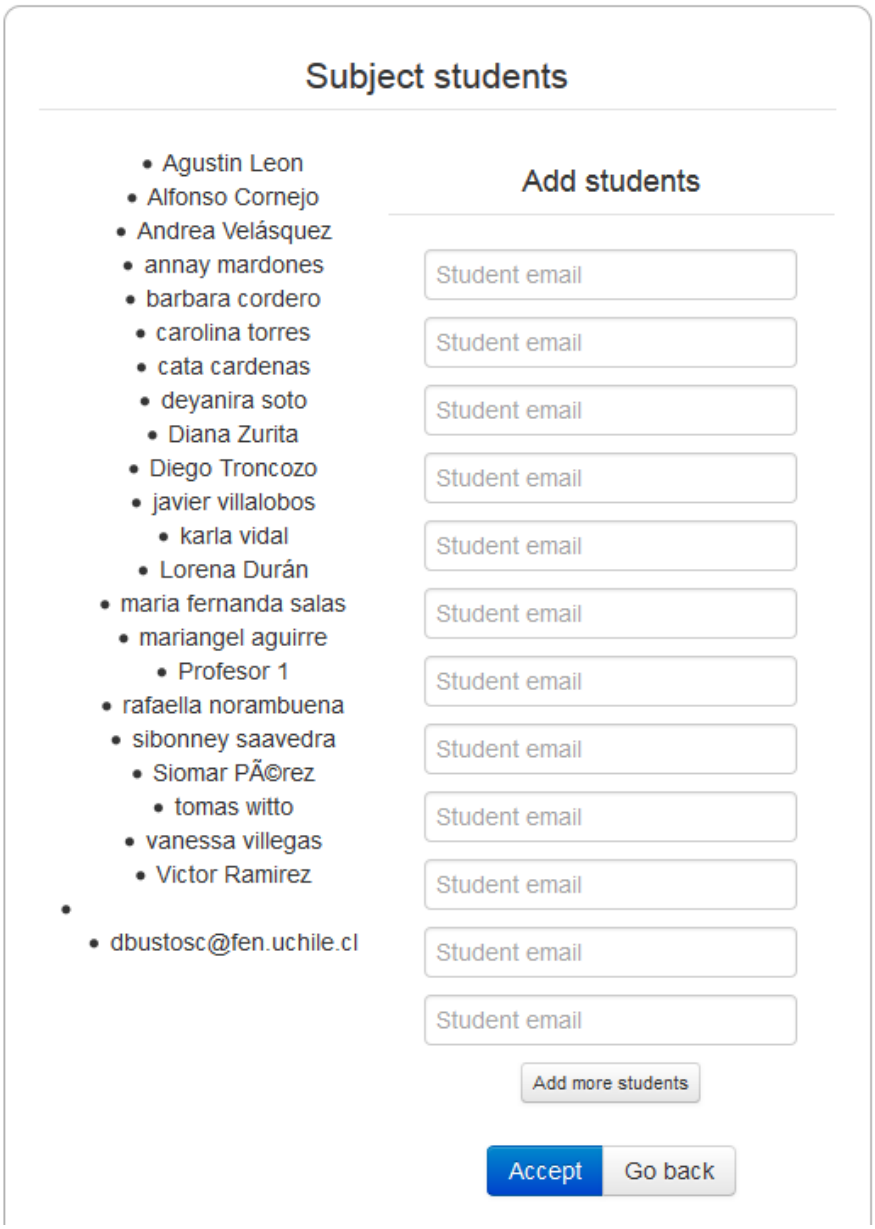


Figura 10. Pantalla de registro de estudiantes en asignatura.

1.8. Acceso usuarios

Figura 11. Página de inicio.

En esta página se puede iniciar sesión usando el sistema de la Facultad de Economía y Negocios. Esta es la primera página a la que se tiene acceso al ingresar a Sketchpad. Por diseño, se requiere que los usuarios tengan una cuenta FEN para entrar.

Cuando los usuarios ingresan a la aplicación por este medio por primera vez, se les solicita llenar un formulario de registro del que se obtienen los nombres, apellidos y correo electrónico.

2. Learning Analytics: Panel de Instrumentos

El Panel de Instrumentos de Learning Analytics, o Analytics Dashboard, está compuesto por 5 vistas que muestran distintos aspectos del comportamiento de los usuarios. El propósito de estas vistas es presentar al usuario con información valiosa de su nivel de uso de la aplicación respecto al de sus compañeros y respecto de su desempeño académico.

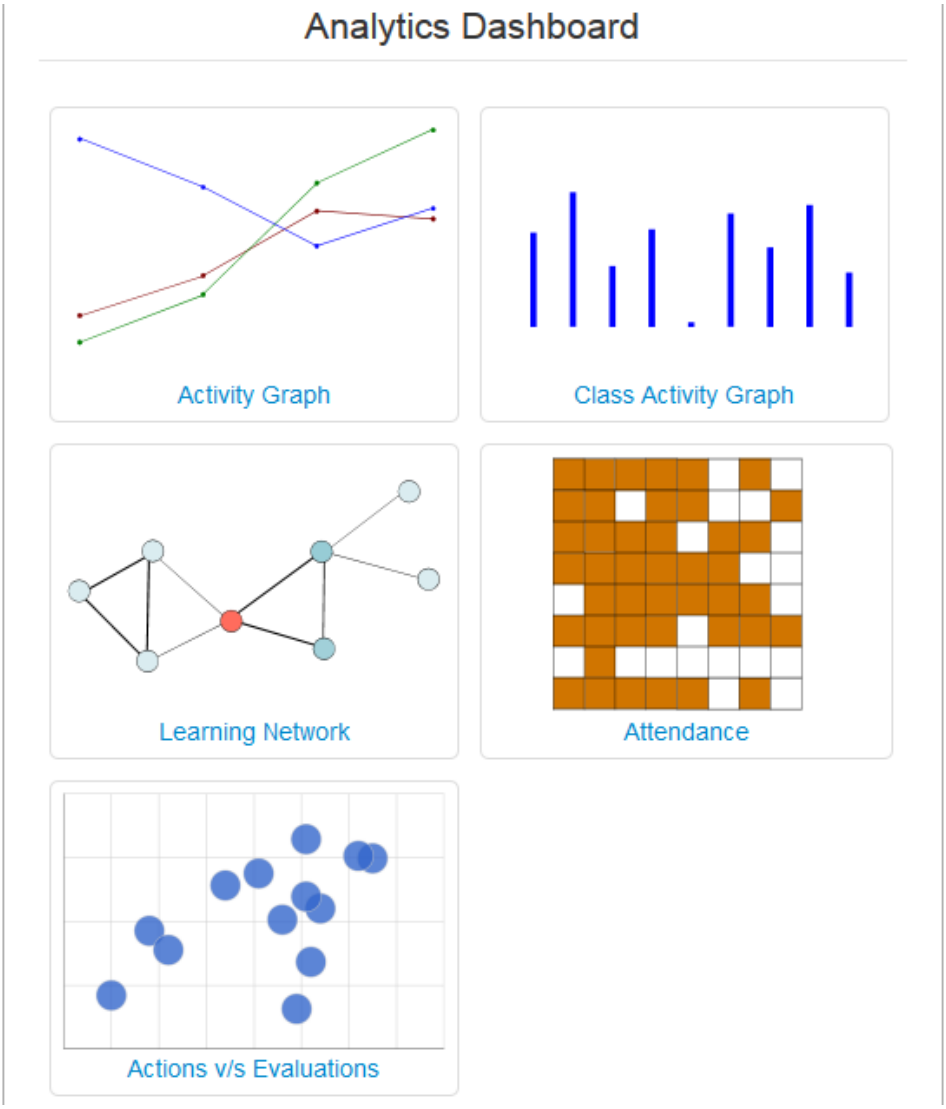


Figura 12. Analytics Dashboard.

Para ingresar al Analytics Dashboard se debe ingresar desde las páginas de selección de asignatura o de selección de actividad haciendo click en el botón “Analytics Dashboard” al inicio de la página. Dentro del Analytics Dashboard hacer click en uno de los gráficos en miniatura abre la vista correspondiente.

2.1. Visualización “Activity Graph”

Esta es una de las visualizaciones más detalladas del Dashboard y consiste en un set de gráficos de línea que muestran la progresión en el número de acciones realizadas por el usuario en las actividades, separadas por tipo de acción y comparadas con el promedio de acciones realizadas por todos los estudiantes de la asignatura.

Además, en cada gráfico se agregan como barras el resultado de las evaluaciones del estudiantes y la comparación con el promedio del curso por evaluación. Estas barras aparecen en el gráfico sobre la actividad más cercana en fecha a la evaluación correspondiente.

El número de acciones realizadas por los usuarios que se cuentan incluye las acciones:

- Trazos dibujados
- Texto escrito
- Objetos removidos
- Objetos movidos
- Cambios de color
- Deshacer/Rehacer
- Copiar/Pegar
- Diapositivas Creadas

Los distintos gráficos separan el número de acciones realizadas en las siguientes categorías.

2.1. Categorías de Slides

2.1.1. Categoría “Personal Slides”

Número de acciones realizadas sólamente en diapositivas personales.

2.1.2. Categoría “Notes Groups”

Número de acciones realizadas sólamente en diapositivas de grupos de apuntes.

2.1.3. Categoría “Work Groups”

Número de acciones realizadas sólamente en diapositivas de grupos de trabajo.

2.1.4. Categoría “Teacher Slides”

Número de acciones e interacciones realizadas con diapositivas de pantalla.

Figura 13. Vista agregada del Activity Graph de un usuario. En los gráficos se puede ver que este usuario no asistió a la actividad “Clase 16 Junio” y por ende no tiene datos en ese día.

En la parte superior de los gráficos se encuentran ciertos controles para filtrar o extender la visualización.

El primer selector que se ve en la figura 13 es el selector de asignatura, este permite visualizar las estadísticas de distintas asignaturas en caso de que un usuario participe en más de una asignatura de Sketchpad.

El segundo selector es un selector de usuarios que está solo disponible para el profesor y le permite ver el Activity Graph de distintos usuarios de un asignatura.

Se dispone además del checkbox “Deaggregate by action type” el cual cambia la visualización a la que se ve en la figura 14 y permite ver más en detalle las estadísticas personales mostrando para cada clase el número de acciones realizadas por el usuario separadas por tipo de acción en un gráfico de área apilado. Estos gráficos también incluyen los resultados de evaluaciones pero no las comparaciones entre número de acciones personales con las del curso ni evaluaciones personales contra promedio del curso.

En la vista desagregada del Activity Graph aparece también un 5 gráfico que corresponde a acciones que sólo son posibles sobre grupos de apuntes. Estas acciones son:

- Grupos Creados
- Invitaciones Enviadas
- Invitaciones Recibidas

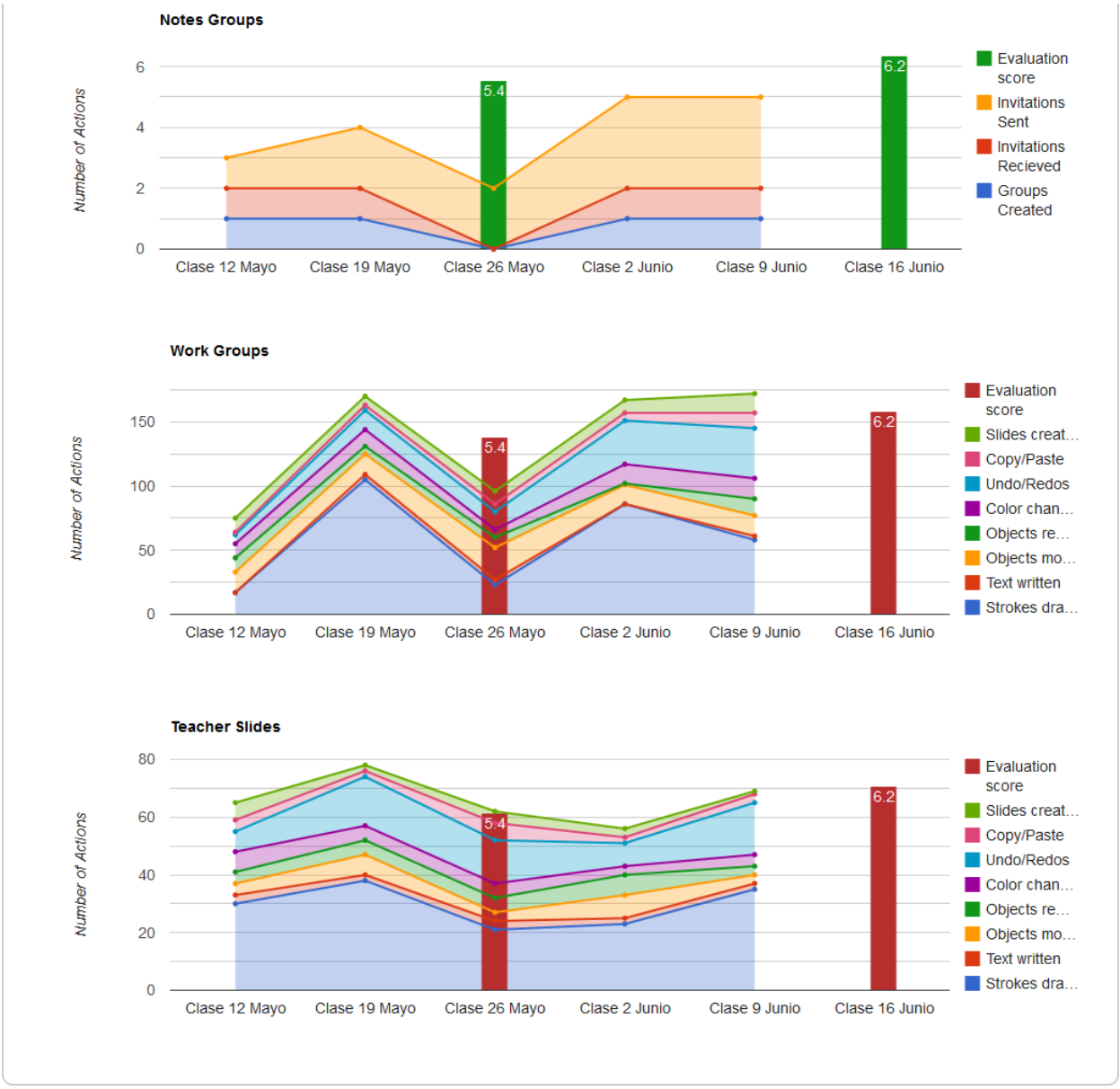


Figura 14. Extracto del Activity Graph desagregado por tipo de actividad. Esta vista corresponde al de la misma usuario de la figura 13. El primer gráfico que se puede ver en la figura corresponde al de las acciones específicas de grupos de apuntes.

2.2. Visualización “Class Activity Graph”

Esta visualización consiste de un gráfico de barras que muestra el promedio total de acciones realizadas por los estudiantes de una asignatura clase a clase separadas por categoría de acciones y contrastada con el promedio del resultado de evaluaciones. El promedio de evaluaciones de un alumno corresponde a la media aritmética de sus resultados de todas las evaluaciones de la asignatura.

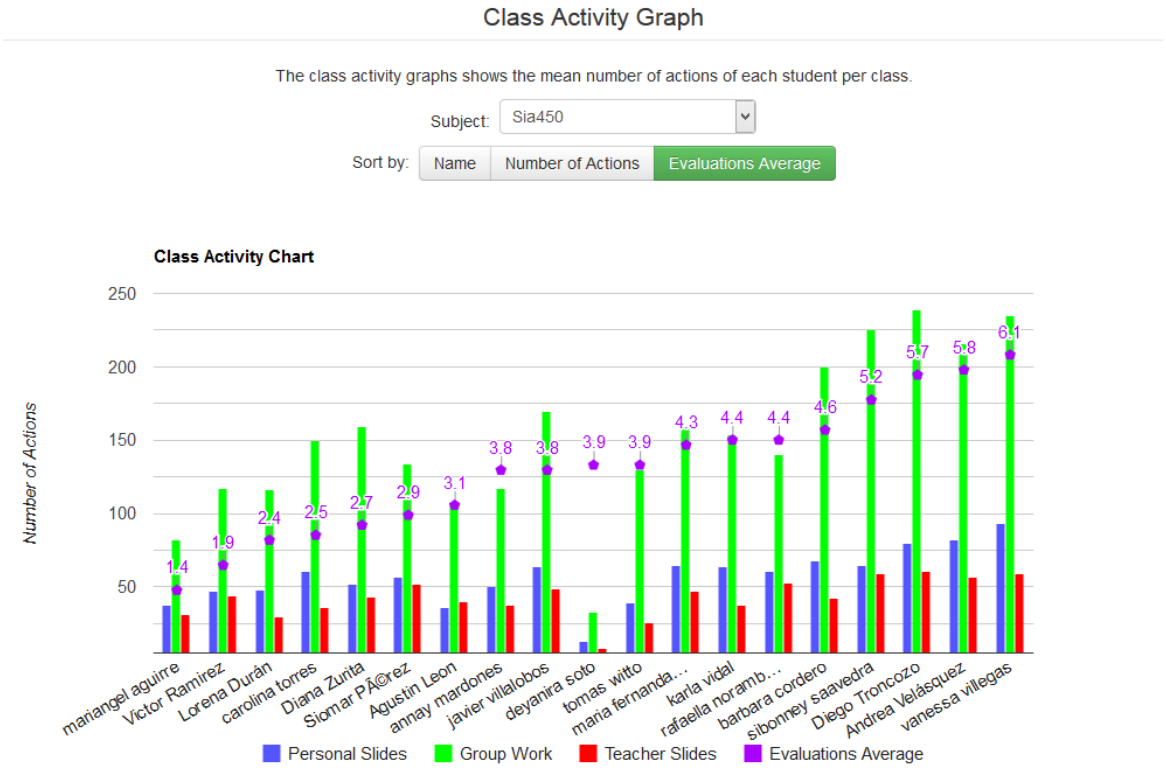


Figura 15. Class Activity Graph de una asignatura ordenado por promedio de evaluaciones.

La visualización incluye un selector de asignatura al igual que en el Activity Graph y permite ordenar el eje horizontal del gráfico según uno de los siguientes criterios:

- Nombre estudiante
- Número de Acciones (Suma total de las 3 categorías)
- Promedio Evaluaciones

Las categorías de las barras por alumno corresponden a las siguientes.

- **Personal Slides**: Número de acciones realizadas en diapositivas personales
- **Group Work**: Número de acciones realizadas en grupos (de apuntes o de notas)
- **Teacher Slides**: Número de acciones realizadas sobre diapositivas de pantalla.

2.3. Visualización “Learning Network”

En esta visualización se utiliza un grafo que representa la red de usuarios que el estudiante ha desarrollado durante el transcurso de la asignatura mediante el trabajo en grupo. En el centro del grafo se encuentra el usuario principal que ve la visualización, los demás nodos del grafo son los usuarios con los que el usuario principal ha trabajado, ya sea en grupos de trabajo o de apuntes, y se conectan al usuario principal mediante enlaces de colores.

El color de los enlaces representan un grupo, de modo que 3 nodos conectados por nodos del mismo color representan a 3 usuarios que trabajaron en un mismo grupo en alguna actividad. El grosor de un enlace representa la cantidad de veces que los dos usuarios conectados han trabajado juntos, de modo que mientras más grueso el nodo más veces estos usuarios han trabajado en grupos distintos.

El color de los nodos representa el número relativo de acciones realizadas por los usuarios. Un color más oscuro indica que el usuario realizó una gran cantidad de acciones mientras que un color más claro indica una baja cantidad de acciones.

Al igual que las visualizaciones revisadas previamente, esta visualización incluye un selector de asignatura. Adicionalmente se incluyen selectores de filtros que modifican el grafo.

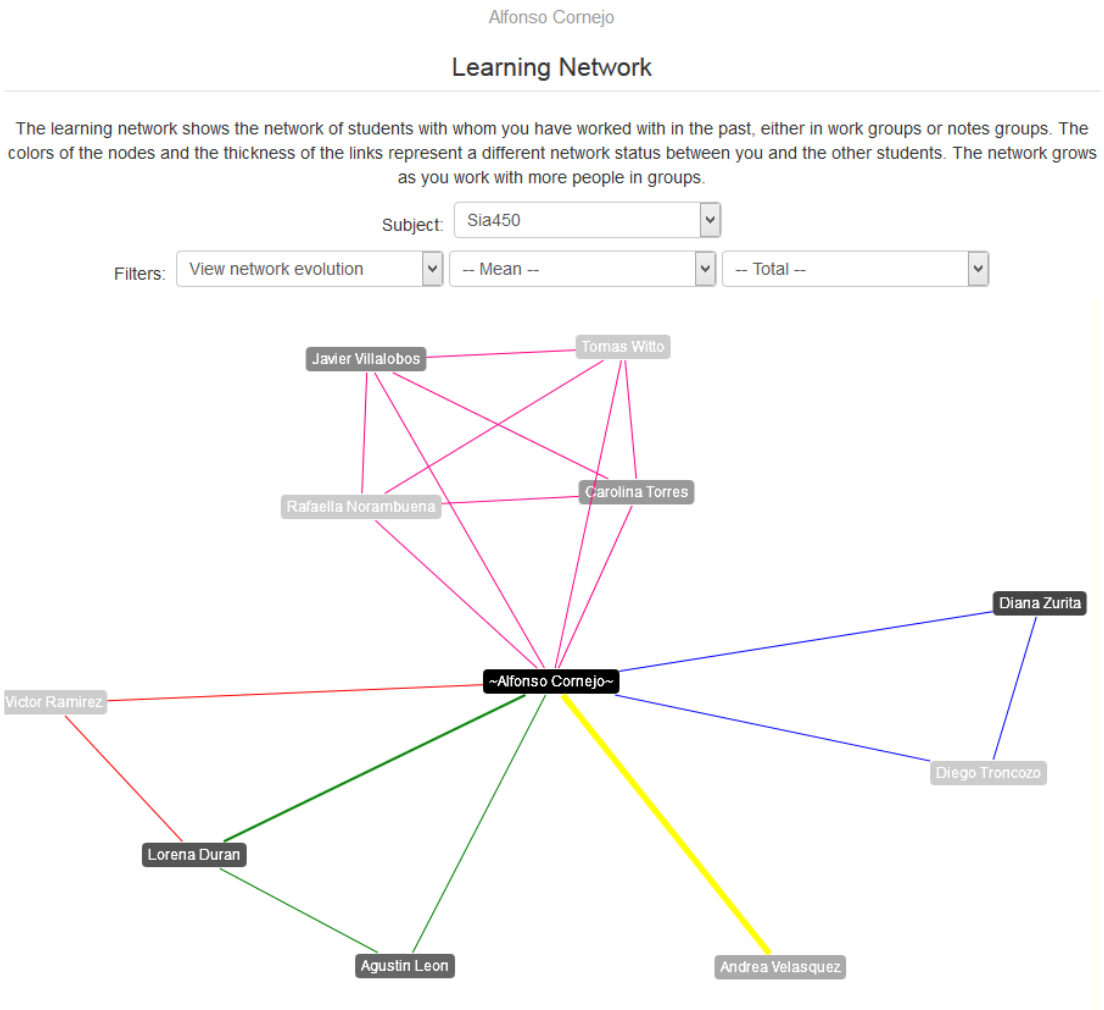


Figura 16. Learning Network evolutiva del usuario Alfonso Cornejo en la asignatura Sia450 como ejemplo.

2.3.1. Filtro Tipo de red

Se pueden seleccionar dos tipos de visualización de red.

1. **View network evolution**: Muestra la red construyéndola aditivamente clase a clase de modo que el grafo final incluye los grupos y usuarios con los que se ha trabajado durante el transcurso de todas las actividades hasta la última seleccionada en el filtro de actividad.
2. **View activity snapshot**: Muestra la red correspondiente a la actividad particular seleccionada en el filtro de actividad sin incluir usuarios y grupos con los que se haya trabajado en otras actividades.

2.3.2. Filtro Actividad

Permite seleccionar una actividad en particular que se quiera revisar en detalle o si en cambio se desea ver las estadísticas en promedio.

Si el Tipo de red seleccionado corresponde a la *Network evolution* la visualización resultante mostrará la evolución de la red hasta ya sea la actividad seleccionada o hasta la última actividad si es que el filtro de actividad se selecciona en "-- *Mean* --".

Si el Tipo de red seleccionado es el *Activity snapshot* entonces la visualización resultante mostrará sólo la red generada en la actividad seleccionada y los colores de nodos y grosores de enlaces representarán las acciones y grupos generados en esa actividad.

2.3.3. Filtro Tipo de acción

Este filtro permite filtrar la representación del color de los nodos seleccionando el tipo de acción que se desee considerar para pintar los nodos. La opción "-- *Total* --" corresponde a la suma de todos los tipos de acciones. Los tipos de acciones posibles a seleccionar son: Trazos dibujados, Texto escrito, Objetos movidos, Objetos removidos, Cambios de color, Deshacer/Rehacer, Copiar/Pegar, Diapositivas creadas, Invitaciones enviadas, Invitaciones recibidas, Grupos creados.

2.4. Visualización “Attendance”

Esta visualización consiste en una tabla que muestra la asistencia a clases de cada usuario integrante de una asignatura. En las filas se encuentran los usuarios y en las columnas están todas las actividades. Las celdas pintadas amarillas indican que el usuario asistió a la actividad correspondiente y si la celda está en blanco significa que el usuario se ausentó. La intensidad del color de las celdas amarillas indican el número de acciones que el usuario realizó aquel día en esa actividad relativo al total de acciones realizadas por el resto de usuarios.

Se agregan además columnas por cada evaluación de la asignatura entre medio de las actividades según la fecha que corresponda relativa a las actividades y una columna final que incluye el promedio de las evaluaciones de cada alumno. Los resultados de las evaluaciones de cada estudiante aparecen en la celda correspondiente o la celda estará vacía si no hay resultado de la evaluación aún.

Al igual que las demás visualizaciones hay un selector de asignatura y esta visualización es ordenable de arriba a abajo según nombre de estudiante, número de asistencias a clase o evaluaciones.

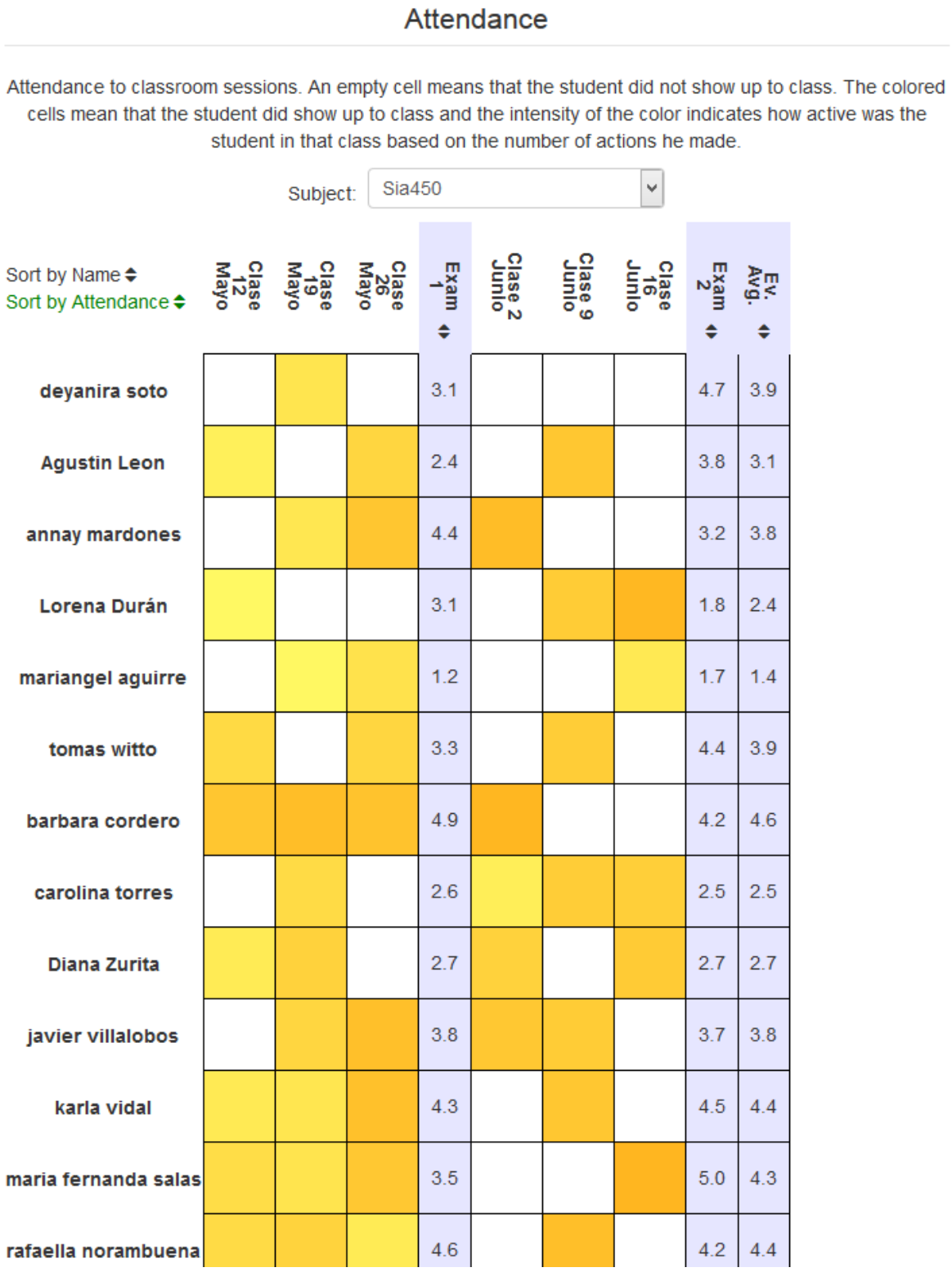


Figura 17. Attendance de la asignatura Sia450 ordenada por la asistencia total a clases.

2.5. Visualización “Actions v/s Evaluations”

Esta visualización consiste en un gráfico de burbujas que compara el número de acciones de los usuarios contra el resultado de sus evaluaciones. En el eje horizontal del gráfico se encuentra el puntaje de evaluación y en el eje vertical el número de acciones. Cada punto en el gráfico representa a un usuario. El color de los puntos es irrelevante pero el tamaño de cada punto representa el número de asistencias a clases del usuario.

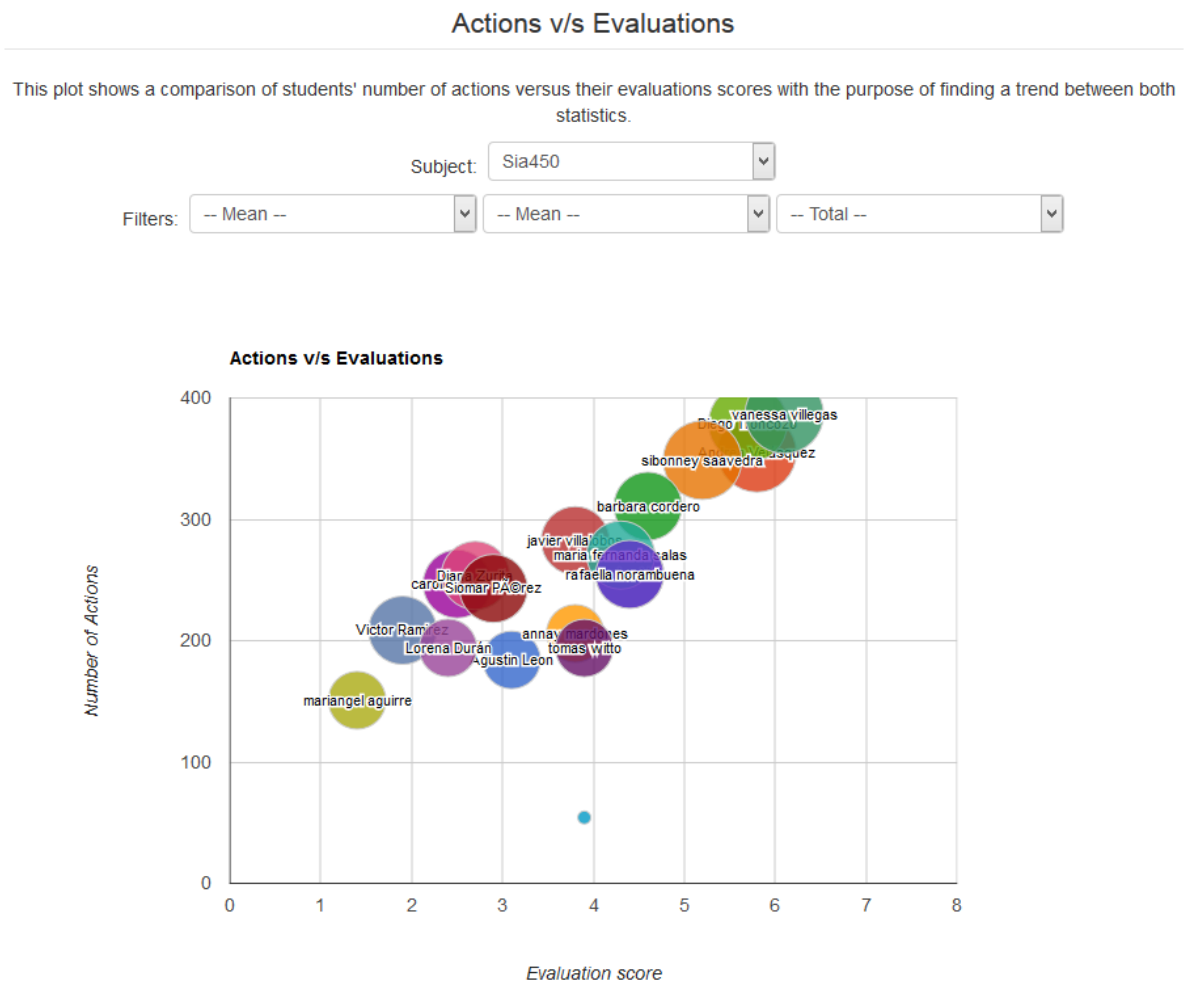


Figura 18. Visualización Actions v/s Evaluations de la asignatura Sia450 sin aplicar filtros.

En esta visualización encontramos nuevamente el selector de asignatura y los siguientes filtros.

2.5.1. Filtro Evaluación

Selecciona el valor que se usará para el eje horizontal en cada punto. La opción "-- *Mean* --" utiliza la media aritmética de todas las evaluaciones como puntaje de evaluación mientras que seleccionar una evaluación particular mostrará el resultado de esa evaluación como valor del punto.

2.5.2. Filtro Actividad

Selecciona la actividad para la que se calcularán el número de acciones que se usará para el eje vertical de los puntos. La opción "-- *Mean* --" utiliza la media aritmética de las acciones de todas las actividades.

2.5.3. Filtro Tipo de acciones

Selecciona el tipo de acciones que se utilizarán para calcular el valor mostrado en el eje vertical. La opción "-- *Total* --" usa el total de acciones realizadas como valor.

VI. Arquitectura y Diseño

El diseño de la aplicación se divide en varios componentes. Cada componente por su parte tiene en algunos casos su propio diseño dependiendo de su complejidad. La mayor parte del software está implementado como una aplicación web JavaEE sobre un proyecto Spring MVC.

La figura 19 muestra el diagrama general de la arquitectura de Sketchpad; esta consiste en una aplicación web que se despliega en un servidor Tomcat en conjunto con el Servidor de Objetos Acoplados (o Coupled Objects Server, COS) y dos bases de datos, una para la aplicación web y otra para el COS. En breve se explicará este diagrama elemento a elemento con la descripción de la funcionalidad del elemento y su diseño particular.

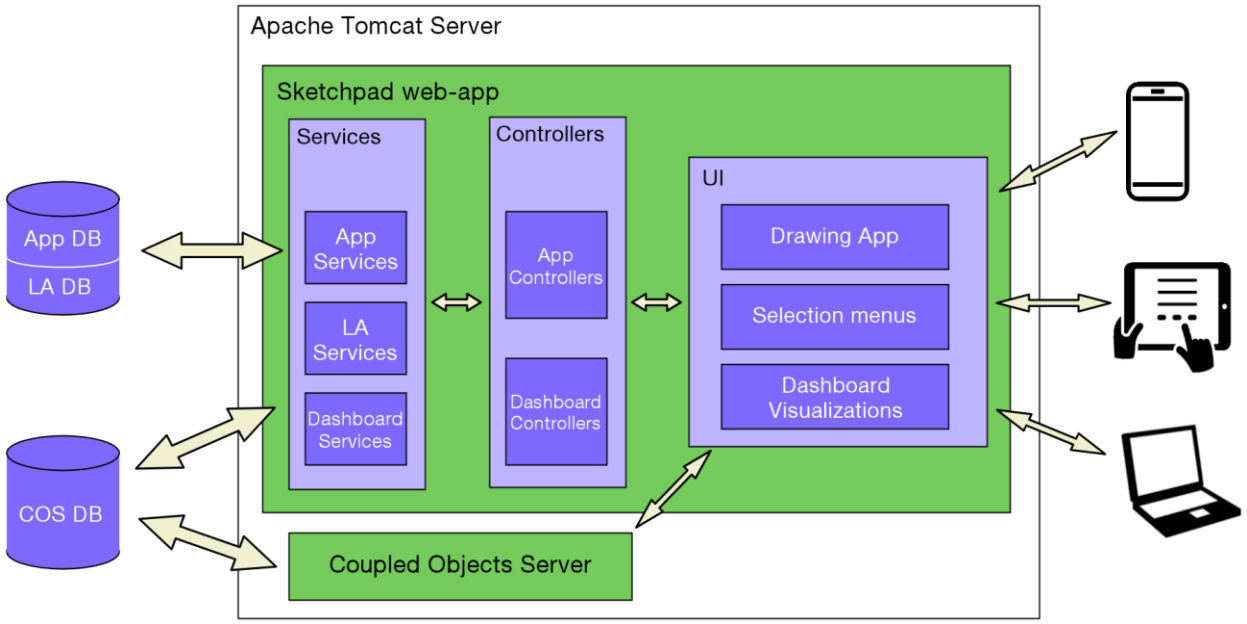


Figura 19. Diseño arquitectónico de Sketchpad.

Cómo se puede ver en la figura, el flujo de información e interacciones dentro de la aplicación ocurre, en términos generales, en las direcciones indicadas por las flechas. Cuando un usuario accede a la aplicación utilizando un dispositivo con acceso a internet mediante un navegador web (Celular, tablet, laptop u otro), este gatilla un request en el servidor tomcat que es respondido por el contexto web *Sketchpad web-app* el cual al momento de generar la vista o UI de respuesta al usuario se comunica con el controller específico de aquella vista el cual a su vez se comunica con los servicios necesarios para recuperar los datos a mostrar en la vista los cuales del mismo modo acceden a la base de datos requerida para obtener los datos. En el camino los datos pueden ser procesados y post-procesados por los servicios antes de ser

entregados al controlador y finalmente son presentados en la vista dibujando el contenido de esta dinámicamente según los datos recibidos.

Por su parte el COS se despliega como un contexto web sobre el mismo servidor Tomcat pero en un contexto separado de Sketchpad. El COS es el encargado de la mayor parte de la comunicación y sincronización de diapositivas entre usuarios en tiempo real. Este no contiene vistas sino sólo interfaces de comunicación usando JSON que son usadas intensivamente en la *Drawing App*.

1. Servidor “Coupled Objects Server”

Para lograr hacer sincronización y colaboración, la aplicación se comunica con el servidor CoupledObjectsServer (COS). Este es un servidor que se despliega como aplicación web en el servidor Tomcat y permite acoplar objetos Javascript para, a través de mensajes que son enviados cada cierto período de tiempo, mantener coordinado el estado del objeto Javascript en todos los dispositivos que se encuentran conectados a la sesión. De esta manera al acoplar una diapositiva con un identificador único al servidor COS este se encarga de que todos los usuarios conectados vean el mismo estado de la diapositiva y en definitiva puedan trabajar colaborativamente sobre ella.

La representación lógica de las diapositivas que son acopladas al COS será descrita más adelante cuando se revise el diseño del elemento “*Drawing App*” como aplicación Javascript. Por ahora es importante tener presente que el COS es el servidor que conoce el estado de los dibujos en las diapositivas en todo momento y es el encargado de notificar a las aplicaciones clientes que el estado de una diapositiva ha cambiado para mantener el dibujo sincronizado entre usuarios.

Una vez montado el COS sobre el servidor Tomcat este queda accesible en un context-path específico. Para usar las funcionalidades del COS desde la Drawing App se incluye una librería Javascript provista por el COS que se encarga de hacer la lógica necesaria para acoplar objetos. La única configuración que se necesita en la librería es ser inicializada usando la URL completa con el context-path del COS dentro del Tomcat.

Al acoplar un objeto al servidor COS, la librería Javascript se encarga de modificar el objeto redefiniendo los atributos de este, especialmente las funciones del objeto, para recibir actualizaciones de parte del COS y modificar el estado del objeto correspondientemente.

La librería inicializada del COS se encarga también de enviar una solicitud de actualización al servidor mediante AJAX consultando por nuevos eventos y mensajes que se hayan replicado en objetos acoplados. Este request es enviado cada una cierta cantidad configurable de segundos (en el caso de Sketchpad cada 1 segundo) y la respuesta contiene una lista de los nuevos mensajes replicados o de los nuevos estados de objetos, los cuales son informados a los objetos acoplados acordemente. En la figura 45 se puede ver un diagrama del funcionamiento del servidor COS y de los objetos acoplados.

Todos las acciones sincronizadas realizadas por los usuarios mientras ocupan Sketchpad quedan registradas en la base de datos del COS. Por acciones sincronizadas nos referimos a todas las acciones que son replicadas al resto de los usuarios cuando se realizan, tales como trazos dibujados, cambios de vista de la diapositiva actual, creación de grupos e invitaciones a ellos, etc.

Hay ciertas acciones que no quedan registradas en la base de datos ya que no son replicadas a los demás usuarios tales como seleccionar una herramienta de dibujo distinta o abrir y cerrar las pestañas laterales. Se decidió no recopilar este tipo de acciones ya que esto agregaría una carga de uso de red al tener que enviar el registro de estas acciones al servidor que no valdría la pena ya que se consideró que este tipo de son de poca relevancia y no contribuirán en gran medida a encontrar comportamientos o tendencias interesantes de los usuarios.

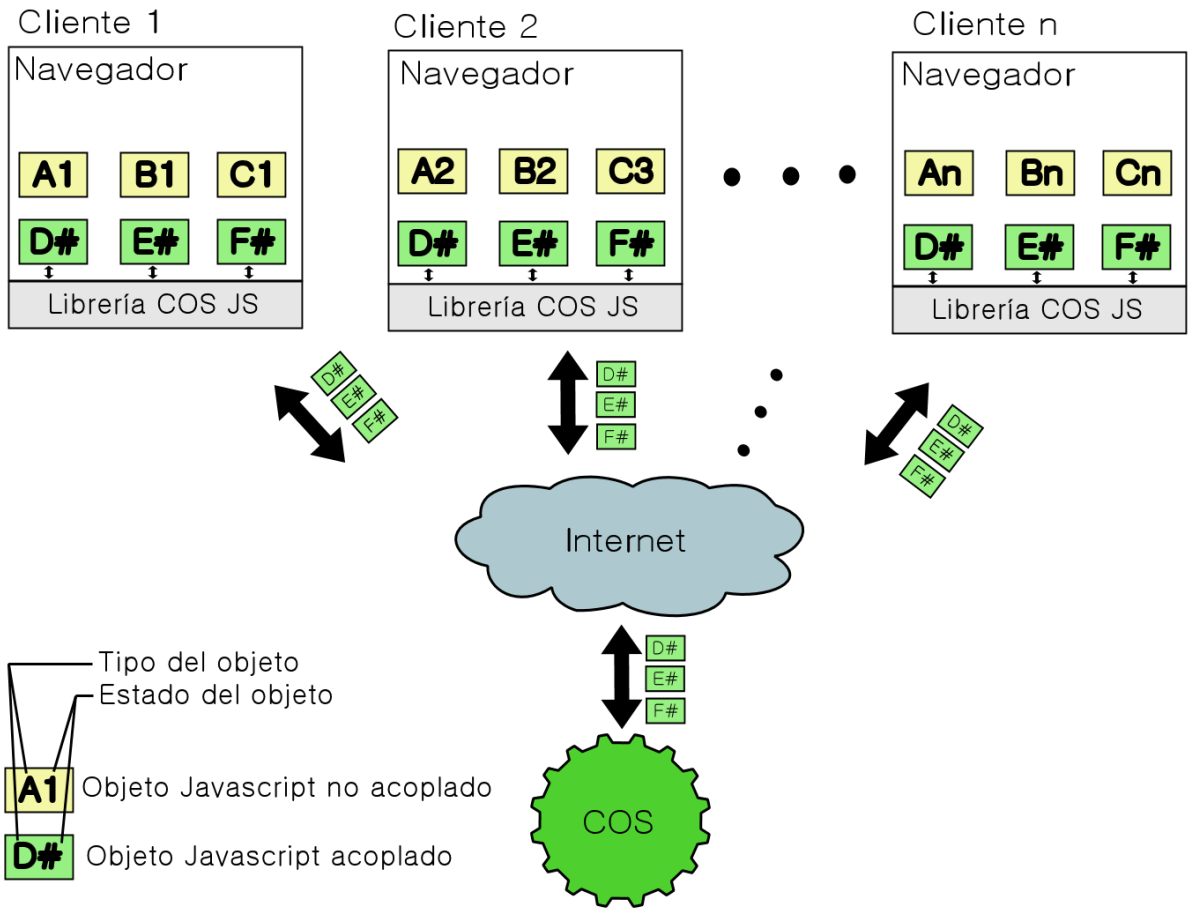


Figura 45. Diagrama de funcionamiento del COS con objetos acoplados Javascript.

2. Bases de Datos

Las bases de datos usadas comparten una misma instancia de un servidor de bases de datos Postgres 9. App DB y LA DB comparten el mismo esquema y datos de conexión y se dividen en tablas con notaciones y propósitos distintos mientras que COS DB es una instancia de base de datos separada con credenciales de conexión propios del servidor COS.

2.1. Base de datos: COS DB

Base de datos del Coupled Objects Server (COS). Esta base de datos se encarga de persistir todos los mensajes que son replicados a los usuarios sincronizados, es decir todas las acciones que ocurren en la aplicación quedan en esta base de datos. Aquí se almacena el estado de los dibujos en las diapositivas de una sesión y se persisten para que sean accesibles posteriormente a una sesión de clases.

La comunicación con esta base de datos ocurre en dos instancias; a través del COS durante el transcurso de una sesión de clases en el que se persisten y leen los mensajes de cambio de estado de las diapositivas y al consultar por el Analytics Dashboard. Los datos mostrados en el Analytics Dashboard son obtenidos inicialmente de la COS DB, ya que esta es la base de datos que contiene el registro de todas las acciones sincronizadas que ocurrieron durante una clase, sin embargo, los datos están persistidos en un formato específico del servidor COS por lo que estos se procesan y transforman a un formato más útil de leer para el Dashboard y se persisten en este nuevo formato en la base de datos de Sketchpad, específicamente en tablas específicas para este propósito que son las que se describirán en "*LA DB*". Los servicios usados para transformar los datos de la COS DB y cargarlos en LA DB serán descritos más adelante.

2.2. Base de datos: App DB

Base de datos principal de la herramienta de dibujo colaborativo de Sketchpad. Consiste de las tablas necesarias para mantener el registro y control de estudiantes y profesores, actividades, asignaturas y sus inscripciones. La comunicación con esta base de datos ocurre mayormente en los menús de selección y registro y para algunos detalles de la vista Drawing App que no son persistidos por el COS tales como la lista de usuarios e identificadores únicos de diapositivas y grupos.

Las tablas encontradas en esta base de datos se corresponden casi directamente con una entidad del modelo de la arquitectura MVC de la aplicación web, estas entidades serán descritas en profundidad más adelante cuando se revise dicho modelo.

2.3. Base de datos: LA DB

Esta base de datos comparte el esquema de la *App DB* pero contiene sólo datos del Analytics Dashboard en un formato conveniente para ser mostrados en él. Esta base de datos se carga mediante un servicio que se encarga de extraer y transformar los datos encontrados en la *COS DB*.

Al igual que en la *App DB*, las tablas encontradas aquí tienen una correspondencia directa con las entidades del modelo de datos utilizado para el Analytics Dashboard el cuál será descrito más adelante.

3. Modelo de Datos

Usando las entidades de la *Sketchpad web-app* y las tablas de las bases de datos *App DB* y *LA DB* se implementa un Modelo de Datos de la aplicación. En este encontramos cada entidad o elemento de la aplicación que debe ser persistido que no se incluyen en los elementos persistidos por el *COS*. A continuación se explican los componentes del modelo de datos. Si bien la implementación física en las base de datos de este modelo no está separada en dos distintas bases de datos, en este informe se explicarán como dos diagramas de entidades distintos el modelo de datos que está relacionado exclusivamente con la *Drawing App* y el modelo de datos que se utiliza exclusivamente para el *Analytics Dashboard*.

3.1. Modelo de datos para “Drawing App”

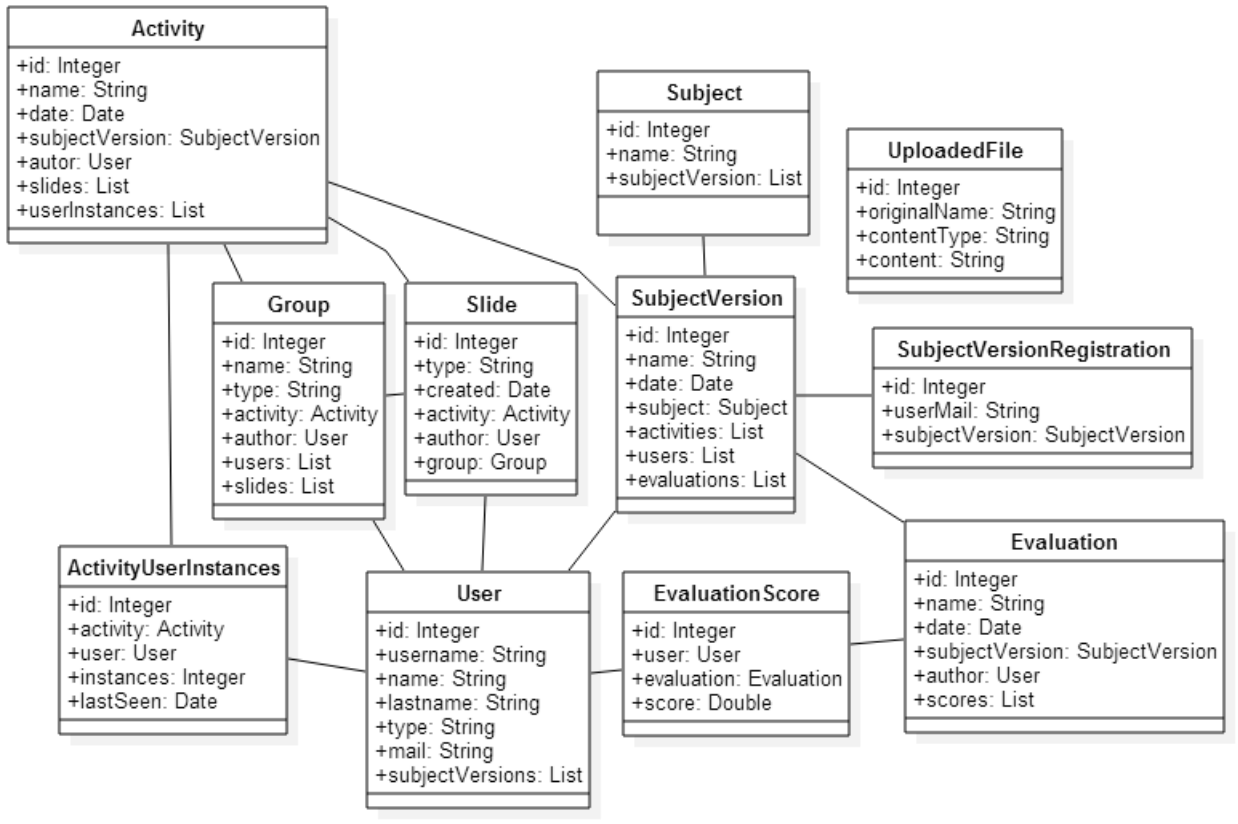


Figura 20. Modelo de datos con las entidades usadas en la Drawing App.

Este modelo de datos está diseñado para su uso en los Selection Menus y en la Drawing App.

3.1.1. User

Contiene la información de los usuarios para autenticarse en la aplicación junto con su nombre, email y tipo de usuario.

3.1.2. Subject

Contiene el nombre de las asignaturas registradas en Sketchpad.

3.1.3. SubjectVersion

Instancias de cursos que se dan en una fecha determinada y con un nombre específico, en esta tabla se identifican los cursos que se dictan por semestre.

3.1.4. SubjectVersionRegistration

Se usa para registrar alumnos en un SubjectVersion, es decir, registrar usuarios como alumnos de un curso en un semestre. Para esto se ingresan los mails de los alumnos en esta tabla y luego se revisan los usuarios registrados buscando mails que coincidan con los encontrados en esta tabla, los usuarios que se encuentren son agregados a la lista de usuarios del SubjectVersion.

3.1.5. Activity

Esta es la entidad que representa una clase, laboratorio o cualquier actividad educativa presencial. Contiene la información del nombre de la clase, la fecha en la que se hizo y el profesor que hizo/hará la clase.

3.1.6. Slide

Esta entidad es usada para almacenar la información relevante de las diapositivas y están asociados a una actividad. De todos los campos que esta entidad tiene, los campos ocupados por esta versión de la aplicación son el número identificador único de diapositiva, el tipo de diapositiva, el creador de la diapositiva, el identificador del archivo de fondo para diapositivas importadas.

3.1.7. UploadedFile

Guarda la información de imágenes que se suban a la aplicación al hacer importación de diapositivas externas. Las imágenes son guardadas en un campo de texto en la base de datos con la información de la imagen transformada a base64 en este.

3.1.8. Group

Información de los grupos tales como número identificador único de grupo, nombre y tipo.

3.1.9. ActivityUserInstances

Esta entidad se ocupa para registrar las instancias de usuario actualmente usando la aplicación. Para cada actividad se mantiene un registro de todos los usuarios que ingresan a ella y este registro es el que se utiliza para dibujar el estado de conexión de los usuarios en la pestaña *Users* de la Drawing App.

3.1.10. Evaluation

Información de evaluaciones. Tal como una actividad, una evaluación pertenece a un SubjectVersion y tienen un nombre y una fecha determinada.

3.1.11. EvaluationScore

El resultado de evaluación de los estudiantes por evaluación. Este es el registro que se lee por usuario para determinar su puntaje en los gráficos del Analytics Dashboard.

3.2. Modelo de datos para “Analytics Dashboard”

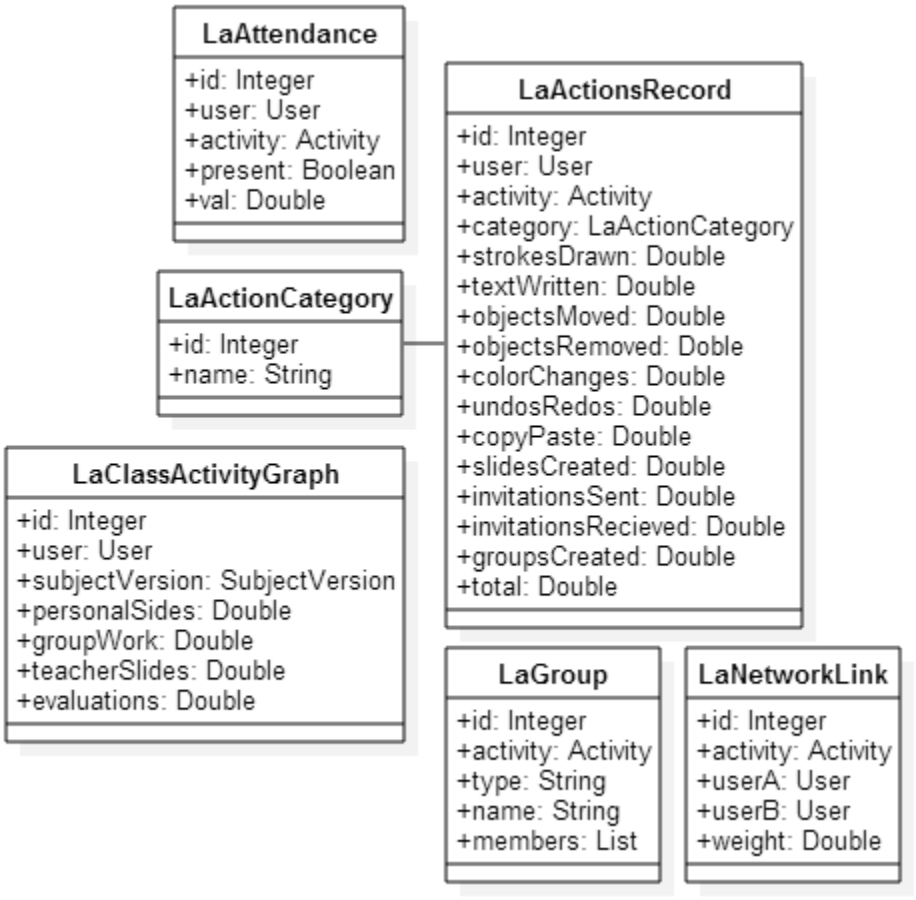


Figura 21. Modelo de datos usado para el Analytics Dashboard.

Este modelo de datos por su parte está diseñado para que los datos en ellos sean recuperados fácilmente al momento de dibujar las visualizaciones del Analytics Dashboard.

3.2.1. LaAttendance

Contiene los registros de asistencia a clases de los alumnos. Un registro de esta entidad corresponde a una celda en la visualización Attendance.

3.2.2. LaActionCategory

Categorías de LaActionsRecord. Los registros de esta entidad son insertados al momento de crear la base de datos y sus valores corresponden a “Teacher Slides”, “Personal Slides”, “Work Groups” y “Notes Groups”.

3.2.3. LaActionsRecord

Esta es la entidad más compleja de este modelo de datos y la más reutilizada a través de los servicios, es de hecho la primera tabla en llenarse al ejecutar el proceso de ETL y de esta se infiere la mayoría de las otras entidades en este diagrama. Los registros de esta entidad contienen por usuario, por categoría y por actividad, el número de acciones realizadas por cada usuario de cada uno de los tipos de acciones posibles además del total de acciones de la categoría. Esta entidad se utiliza para las visualizaciones Activity Graph, Learning Network y Actions v/s Evaluations.

3.2.4. LaClassActivityGraph

Entidad usada para el Class Activity Graph. Cada registro de esta clase corresponde a una columna del gráfico de la visualización de una de las asignaturas.

3.2.5. LaGroup

Registros de los grupos formados durante una actividad con sus integrantes. Esta entidad es utilizada en la Learning Network y su diferencia con la entidad Group del modelo de la Drawing App es que esta está optimizada para ser dibujada fácilmente usando la librería destinada para ella que será explicada más adelante.

3.2.6. LaNetworkLink

Esta entidad también es usada para la Learning Network y su atributo de importancia es “*weight*”, este atributo se utiliza para determinar el grosor de los enlaces del grafo en la visualización.

4. Controladores

Sketchpad está implementado usando el framework web Spring MVC [34]. En el punto anterior revisamos el Modelo de la aplicación, el cuál se implementa cómo entidades dentro del proyecto Spring. La definición específica de las entidades será descrita en el capítulo de implementación de Sketchpad, por ahora se debe tener presente que las entidades del modelo existen dentro del contexto web de Sketchpad y son referenciadas intensivamente en DAOs, Servicios y Controladores.

En la capa de Controladores encontramos los componentes que se denominan como DAOs, Services y Controllers. Los servicios son el conjunto de definiciones de interfaces para interactuar transaccionalmente con objetos en la base de datos que utilizan a los DAOs para realizar las transacciones.

Los controladores por su parte consumen servicios, definen rutas aceptadas por el servidor web, especifican las vistas correspondientes a cada ruta y se encargan de poner visible los datos para cada vista según los requieran.

Si se piensa a Sketchpad como una aplicación MVC, sus componentes estan divididos en la siguiente estructura:

- Model
    - Entities
    - Serializables
- Controller
    - DAOs
    - Services
    - Controllers
- View
    - JSP Pages
    - JSTL
    - Javascript
    - HTML5
    - Google Charts

En este punto se describirá el propósito y el diseño de los controladores y en el siguiente se explicarán las vistas.

4.1. DAOs

El propósito de los DAOs es bastante simple aunque la complejidad de su implementación varía dependiendo de la entidad asociada. Para cada entidad del modelo de datos se desarrolla un DAO asociado a ella cuyos propósitos son:

- Buscar registros particulares de la entidad de la base de datos
- Obtener listas de registros de la entidad en la base de datos
- Insertar registros de la entidad en la base de datos

Para esto los DAOs hacen uso de un EntityManager, definido en el contexto de aplicación Spring, sobre el cual se realizan consultas en lenguaje HQL especificando los parámetros y las consultas requeridas.

HQL corresponde a la sigla de Hibernate Query Language. Este es un mensaje de consultas similar en apariencia a SQL pero completamente orientado a objetos que entiende nociones como herencia, polimorfismo y asociación [31].

Hay ciertas consultas comunes a todos las entidades del modelo tales como las de buscar un cierto objeto por su ID o la de persistir una entidad en la base de datos. Convenientemente, para lograr este tipo de transacciones, el EntityManager ofrece funciones que realizan estas acciones sin necesidad de redactar consultas en HQL. Sin embargo, para muchas de las entidades de Sketchpad se requirió crear consultas HQL específicas de ciertas entidades con el fin de recuperar datos de manera simple en los servicios.

Ejemplos de consultas HQL usadas por Sketchpad serán vistas en la sección de implementación de DAOs más adelante.

4.2. Componentes “Controllers”

Los controladores son la capa intermedia entre las vistas y los servicios. En los controladores se declaran las rutas de las diferentes vistas y se define la lógica que se debe ejecutar para construir la vista con los datos necesarios.

Algunas de las tareas comunes de los controladores son:

- Definir rutas URL a las que los usuarios pueden acceder y la vista o tipo de respuesta obtenida.
- Validar que las solicitudes a vistas sean hechas por usuarios registrados en la aplicación y con su sesión iniciada. En caso de fallar esta validación generalmente se redireccionará al usuario de vuelta a la página de acceso o se le notificará que su sesión a expirado según el caso.
- Parsear e interpretar los parámetros recibidos por solicitud. Estos parámetros pueden venir en la ruta de la solicitud mediante el uso del caracter “?” en la ruta de solicitud en el caso de las solicitudes GET o como parámetros POST en caso de solicitudes de este tipo.
- Comunicarse con los Servicios necesarios para obtener la información requerida a mostrar en la vista específica. Cada controlador sabe para cada vista particular cuales son los servicios necesarios de usar en esta.
- Usando los datos recopilados por el o los servicios, asignarlos a una variable específica para que sea leída usada por la implementación de la vista o construir la respuesta a la solicitud como objeto JSON según el caso.
- Algunos controladores tienen por función la de crear o modificar un registro en la base de datos. Solicitudes a rutas definidas para este propósito pueden no retornan datos ni vistas sino un simple booleano que indica el éxito o fallo de la operación.

Algunas respuestas a las rutas definidas a los controladores consisten de objetos en notación JSON que son parseados luego en las vistas usando Javascript. Algunos de estos objetos JSON corresponden en Java directamente a una de las entidades del modelo de datos, pero hay otros más específicos de cada vista en los que es necesario crear un objeto Wrapper de una entidad que contiene datos adiconales además de los de la entidad.

Estos objetos son lo que en los componentes del modelo de la arquitectura MVC definimos como Serializables; objetos que implementan la interfaz Java Serializable, cuyo propósito es ser usados como respuesta a solicitudes y no existen en otra etapas

de la ejecución de la aplicación web. Por ser Serializables, estos objetos son convertidos a objetos JSON por Spring declarando en el controlador específico que la respuesta retornada debe ser de tipo “application/json”.

A continuación se verá los controladores definidos en la aplicación, describiendo sus propósitos generales y las rutas que definen con sus descripciones.

4.2.1. Controlador “UsersController”

Controlador que define las rutas asociadas a tareas de usuario tales como inicio y fin de sesión, registro por primera vez y notificación de sesiones expiradas. La ruta raíz del servidor “/” se define en este controlador ya que en esta vista se solicita al usuario iniciar sesión.

4.2.1.1. “/”

Ruta de acceso raíz del servidor. Aquí se muestra el cuadro de diálogo de inicio de sesión de cuenta FEN.

4.2.1.2. “/expired”

Esta es la vista a la que se redirige al usuario si es que intenta acceder a una vista que no sea “/” y no ha iniciado sesión en la aplicación o si es que algún error ocurre durante la navegación o el uso de la Drawing App que provoca que pierda la sincronización con el desarrollo de una clase.

4.2.1.3. “/login-fen”

Esta es una ruta especial que se le informa a la utilidad de Login-FEN para que redirija a los usuarios a ella en caso de que se identifique con su cuenta FEN exitosamente. Esta ruta recibe como parámetros los datos del usuario devueltos por el servicio de Login-FEN y registra al usuario con su sesión iniciada usando las variables de HttpSession de JavaEE.

4.2.1.4. “/logout”

Ruta a la que se envía al usuario cuando cierra su sesión haciendo click en cualquiera de los íconos de fin de sesión que se encuentran en la aplicación. Al ingresar a esta ruta se destruyen las variables de sesión del usuario haciendo que este no sea reconocido como un usuario registrado por la aplicación hasta que vuelva a iniciar sesión.

4.2.1.5. “/firstLoginRegistration”

Vista con el formulario de registro por primera vez. Los usuarios son reenviados a esta pantalla en caso de que se autentifiquen exitosamente usando su cuenta FEN y es su primera vez usando Sketchpad.

4.2.2. Controlador “SubjectsController”

Este es el principal controlador de los Selection Menus. Aquí encontramos la definición de la mayor parte de las vistas que podemos encontrar navegando los menús de selección.

4.2.2.1. “/subjectVersionSelection”

Vista de selección de asignatura. La función de este controlador es bastante simple y consiste en simplemente recuperar las asignaturas del usuario usando el SubjectVersionService y entregárselas a la vista.

4.2.2.2. “/addSubjectVersion”

Ruta de creación de asignaturas a la que solo tienen acceso profesores. Al usar la funcionalidad de crear asignaturas encontrada en la vista de selección de asignatura, se envía al usuario a esta ruta donde se realiza la operación de creación y se reenvía al usuario a la vista de selección de asignatura informándo si la operación se realizó con éxito o no.

4.2.2.3. “/registerSubjectVersionUsers”

Vista de registro de usuarios en asignatura. Esta ruta acepta los tipos de solicitudes GET y POST; usando GET la ruta devuelve la vista para registrar usuarios desde la cual al completar el ingreso de usuarios en los campos correctos se hace una solicitud a esta misma ruta pero esta vez usando POST, lo que registra los estudiantes en la asignatura y redirige a la vista de selección de asignatura informando al usuario del éxito de la operación.

4.2.2.4. “/activitySelection”

Vista de selección de actividad, similar a “/subjectVersionSelection” pero para actividades.

4.2.2.5. “/addActivity”

Ruta de creación de actividad, funciona de la misma manera que “/addSubjectVersion” pero para actividades.

4.2.3. Controlador “FilesController”

Controlador para subir y descargar archivos en el servidor. Este controlador es usado para la carga de imágenes cómo fondo de diapositivas. La subida de archivos se logra en gran parte gracias a la funcionalidad de Spring respecto al uso de MultipartFile [29].

4.2.3.1. “/upload”

Ruta que permite la carga de una sola imagen usando un parámetro de tipo MultipartFile. El archivo es entregado a UploadedFileService para ser guardado en la base de datos como una entidad UploadedFile.

4.2.3.2. “/uploadMany”

Ruta que permite la carga de varias imagenes usando un arreglo de MultipartFile. Recibe un arreglo de MultipartFile como parámetro los cuales son ordenados por nombre dentro del controlador y con ellos se construye una lista que es entregada al UploadedFileService para que sean persistidos en la base de datos.

4.2.3.3. “/files/{fileName}”

Ruta para recuperar los archivos cargados mediante upload. El controlador reconoce el nombre del archivo en la ruta de solicitud, recupera sus contenidos usando el UploadedFileService y luego imprime el contenido del archivo como respuesta a la solicitud usando un OutputStream.

4.2.4. Controlador “SketchpadController”

Controlador principal de la Drawing App. Este contiene las rutas que son solicitadas al usar la Drawing App, tanto para ingresar a ella, como para recuperar información de la clase durante el transcurso de ella. Este controlador es usado intensivamente durante el transcurso de una clase a medida que se van creando nuevas diapositivas y grupos en la sesión.

4.2.4.1. “/app”

Ruta de acceso a la Drawing App, esta es la dirección a la que se envía al usuario una vez que selecciona una actividad a la que entrar. El controlador recupera y configura todos los datos necesarios por la Drawing App tales como la SubjectVersion objetivo, la Activity objetivo y los miembros de ella.

4.2.4.2. “/app/newSlide”

Ruta que produce un JSON de respuesta de creación de una nueva diapositiva. Solicitudes a esta ruta se hacen usando AJAX con el propósito de informar que se desea crear una nueva diapositiva; llamados de este tipo se generan por usuarios que están ocupando la Drawing App y presionan uno de los botones de creación de diapositivas. El controlador verifica que la creación de la diapositiva sea posible y válida para el usuario que hace la solicitud y devuelve una entidad JSON con un identificador único de diapositiva que es utilizado para reconocer a esta dentro de las diapositivas manejadas por el COS.

4.2.4.3. “/app/newGroup”

Similar a la ruta anterior, esta ruta tiene como propósito validar la creación de un nuevo grupo por un usuario. De la misma manera, una exitosa creación del grupo retornará una entidad JSON con un identificador único de grupo para reconocer al grupo dentro de la aplicación.

4.2.4.4. “/updateMeImAlive”

Esta es una ruta que cumple una doble funcionalidad para usuarios ocupando la Drawing App. Llamados a esta ruta se generan automáticamente por la Drawing App cada un cierto intervalo de tiempo para asegurar que el usuario cliente aún tiene conexión con el servidor y que el servidor esté informado que el usuario aún está conectado. Una aplicación cliente Drawing App que falle de enviar este mensaje al servidor luego de cierto período de tiempo se considerará cómo desconectado del servidor y en caso de volver a entrar se le pedirá que reinicie su sesión. De la misma manera, una aplicación cliente que no reciba respuesta de esta solicitud será automáticamente enviado a la página de inicio de sesión por un script Javascript diseñado para asegurar que los usuarios tengan feedback de cuando pierden conexión con el servidor.

Una exitosa solicitud a esta ruta por parte de una aplicación cliente indicará al servidor que el usuario está activo, actualizando su registro de la entidad ActivityUserInstances con el tiempo del llamado en el atributo “*lastSeen*” y responderá al usuario con una lista actualizada de los usuarios conectados y desconectados a la actividad en ese instante.

4.2.4.5. “/uploadSlides”

Esta es la ruta de la vista usada para la carga masiva de imágenes. Se decidió incluir esta ruta en este controlador ya que se deben utilizar las librerías de la Drawing App para agregar las imágenes cargadas como diapositivas a la actividad.

4.2.5. Controlador “DashboardController”

Aquí se definen las rutas usadas en el Analytics Dashboard. Todas las rutas correspondientes a visualizaciones en este controlador cumplen con la misma funcionalidad, la cual consiste en realizar la lógica de interpretar los parámetros recibidos en la ruta de la solicitud GET o asignarles valores por defecto en caso de venir vacíos, luego se comunican con el servicio correspondiente entregandole los parámetros y finalmente asignando la respuesta de los servicios a variables con nombres específicos para ser recuperadas por la vista. La lógica de ordenamiento y filtrado de los datos de la visualización es realizada por los servicios correspondientes y no en esta capa de controlador.

4.2.5.1. “/dashboard”

Ruta raíz del Analytics Dashboard. Esta ruta corresponde a la vista de selección de visualización en el Dashboard.

4.2.5.2. “/dashboard/activityGraph”

Ruta correspondiente al Activity Graph.

4.2.5.3. “/dashboard/learningNetwork”

Ruta de solicitud de la Learning Network.

4.2.5.4. “/dashboard/classActivityGraph”

Ruta correspondiente al Class Activity Graph.

4.2.5.5. “/dashboard/attendance”

Ruta correspondiente a la visualización Attendance.

4.2.5.6. “/dashboard/actionsEvaluations”

Ruta correspondiente a la vista Actions v/s Evaluations.

4.3. Servicios

Servicios que se comunican con los DAOs para realizar lógica cómo autenticar usuarios, registrar alumnos en cursos, crear nuevas actividades y otros.

Con frecuencia los servicios ofrecen una interfaz de comunicación para los controladores que incluyen el acceso a los datos de la misma forma que los DAOs lo hacen, pero también agregan mayor funcionalidad que los DAOs al encargarse de procesar la información obtenida de estos según como sea necesario o de realizar varias consultas de un mismo tipo a los DAOs y agregar las respuestas a una lista u otro tipo de respuesta para ser retornado a los controladores.

4.3.1. Servicios de la aplicación: “App Services”

Estos servicios son usados en los Selections Menus y en la Drawing App. Estos hacen uso de las entidades descritas en el Modelo de datos *Drawing App* y su objetivo general es procesar los datos recogidos usando los DAOs para entregarselos a los Controllers en el formato y orden requerido por estos.

Se desarrolló un servicio por cada entidad del modelo de datos las cuales en la mayoría de los casos ofrecen a los controladores los métodos de acceso a datos ofrecidos por los DAOs más funcionalidades extras específicos de la necesidad de cada controlador.

El servicio asociado a la entidad UploadedFile es uno de los más simples de entender que explica la diferencia entre el servicio asociado y el DAO asociado. En UploadedFileDao se encuentra el método “*persist”*, el cual tiene la funcionalidad de persistir en la base de datos un simple registro de UploadedFile. Luego en el servicio UploadedFileService encontramos los métodos “*saveFile*” y “*saveFiles*”, los cuales facilitan el trabajo del controlador recibiendo los parámetros como MultipartFile en donde son transformados a entidades UploadedFile y persistiendo cada objeto haciendo llamados al método “*persist*” del DAO todas las veces necesarias.

4.3.2. Servicios de Learning Analytics: “LA Services”

Servicios que leen las acciones persistidas en la base de datos COS DB y los transforman para ser mostrados de forma gráfica en el Analytics Dashboard. En este módulo es donde se implementa la lógica de parseo de los registros en bruto de las acciones de los usuarios para interpretar las estadísticas mostradas en el Dashboard.

El proceso realizado por estos servicios es lo que es conocido como un proceso de extracción, transformación y carga (o ETL). En donde se extraen los datos de una base de datos cuyos registros deben ser procesados antes de poder ser usados en las interfaces de destino lo cual se conoce como la transformación, para finalmente estos datos transformados ser cargados en la base de datos de destino con el formato deseado.

Para el caso del Dashboard, los datos encontrados en la base de datos COS DB se alejan mucho del formato deseado para ser mostrados en el Dashboard. Las tablas usadas por el COS están pensadas con el objetivo de optimizar la replicación de los eventos y estados de una sesión entre muchos usuarios, lo que se logra mediante el uso de un modelo de datos normalizado en el que se encuentra un registro por cada mensaje y estado en las tablas. El contenido relevante en estos registros están además en un formato de texto plano de modo que sean fáciles de parsear usando Javascript ya que al servidor COS no le importa entender el contenido de los mensajes registrados, sino solamente enviarlos rápidamente a los respectivos clientes.

El objetivo de los servicios LA Services es el de obtener de la COS DB los registros de mensajes que son relevantes para las distintas visualizaciones, procesarlos contando los datos requeridos y almacenar estas estadísticas como registros en la base de datos de Sketchpad según el formato definido por las entidades del modelo de datos del Dashboard.

Para esto se desarrollaron 5 servicios encargados de hacer este trabajo, denominados LA Services, uno por cada visualización. Estos servicios son los encargados de hacer el proceso de ETL para generar los datos de las entidades del Dashboard.

4.3.3. Servicios del panel de Instrumentos: “Dashboard Services”

Estos servicios por su parte se encargan de recoger los datos de las entidades del modelo de datos del Analytics Dashboard y realizar algunas operaciones de procesamiento sobre ellas como filtrado y ordenamiento para ser entregadas a los controlador.

En el caso de estos servicios se desarrolló un servicio por cada entidad del modelo de datos del Dashboard y un servicio extra denominado “*LaService*”. Los servicios de cada entidad son los encargados de recuperar los datos necesarios de cada visualización según los filtros seleccionados en los parámetros mientras que el servicio *LaService* es el encargado de ordenar los resultados entregados por los demás servicios según el orden escogido en las visualizaciones.

5. Vistas

Las vistas son en su mayoría páginas definidas por el estándar Java Server Page (o JSP) que usan la librería "*JSP Standard Tag Library*" ó JSTL para aprovechar las características de usar una arquitectura MVC. En estas se ocupa generalmente Javascript y las características del estándar HTML5 para generar contenido.

Todas las vistas están relacionadas con un controlador el cual hace disponible ciertos datos a la vista.

La implementación de las vistas como páginas web será descrita en la siguiente sección cuando se explique la implementación de las vistas. En este punto describiremos la estrategia que se usó para diseñar la implementación de cada vista.

5.1. Vistas "Selections Menus"

Las vistas que se consideran parte de los Selections Menus son:

- Selección de Asignatura
- Selección de Actividad
- Registro de Usuario por primera vez
- Índice
- Registro de Usuarios en asignaturas
- Carga masiva de imágenes

La estrategia usada para desarrollar la implementación de estas vistas consistió en el uso de HTML plano para generar el contenido estático de las páginas en conjunto con JSTL para construir los contenidos dinámicos de las páginas tales como listas y elementos que se deben dibujar condicionalmente según el tipo de usuario u otros. Además se usó Javascript y JQuery para responder a acciones del usuario en la página tales como seleccionar un elemento u otras por el estilo.

Estas estrategias de implementación fueron usadas también en las otras vistas de la aplicación, es decir, en la Drawing App y en las visualizaciones del Dashboard. Dichas visualizaciones sin embargo, al ser un poco más complejas, utilizan además otro tipo de herramientas para implementar las vistas finales.

5.2. Vistas "Drawing App"

Es la vista donde ocurre toda lo relacionado con dibujar diapositivas, usada durante el transcurso de una clase (Figura 1). Al igual que en los Selection Menus se hace uso de HTML, JSTL y Javascript para construir el contenido pero adicionalmente, debido a la

complejidad de esta vista se debió implementar una aplicación Javascript con su propia arquitectura.

La implementación de esta vista considera la implementación de varios archivos Javascript con distintos propósitos que se dividen a grandes rasgos en archivos con los propósitos descritos a continuación. Todos estos tipos de archivos Javascript más las librerías necesarias usadas por esta vista constituyen la librería de Sketchpad necesaria para que funcione la Drawing App como aplicación Javascript.

5.2.1. Clases Lógicas

Archivos que describen objetos lógicos cuyos estados y atributos son usados durante el transcurso de la clase y sobre los que se realizan operaciones. Algunas de estos objetos lógicos son acoplados al servidor COS y otros no.

El uso de estos objetos internamente en la aplicación se trabajó por convención cómo si fuesen objetos Java en los que los atributos son considerados como atributos privados, de modo que se evita acceder a ellos directamente por las propiedades de objetos Javascript y en cambio se consideran a las funciones del objeto como las únicas interfaces válidas de acceso a ellos.

Esto convenientemente facilita la notificación de cambios en el estado del objeto ya que se utiliza la convención de que al ocurrir cambios de estado en este tipo de objetos, estos disparan un evento con un nombre específico con el propósito de que se adjunten escuchadores a estos tipos de eventos que se encarguen de actualizar las representaciones gráficas de los objetos. De esta manera por ejemplo, cada vez que se actualiza el estado de una diapositiva se dispara un evento que es escuchado por una función que ejecuta la lógica de repintar la diapositiva en la vista acorde a su nuevo estado.

5.2.2. Actualizadores de UI

Estas se constituyen de un conjunto de funciones Javascript cuyo propósito es, recibiendo el nuevo estado de un objeto como parámetro, actualizar la representación gráfica del objeto en la vista modificando los elementos HTML necesarios para dibujar dicha representación.

Los archivos de este tipo contienen una gran cantidad de funciones utilitarias diseñadas para facilitar el proceso de formar el HTML usado en la representación gráfica de los objetos en la UI. Estas funciones utilitarias son usadas en las funciones específicas encargadas de actualizar los objetos tal como a nivel de arquitectura de la aplicación web Java los controladores utilizan un conjunto de servicios para construir los datos necesarios por las vistas. Las funciones encargadas de actualizar los objetos leen el

estado del objeto, interpretan los atributos de este y sus valores y generan un nuevo código HTML o una nueva propiedad CSS que reemplazará la anteriormente seteada en el lugar indicado de la visualización.

Veremos que para encontrar los elementos HTML a actualizar o reemplazar se usará principalmente JQuery ya que sus funcionalidades permiten realizar este tipo de operaciones rápida y fácilmente.

5.2.3. Escuchadores

Los escuchadores son los que hacen el nexo final entre las funciones de actualización y los objetos lógicos. En la implementación de la aplicación Javascript se procura que luego de la creación de cada elemento lógico cuyos cambios deben verse reflejados en la interfaz de usuario, a este se le asigne el o los escuchadores correspondientes encargados de llamar a la o las funciones de actualización de UI respectivas.

De este modo por ejemplo cada vez que se crea una nueva diapositiva exitosamente, a esta se le adjuntan varios escuchadores específicos que se encargan de correr funciones de actualización cuando ocurren acciones como que se dibujen trazos nuevos, se hagan cambios de colores, ingresen nuevos usuarios a trabajar en la diapositiva o que la diapositiva sea eliminada entre otros.

5.3. Vistas del panel de Instrumentos: “Dashboard Visualizations”

Cada visualización se implementa mediante una página JSP que hace uso de las mismas funcionalidades encontradas en los Selection Menus.

En las visualizaciones del Analytics Dashboard se utiliza JSTL para generar contenido dinámico pero mayormente para iterar sobre listas de datos entregadas por los controladores, construyendo tablas o arreglos de datos Javascript que son usados por las librerías Google Charts o halfviz para dibujar la visualización deseada.

Se ocupa Javascript para implementar el uso de los selectores de filtros adjuntando escuchadores de eventos a los distintos selectores que al activarse redirigen al usuario a la visualización que considera los nuevos parámetros seleccionados.

A excepción de la visualización “*Attendance*”, la iteración sobre los datos entregados por los controladores se usa para configurar los datos obtenidos de manera que sean apropiados para la librería de visualización usada, en el formato que la librería espera. En el caso de los gráficos construidos usando Google Charts se deben construir arreglos con los datos obtenidos por el controlador. La visualización Learning Network

por su parte ocupa halfviz para dibujar el grafo, el cual usa un parseador de texto para construir el grafo. Para la visualización Attendance en cambio se utiliza sólo HTML para hacer el dibujo.

Luego las librerías o estrategias usadas para construir cada visualización son:

- **Activity Graph**: Google Charts
- **Class Activity Graph**: Google Charts
- **Learning Network**: halfviz
- **Attendance**: HTML
- **Actions v/s Evaluations**: Google Charts

VII. Implementación

La implementación de Sketchpad se realizó en un proceso iterativo durante el transcurso de alrededor de un año. El diseño de aplicación descrito en las secciones anteriores corresponde al diseño definitivo en el que quedaron implementadas las funcionalidades, pero a este diseño se llegó después de varias iteraciones de implementación de funcionalidades y conversaciones con los usuarios finales de la aplicación.

El proceso de implementación de funcionalidades de la Sketchpad web-app se desarrolló principalmente usando el entorno de desarrollo en el IDE proporcionado por Spring para el desarrollo de proyectos: Spring Tool Suite. La mayoría de íconos e imágenes usadas en las vistas fueron principalmente proporcionados por el cliente de acuerdo a sus criterios para las interfaces de usuario.

En esta sección se describirá la implementación de los tipos de componentes encontrados en la aplicación, intentando no describir la implementación una a una de cada elemento, entidad o clase, sino entregando una idea general de cómo se implementa un elemento del tipo descrito de manera que el lector pueda inferir luego la implementación de cada elemento en particular basado en la descripción del elemento dada en la sección anterior de diseño.

Cabe mencionar que se realizó un proceso de implementación de todo el sistema en un servidor de producción encontrado en la Facultad de Economía y Negocios. Esta implementación incluyó la instalación de un servidor Tomcat 7 y un servidor Postgres 9 en una máquina iMac Mini bajo un sistema operativo OSX. Este computador se configuró como servidor siendo programado para mantenerse siempre encendido asignado a una IP específica entregada por los administradores de red de la FEN y con ambos servidores Tomcat y Postgres iniciándose al encender la máquina y escuchando en sus respectivos puertos.

Consecuentemente se instaló Sketchpad y COS con sus respectivas bases de datos en dicho servidor. Adicionalmente, las tablets iPads destinadas a ser usadas en la sala de clases fueron configuradas con un acceso directo a la URL de entrada de la aplicación en el servidor de producción. Dicha URL es sólo accesible usando una conexión a internet desde dentro de la FEN, accediendo a la IP específica del servidor dentro de la red interna de la facultad.

1. Bases de Datos

La implementación de las bases de datos, tanto durante el desarrollo de la aplicación cómo al momento de instalarse en el servidor de producción, se realizó instalando una instancia del servidor PostgreSQL 9.

En este servidor PostgreSQL se crean las bases de datos “sketchpad_bd” y “cos”, correspondientes a Sketchpad y al COS respectivamente. Luego se asignan usuarios con privilegios de lectura y escritura sobre cada base de datos. Estos usuarios y sus credenciales constituyen datos de conexión a la base de datos que deben ser configurados en el descriptor de contexto web de Spring para que el EntityManager de Spring pueda conectarse y realizar consultas a las bases de datos.

El paso final en la implementación de las bases de datos consiste en crear las tablas de cada base de datos y poblarlas con los datos básicos necesarios para el correcto funcionamiento de Sketchpad y COS. Para esto, se redactaron archivos de texto plano con las instrucciones en SQL necesarias para crear las tablas en las bases de datos necesarias y poblarlas. Durante el proceso de desarrollo se debió mantener consistencia siempre entre estos archivos de texto y el estado del modelo de datos usado por la aplicación en las iteraciones de desarrollo que significaron un cambio en el modelo.

1.1. Script: “sketchpad.sql”

Este archivo de texto contiene las instrucciones de creación de tablas de la base de datos sketchpad_bd. El siguiente es un extracto de instrucciones encontradas en este archivo.

```
drop table if exists sketchpad_bd.public.activity;

create table sketchpad_bd.public.activity (
    id serial not null primary key,
    name varchar(255) not null,
    date timestamp null,
    subject_version_id int not null,
    author_user_id int null
);

alter table sketchpad_bd.public.activity
    add constraint fk_activity_subject_version
    foreign key (subject_version_id)
    references sketchpad_bd.public.subject_version(id)
    on update restrict
```

```
    on delete cascade;

alter table sketchpad_bd.public.activity
    add constraint fk_activity_author
    foreign key (author_user_id)
    references sketchpad_bd.public.user_(id)
    on update restrict
    on delete set null;
```

Figura 22. Extracto de archivo sketchpad.sql que incluye instrucciones para crear tabla *activity* y setear llaves foráneas en ella.

En este extracto se puede ver en particular las instrucciones para crear la tabla activity junto con las instrucciones para setear sus llaves foráneas y restricciones de campos. Las demás tablas de la base de datos se crean de esta misma forma.

1.2. Script: “cos.sql”

Este archivo contiene las instrucciones para crear las tablas necesarias por el COS para funcionar. Tiene una estructura similar a la de sketchpad.sql. Este archivo en particular fue modificado de su versión original del COS para incluir campos necesarios para las estadísticas del Analytics Dashboard.

1.3. Script: “seeds.sql”

Contiene instrucciones de llenado de tablas con datos de utilidad general de la aplicación tales cómo usuarios de prueba de acceso privado a la aplicación y una asignatura genérica necesaria para usar en la lista de asignaturas.

```
insert into
sketchpad_bd.public.user_(username,name,lastname,type,mail) values
  ('pgz1','Administrador','1','admin','administrador@fen.uchile.cl'),
    ('pgz2','Profesor','1','teacher','profesor@fen.uchile.cl'),
    ('pgz3','Alumno','1','student','alfonso@fen.uchile.cl'),
    ('pgz5','Monitor','1','tv','monitor1@fen.uchile.cl'),
    ('pgz6','Alumno','2','student','student2@fen.uchile.cl'),
    ('pgz7','Alumno','3','student','student3@fen.uchile.cl'),
    ('pgz9','Alfonso','Cornejo','teacher','alfunkso@hotmail.com')
;
insert into sketchpad_bd.public.subject(name) values
    ('Curso Generico')
;
```

```
insert into sketchpad_bd.public.subject_version(name, date,
subject_id) values
    ('Curso de Prueba',CURRENT_TIMESTAMP,1)
;

insert into
sketchpad_bd.public.activity(name,date,subject_version_id,
author_user_id) values
    ('Clase de Ejemplo',CURRENT_TIMESTAMP,1,2)
;
```

Figura 23. Extracto de archivo seeds.sql.

1.4. Script: “testData.sql”

Un archivo muy similar en formato a seeds.sql pero usado específicamente para poblar las tablas con datos de prueba. Los datos con los que llena las tablas este archivo fueron recopilados de una de las sesiones de pruebas que se realizaron con usuario.

Este es un archivo que no se puede usar en el servidor de producción ya que invalida los datos de la base de datos con información desactualizada y su único propósito es testear situaciones en el proceso de desarrollo de nueva funcionalidad.

2. Componentes Aplicación Web

Si bien todas las componentes de la aplicación que no son la implementación de las bases de datos fueron desarrolladas e implementadas en el contexto de la misma aplicación web, en este punto se explicará la implementación de Entidades, DAOs, Servicios y Controladores, dejando la implementación de las vistas como un punto aparte debido a la complejidad y profundidad que presentan algunas de las vistas, específicamente la Drawing App y las visualizaciones del Analytics Dashboard.

2.1. Entidades

Por cada una de las entidades del modelo de datos se creó una clase Java que implementa la interfaz Serializable y en estas se definieron las reglas de mapeo de las clases con sus correspondientes tablas en la base de datos usando las anotaciones estándar de persistencia usadas por Spring.

En el siguiente ejemplo se ve un extracto de la definición de la clase User con sus anotaciones y atributos.

```
@Entity
@Table(name = "user_")
@JsonIgnoreProperties({"subjectVersions"})
public class User implements Serializable {
  @Id
  @GeneratedValue(strategy = GenerationType.IDENTITY)
  private Integer id;

  private String username;
  private String name;
  private String lastname;
  private String type;
  private String mail;

  @ManyToMany(fetch = LAZY, cascade = CascadeType.ALL)
  @JoinTable(name="subject_version_user",
    joinColumns = { @JoinColumn(name="user_id") },
    inverseJoinColumns = @JoinColumn(name="subject_version_id")})
  @OrderBy("name")
  private List<SubjectVersion> subjectVersions;
```

Figura 24. Extracto de la clase User.

La anotación *Entity* al comienzo de la clase le indica al contexto Spring que esta clase representa una entidad del modelo y por ende los objetos de esta clase deben ser considerados por el contexto de persistencia del servidor para ser persistibles.

La anotación *Table* indica la tabla de la base de datos a la que corresponde esta clase. La anotación *JsonIgnoreProperties* es utilizada para indicarle al motor de FasterXML los atributos del objeto que no debiesen ser serializados en un JSON a no ser que se explicite lo contrario. El motor de FasterXML es usado por Spring en la ejecución de algunos controladores que especifican que la respuesta a solicitudes de una cierta ruta corresponden a un objeto del tipo “application/json”. Cuando esto sucede, si el objeto es efectivamente Serializable, Spring imprime como respuesta la transformación a formato JSON del objeto en lugar de retornar una vista. En el caso de este ejemplo vemos que la lista de asignaturas del usuario no será incluida por defecto en una representación JSON de este objeto.

Luego se definen las reglas de mapeo de todos los atributos del objeto con respecto a su tabla y a las tablas que están relacionadas con ella en caso de que haya referencia por llaves foráneas.

En el ejemplo vemos que el número ID de estas entidades consiste de un entero correspondiente a la columna “*id*” de la tabla en la base de datos y que la estrategia para generar un nuevo número de ID es usar el siguiente valor en la secuencia de ids disponibles de la entidad. Esta secuencia de ids es creada automáticamente en PostgreSQL al identificar el tipo de uno de los atributos de la tabla como “*serial*”. Spring luego usa esta estrategia al crear nuevos registros a persistir en la base de datos. Todas las entidades de Sketchpad vistas en los distintos modelos de datos ocupan esta misma estrategia de generación de ID.

Por último vemos el uso de una anotación *ManyToMany*. Cómo esta también existen las anotaciones *OneToMany*, *ManyToOne* y *OneToOne*. Estas anotaciones son usadas para denotar referencias de atributos que corresponden a entidades de otro tipo dentro de esta entidad. En el caso del ejemplo ocurre uno de los poco frecuentes casos en los que se utiliza *ManyToMany* para anotar la referencia. Este tipo de relaciones requiere el uso de una tabla intermedia que conoce las relaciones entre Users y SubejctVersions, en este caso llamada “subject_version_user”.

Cómo este ejemplo se repite el uso de estas anotaciones en varias implementaciones de las entidades del modelo de datos. El lector podrá inferir que entidades requieren qué tipo anotaciones según las relaciones indicadas en el modelo de datos y sus atributos.

2.2. DAOs

Estos se implementan usando 2 tipos Java, primero se define la interfaz del DAO, la cual define cómo será la comunicación entre los servicios y estos DAOs. A continuación se puede ver un ejemplo de definición de la interfaz UserDAO.

```
public interface UserDAO {
  public User getFenUser(String username);
  public User findById(Integer id);
  public User findByMail(String mail);
  public User findByUsername(String username);
  public List<User> getAll();
  public void persist(User user);
}
```

Figura 25. Definición de interfaz DAO para entidad User.

Se crea una interfaz DAO por cada entidad del modelo de datos. Todos los DAOs incluyen las funciones básicas: *findById*, *getAll* y *persist*, y muchos de ellos implementan además métodos específicos de cada entidad como se puede ver en este caso para el UserDAO en el que se incluyen los métodos particulares: *getFenUser*, *findByMail* y *findByUsername*. Estos métodos son usados en los distintos servicios disponibles según se requieran.

Luego se crean las clases implementaciones de estos DAOs, las cuales implementan las interfaces de los DAOs y cumplen la función definida en las interfaces accediendo al EntityManager del contexto de persistencia de Java y usando HQL según sea necesario. A continuación se ve la implementación de dos métodos de la interfaz UserDAO, uno que requiere HQL para completarse y uno en el que solo basta con una función del EntityManager.

```
@Repository
public class UserDAOImpl implements UserDAO {

  @PersistenceContext
  private EntityManager em;

  @Override
  public User findById(Integer id) {
    return em.find(User.class, id);
  }
```

```
  @Override
  public User findByUsername(String username) {
       TypedQuery<User> query = em.createQuery("from User u where
u.username= :username", User.class);
       query.setParameter("username", username);
       List<User> result = query.getResultList();
       return result.size() > 0 ? result.get(0) : null;
   }
}
```

Figura 26. Extracto de la implementación de interfaz DAO para entidad *User*. En el método *findByUsername* se puede ver el uso de HQL para generar la query requerida.

Al igual que en la implementación de entidades, para la implementación de DAOs se ocupan ciertas anotaciones con el propósito de entregar cierta información al contexto de Spring respecto a esta clase.

La anotación *Repository* identifica esta clase como la definición de un repositorio de datos esto es interpretado por Spring al inicializar su contexto lo que produce que este instancie un objeto único de esta clase el cual luego es accesible por los servicios que lo necesiten al usar la anotación *Autowired*.

La anotación *PersistenceContext* definida sobre el atributo EntityManager em le indica a Spring que al instanciar el objeto de esta clase debe asignarle el contexto de persistencia a esta variable para que esté disponible para los métodos que hacen uso de ella internamente. El EntityManager es usado en todos los DAOs sin excepción.

2.3. Servicios

Un servicio es similares a un DAO en la forma en las que es implementado, también se define su interfaz inicialmente y luego se define su implementación.

La mayoría de los servicios definen en sus interfaces los mismos métodos que los de sus correspondientes DAOs, pero algunos particulares incluyen métodos adicionales específicos que implementan lógica de aplicación más compleja que la encontrada en los DAOs. Esto se hace con el propósito de que los controladores correspondientes no se comuniquen directamente con los DAOs, sino que consuma solamente servicios para todas las transacciones que necesita.

De esta forma, se creó un servicio por cada entidad del modelo de datos además de ciertos servicios adicionales no relacionados específicamente con ninguna entidad sino

con un conjunto de ellas. A continuación se muestra un ejemplo de definición de una interfaz de servicio.

```
public interface SubjectVersionRegistrationService {
  public List<SubjectVersionRegistration> getAll();
  public void persist(SubjectVersionRegistration
subjectVersionRegistration);
  public void persistMailRegistrations(List<String> mails, SubjectVersion
subjectVersion);
  public HashMap<SubjectVersion, List<User>> assignSubjectVersionUsers();
}
```

Figura 27. Definición de interfaz de servicio de entidad SubjectVersionRegistration.

Y un ejemplo de un método de su correspondiente implementación.

```
@Service
public class SubjectVersionRegistrationServiceImpl implements
SubjectVersionRegistrationService {
  @Autowired private SubjectVersionRegistrationDao subjectVersionRegistrationDao;
  @Override
  public void persistMailRegistrations(List<String> mails, SubjectVersion
subjectVersion) {
        SubjectVersionRegistration svr = null;
        for ( String mail : mails ) {
              svr = new SubjectVersionRegistration();
              svr.setSubjectVersion(subjectVersion);
              svr.setUserMail(mail);
              this.subjectVersionRegistrationDao.persist(svr);
        }
  }
```

Figura 28. Implementación del servicio de la entidad SubjectVersionRegistration.

Aquí nuevamente se ve el uso de ciertas anotaciones usadas por Spring. La anotación *Service* cumple un rol similar al de la anotación *Repository* que se mencionó en la implementación de los DAOs. Esta anotación permite que se tenga acceso a una instancia de esta clase en los controladores. Por su parte la anotación *Autowired* es otra anotación que le indica a Spring que el atributo indicado debiese ser asignado a una instancia de la clase indicada para poder tener acceso a él durante la ejecución de alguno de sus métodos. Encontraremos la anotación *Autowired* en varios servicios y controladores cumpliendo esta función.

Se debe mencionar en este punto que la implementación de la lógica más compleja necesaria para generar las vistas del Analytics Dashboard se realizan en esta capa de servidores. En el caso particular de los servicios que recuperan los datos de las entidades del modelo de datos del Dashboard se encuentra que se desarrolló un servicio por cada entidad del modelo los cuales tienen la función de recuperar los datos requeridos usando los DAOs, procurando que el resultado entregado a los controladores tome en cuenta los filtros y consideraciones especificadas. De manera que las firmas de los métodos encontrados en este tipo de servicios son similares a la que se puede ver en la figura 29 sobre *actionsVsEvaluations*, considerando los parámetros de filtrado.

```
@Override
public List<ActionsVsEvaluationsEntry> actionsVsEvaluations(
              Integer subjectVersionId,
              Integer activityId,
              Integer evaluationId,
              String actionType) { … }
```

Figura 29. Firma de uno de los métodos encontrados en el servicio LaActionsEvaluationsService que incluye parámetros para las opciones de filtrado disponibles.

Adicionalmente se desarrolló un servicio llamado "*LaService*" encargado de ordenar los registros finales obtenidos de los servicios de las entidades según el criterio de ordenación escogido en las distintas visualizaciones. Esto se implementó de esta forma para evitar la acomulación de mucho código en un mismo servicio, extrayendo parte de la lógica a otro servicio.

2.4. Controladores

En los controladores se definen las rutas que el servidor aceptará y cómo responderá a cada una. Consumen los servicios y definen las variables que las vistas utilizarán.

La siguiente es la definición de la ruta raíz del servidor.

```
@Controller
public class UsersController {
  @Transactional
  @RequestMapping(value = "/", method = { GET })
  public String index(Model model) {
        model.addAttribute("hostName", HttpUtil.getHostName(request));
        return "index";
```

```
}
```

Figura 30. Definición de la ruta raíz en UsersController.

Como se puede ver, la anotación *Controller* se usa para hacerle saber al contexto de Spring que este clase debe ser escaneada por definiciones de rutas y que debe considerar el contexto de la aplicación para tener acceso a las instancias de servicios y otros recursos administrados por Spring.

La anotación *Transactional* indica que esta ruta puede acceder a la base de datos y por lo tanto se necesitará hacer uso de una sesión de conexión con esta que se debe mantener abierta hasta el momento en el que el método del controlador retorna.

La anotación RequestMapping le indica a Spring la ruta a la que se espera que este método sea mapeado y qué tipo de solicitud HTTP debe aceptar, GET ó POST.

El parámetro model que se recibe en los métodos permite definir variables para que sean visibles en la vista. En este caso se define el atributo hostName que podrá ser accedido desde la vista, y se define que la vista que se desplegará en esta ruta será un archivo que tendrá el nombre “index.jsp”. Spring espera que en momento de ejecución, dentro de los archivos encontrados en el contexto web se encuentre uno con el nombre “index.jsp”; de ser esto efectivo, cuando el método del controlador retorna, Spring entrega los contenidos del archivo cómo respuesta de la solicitud luego de procesar las instrucciones Java que se encuentren en este.

```
@Transactional(readOnly = true)
@RequestMapping(value = "/dashboard/classActivityGraph", method = GET)
public String classActivityGraph(
    HttpServletRequest request,
    Model model,
    @RequestParam(required=false) Integer subjectVersionId,
    @RequestParam(required=false) String order
) {
    User user = this.userService.getUser(request);
    if ( user == null || user.getId() == null )
           return "redirect:/expired";
    model.addAttribute("user", user);

    List<SubjectVersion> subjectVersions = user.getSubjectVersions();
    Hibernate.initialize(subjectVersions);
    model.addAttribute("subjectVersions", subjectVersions);

    if ( subjectVersionId == null ) {
           Integer lastSubjectId = -10;
           for ( SubjectVersion subject : subjectVersions )
```

```
                    lastSubjectId = Math.max(lastSubjectId, subject.getId());
                subjectVersionId = lastSubjectId;
        }
        model.addAttribute("subjectVersionId", subjectVersionId);

        SubjectVersion subject =
this.subjectVersionService.findById(subjectVersionId);
        model.addAttribute("subject", subject);

        ClassActivityGraphDetail detail =
this.laService.classActivityGraph(subjectVersionId, order);
        model.addAttribute("order", order);
        model.addAttribute("entries", detail.getEntries());
        model.addAttribute("max", detail.getMax());

        return "dashboard/classActivityGraph";
    }
```

Figura 31. Definición de la visualización Class Activity Graph dentro del controlador del Analytics Dashboard. Aquí se puede ver llamados a los servicios UserService y LaService entre otros.

3. Implementación de “Selection Menus”

La implementación de las vistas de los Selection Menus es una de las más simples de entre las implementaciones de las vistas, se utiliza HTML, y las conveniencias de las instrucciones Java permitidas por los estándares JSP y JSTL. También se implementan algunas funcionalidades con Javascript dependiendo del caso.

A modo de ejemplo veremos la implementación de la vista de selección de asignatura, cuya implementación es muy similar a la implementación de las demás vistas consideradas como parte de los Selection Menus.

La implementación de esta vista consiste de un archivo formato JSP encontrado en los recursos del contexto de la aplicación web. Este es en su mayor parte un archivo HTML estándar que con la ayuda de JSTL importa algunos archivos adicionales para que sus contenidos aparezcan en el código fuente de la página web resultante.

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<!DOCTYPE html>
<html lang="en">
  <head>
    <c:set value="${pageContext.request.contextPath}" var="basePath"></c:set>
    <jsp:include page="include/head.jsp"></jsp:include>
    <jsp:include page="include/js.jsp"></jsp:include>
  </head>
```

Figura 32. Extracto de la página “subjectVersionSelection.jsp”.

En la figura 32 se ve un extracto de la primera parte de la página. La instrucción de la primera línea indica que usaremos JSTL en esta página usando el prefijo “*c*”. En la sección <head> de la página declaramos una variable “*basePath*” que usaremos para referenciar la ruta de contexto de Sketchpad. Luego se importan los archivos “*head.jsp*” y “*js.jsp*”, los cuales incluyen líneas que declaran cosas como el título de página, el favicon, el charset y los estilos CSS usados además de las librerías Javascript usadas en estos menús.

El cuerpo de la página se construye de la misma forma, usando HTML para formar los elementos mostrados en la vista. Cuando se llega al momento de construir la lista de asignaturas del usuario se utiliza el dinamismo de JSTL para iterar por la lista de asignaturas y construir los elementos de la forma mostrada en la figura 33.

```
<legend>Select subject</legend>
  <div>
    <c:if test="${empty subjectVersions}">
      Your subjects list is empty. This might be because you haven't been assigned
to any subject yet or neither of your subjects uses Sketchpad.
    </c:if>
    <ul id="subjectlist" class="panelList nav nav-pills nav-stacked">
      <c:forEach var="subjectVersion" items="${subjectVersions}">
        <li data-id="${subjectVersion.id}" style="position: relative;">
          <a href="#" class="subjectVersionA" data-id="${subjectVersion.id}">
            <div>
              <img class="pull-left img-rounded" style="margin-right:10px"
src="${basePath}/resources/images/icons/subject.png">
                <div>
                  <strong>${subjectVersion.name}</strong>
                  <br/>
                  <small>  </small>
                </div>
              </div>
            </a>
            <c:if test="${user.type eq 'teacher'}">
              <span class="label label-default add-user-icon" data-
id="${subjectVersion.id}">
                <img src="${basePath}/resources/images/icons/add-users.png">
              </span>
            </c:if>
          </li>
        </c:forEach>
      </ul>
      <c:if test="${user.type eq 'teacher'}">
        <span class="btn" data-toggle="modal" data-target="#add-group-modal">
          <i class="icon-plus"></i>  Add new subject version
        </span>
      </c:if>
```

Figura 33. Generación de elementos HTML de la lista de asignaturas en la vista de selección de asignatura.

Finalmente se deben agregar escuchadores al evento de que el usuario seleccione una de las asignaturas haciendo click en ellas. Para esto se utiliza Javascript y JQuery de la forma que se muestra en la figura 34.

```
<script>
  $(function() {
    $(".subjectVersionA").click( function() {
      window.location = '${basePath}/activitySelection/' + $(this).data("id");
    });
    $('.add-user-icon').click( function() {
      window.location =
'${basePath}/registerSubjectVersionUsers/'+$(this).data('id');
    });
```

```
  });
</script>
```

Figura 34. Asignación de funciones escuchadores al evento de click sobre uno de los elementos de la lista en la vista de selección de asignatura.

Las estrategias usadas para la implementación de esta vista se repiten en las demás implementaciones de vistas, no solo de los Selection Menus, sino también en algunos contenidos encontrados en la Drawing App y más frecuentemente en elementos encontrados en las visualizaciones del Analytics Dashboard.

4. Implementación de “Drawing App”

La mayor parte de la funcionalidad de la Drawing App está implementada usando Javascript. Se ocupó un set de librerías y se desarrolló una serie de archivos con la funcionalidad necesaria. Se explicará cómo se distribuyó este código y qué utilidades específicas tiene cada archivo.

4.1. Librerías

- **JQuery**: Utilidades Javascript.
- **JQuery-UI**: Utilidades gráficas.
- **JQuery Simple Color**: Seleccionador de color.
- **JQuery Tab Slide Out**: Utilidad para esconder paneles laterales.
- **JQuery Base64**: Codificar imágenes en base 64.
- **Bootstrap**: Utilidades Javascript y gráficas.
- **Underscore**: Utilidades Javascript.
- **Raphael**: Librería para dibujar sobre el elemento HTML5 SVG.
- **Canvg**: Librería usada para poder descargar diapositivas.
- **FileSaver**: Librería para guardar diapositivas como archivos.
- **ObjectID**: Librería utilitaria para crear objetos con IDs únicos.
- **CoupledObjectServer**: Librería que se encarga de hacer la sincronización entre objetos acoplados colaborativamente.

Esta última librería corresponde a la librería del COS mencionada previamente. Se encarga de hacer toda la sincronización entre los usuarios y es la que en definitiva permite hacer la distribución de contenido en tiempo real a todos los usuarios y permite trabajar colaborativamente al mantener todos los objetos de una actividad sincronizados en sus estados según lo que los usuarios vayan haciendo en en el transcurso de la clase.

Para lograr esto, a algunos objetos específicos de Sketchpad se les da un identificador único de la aplicacion que permite acoplar las instancias de objetos en distintas máquinas de distintos usuarios con el mismo identificador sincronizado en un virtual único objeto compartido por todos los usuarios conectados al mismo tiempo.

Esta librería se encarga también de persistir los cambios que se realizan sobre los objetos acoplados de modo que usuarios que se integran a una sesión de forma posterior a su incio o su estado inicial estén al tanto de todas las actualizaciones por las

que ha pasado el objeto y por ende tengan acceso a la última versión del estado del objeto.

4.2. Estructura

Todo el código Javascript desarrollado fue ordenado en carpetas según su utilidad. La estructura de estos directorios y archivos es la indicada a continuación.

- js
    - logicClasses
        - editorTools
            - CheckboxTool.js
            - NullTool.js
            - PanTool.js
            - PenTool.js
            - SelectTool.js
            - TextTool.js
            - Tool.js
            - ToolOptions.js
        - sketchpadElements
            - CheckboxElement.js
            - Element.js
            - ElementFactory.js
            - Question.js
        - Canvas.js
        - Group.js
        - History.js
        - Model.js
        - Room.js
        - Sketchpad.js
        - UserList.js
    - handlers.js
    - logic.js
    - ui.js

Todos estos archivos son importados en la vista de la Drawing App en un orden específico. Explicaremos brevemente las funcionalidades generales de cada archivo.

4.2.1. editorTools

Los archivos encontrados en esta carpeta contienen la funcionalidad de cada una de las herramientas disponibles para trabajar sobre diapositivas.

4.2.2. sketchpadElements

Archivos utilitarios de Sketchpad para crear y manejar elementos de la diapositiva.

4.2.3. Canvas.js

Definición de los objetos de la clase Canvas, estos objetos contienen la lógica para dibujar vistas en miniatura de las diapositivas para ser vistas en la barra lateral y la lógica para actualizar estas vistas según los cambios que se hagan en el dibujo.

4.2.4. Group.js

Clase lógica de los grupos, contiene la información de los integrantes del grupo y las diapositivas que el grupo trabaja colaborativamente. Los objetos de esta clase son acoplados en el COS.

4.2.5. History.js

Contiene el historial de elementos que se van dibujando en la diapositiva y permite deshacer y rehacer los cambios que se van haciendo. Los objetos de esta clase son acoplados en el COS.

4.2.6. Model.js

Este archivo se encarga de hacer la comunicación entre todos los mayores objetos de la aplicación, incluyendo el objeto Sketchpad y las diapositivas.

4.2.7. Room.js

Clase lógica que se encarga de operar sobre las entidades de una diapositiva. Mientras que los objetos de la clase Canvas contienen la lógica para dibujar en pantalla las diapositivas, los objetos de clase Room conocen la lógica para agregar elementos en diapositivas y notifican a su objeto de clase Canvas asociado que haga la actualización de las vistas.

4.2.8. Sketchpad.js

Esta clase define al objeto Sketchpad, el cual está disponible transversalmente como una variable global en toda la extensión de la aplicación y es la contenedora de todos los elementos de la aplicación. Contiene métodos utilitarios para crear y encontrar

diapositivas, encontrar la diapositiva activa, encontrar y cambiar la herramienta de trabajo actual, y otros.

4.2.9. UserList.js

En este archivo se define la lógica para manipular usuarios, actualizar el estado de conexión de estos a la aplicación y además contiene la información de cada uno de ellos en un arreglo ordenado.

4.2.10. handlers.js

En este archivo se encuentra una serie de funciones que se ejecutan como callbacks de eventos que ocurren en los distintos objetos de la aplicación. Algunos de estos callbacks tienen que ver con actualizaciones de vistas y otros.

4.2.11. logic.js

Lógica para iniciar la aplicación correctamente.

4.2.12. ui.js

Una serie de funciones relacionadas con la interfaz gráfica de la aplicación.

En el diseño de la Drawing App como aplicación Javascript, las diapositivas se corresponden con el objeto lógico *Room*. Un objeto Room representa una diapositiva a la que los usuarios se unen al seleccionar su mini-vista de diapositiva asociada como diapositiva principal. Todas las *Rooms* tienen asociados un objeto *History* como uno de sus atributos, este objeto es el que contiene la información de los elementos dibujados sobre la diapositiva y es el objeto que se acopla al COS para mantener sincronizado.

```
Skt.Room = L.Class.extend({
  includes: L.Mixin.Events,

  initialize: function(id, type, ownerId){
        this._id = Number(id);
        this._type = type;
        this._ownerId = Number(ownerId);

        this._objects = {};
        this._background = null;

        /* set History */
        var self = this;
        var execute = function(cmd) {
              self.execute.call(self, cmd);
        };
```

```
        var rollback = function(cmd) {
              self.rollback.call(self, cmd);
        };

        this._history = new Skt.History(this, execute, rollback);
    },
```

Figura 35. Definición del objeto lógico Room de la aplicación Javascript.

En la figura 35 se ve la definición de los objetos lógicos *Room* de la aplicación Javascript. Aquí se ve la forma en la que se definen los atributos del objeto, los cuales en el ejemplo son _id, _type, _ownerId, _objects, _background y _history. Se utiliza la convención de asignar los nombres de atributos con una barra abajo antes del nombre para indicar que estos son atributos privados que a los que no se debiese acceder desde fuera del objeto. Además se usa la convención de definir todas las clases relevantes a Sketchpad dentro de un objeto especial asignado en la variable de documento "*Skt*".

En el archivo *handlers.js* encontramos lógica de actualización de la interfaz gráfica según los cambios que se registren en los objetos.

```
Skt.UsersSidebar.handleGroupsUpdated = function(params) {
    setTimeout(function(){
        Skt.UsersSidebar.updateGroups(Skt.sketchpad);
    }, 100);
};

Skt.UsersSidebar.updateGroups = function(sketchpad) {
    var user = sketchpad.getUser();
    var workGroups = sketchpad.getWorkGroups();
    var notesGroups = sketchpad.getNoteGroups();

    if ( workGroups.length > 0 ) {
        var html = "";
        for ( var i = 0; i < workGroups.length; ++i )
              html += Skt.UsersSidebar.buildGroupHtml(workGroups[i]);
        $("#work-groups-list").html(html);
    } else {
        $("#work-groups-list").html("No groups created");
    }

    if ( notesGroups.length > 0 ) {
        var html = "";
        for ( var i = 0; i < notesGroups.length; ++i ) {
              if ( notesGroups[i].isMember(user.id) )
                    html += Skt.UsersSidebar.buildGroupHtml(notesGroups[i]);
        }
```

```
        $("#notes-groups-list").html(html);
    } else {
        $("#notes-groups-list").html("No groups created");
    }
};
```

Figura 36. Código de lógica a ejecutar cuando se actualiza el estado de la lista de grupos creados en la clase.

En la figura 36 se ve el código que actualiza la lista de grupos de la barra lateral de *Users* cuando ocurre un cambio en la lista de grupos. La función *handleGroupsUpdated* es el callback que es llamado cuando ocurre un evento de creación de grupos y esta es la encargada de llamar a las funciones necesarias para actualizar los elementos HTML sobre grupos de la vista, lo que en este caso corresponde a llamar a la función *updateGroups*. Esta última función construye el código HTML que representa a los grupos en la vista y se lo asigna al contenedor respectivo.

El último paso que no se alcanza ver en las figuras con el código es hacer el bind entre los eventos que el objeto lógico disparará cuando ocurran cambios y la respectiva función. Esta asignación se realiza en el archivo *logic.js* con la siguiente línea:

```
sketchpad.on("groupsUpdated", Skt.UsersSidebar.handleGroupsUpdated,
Skt.UsersSidebar);
```

5. Implementación “Analytics Dashboard”

La implementación de las visualizaciones del Analytics Dashboard es un poco más simple que la de la Drawing App pero cada una tiene algo de lógica Javascript dependiendo de la visualización.

Al igual que en las implementaciones de las vistas explicadas anteriormente, se utiliza el dinamismo de las páginas JSP y de JSTL para procesar dinámicamente el contenido obtenido desde el controlador de la vista y sobre el resultado de este proceso se dibuja la visualización usando Javascript.

A continuación se verá la implementación de cada visualización del Dashboard enfocando principalmente la lógica de visualización que son particulares de dicha visualización.

5.1. Implementación “Activity Graph”

Para la implementación de los selectores de asignatura, usuario y desagregación se utilizó JSTL y Javascript mientras que para la implementación de los gráficos en sí se usó la librería Google Charts.

```
<select id="selected-subject">
  <c:forEach var="subject" items="${subjectVersions}">
    <option value="${subject.id}">${subject.name}</option>
  </c:forEach>
</select>
<select id="selected-user">
  <c:forEach var="user" items="${users}">
    <option value="${user.id}">${user.name} ${user.lastname}</option>
  </c:forEach>
</select>
<br/>
<span style="margin-left: 5px;">
  <input id="deaggregate-check" type="checkbox" style="position:
relative; top: -4px;"/> Deaggregate by action type
</span>
```

Figura 37. Código de selectores de asignatura, usuario y desagregación.

Cómo se ve en la figura 37, para recorrer la lista de asignaturas obtenida del controlador se utiliza el comando <c:forEach> de JSTL. Veremos esta misma implementación en los selectores encontrados en otras visualizaciones.

Para producir el cambio en la visualización al usar uno de los selectores se ocupó Javascript de manera que al seleccionar una opción se redirecciona el navegador a la visualización con los parámetros seleccionados cómo parámetros de ruta en la URL. La notación *${var}* de JSTL es usada para acceder a los atributos entregados por el controlador.

```
$('#selected-user')[0].selectedIndex = "<%= userIndex %>";
$('#selected-user').change(function(){
  window.location =
'${basePath}/dashboard/activityGraph?subjectVersionId=${subjectVersio
nId}&userId=' + $(this).val();
});
```

Figura 38. Implementación de uso de selectores por el usuario.

Finalmente para instanciar y dibujar los gráficos usamos una combinación de JSTL y Google Charts. Con JSTL leemos los datos del gráfico entregados por el controlador y construimos un arreglo de datos en el formato requerido por Google Charts para dibujar. Luego entregamos este arreglo a la función de Google Charts que dibuja el gráfico en conjunto con las opciones del tipo de gráfico que queremos dibujar y sus colores.

```
function drawPersonalChart() {
  drawAggregatedChart(
    'chart-personal',
    _buildAggregatedDataTable([
      <c:forEach var="entry" items="${aggPersonalSlides.graph}">
        [
          '${entry.activity}',
          ${entry.personalNumber != null ? entry.personalNumber :
"null"},
          ${entry.classroomNumber != null ? entry.classroomNumber :
"null"},
              (${aggPersonalSlides.actionsMax} -
${aggPersonalSlides.actionsMin})*(${entry.evaluationDetail.personalSc
ore}/7.0) + ${aggPersonalSlides.actionsMin},
              '${entry.evaluationDetail.personalScore}',
              '${targetUser.name} ${targetUser.lastname}
${entry.evaluationDetail.evaluationName} score
(${entry.evaluationDetail.evaluationDate}):
${entry.evaluationDetail.personalScore}',
              (${aggPersonalSlides.actionsMax} -
${aggPersonalSlides.actionsMin})*(${entry.evaluationDetail.classroomS
core}/7.0) + ${aggPersonalSlides.actionsMin},
              '${entry.evaluationDetail.classroomScore}',
              'Classroom mean
${entry.evaluationDetail.evaluationName} score
(${entry.evaluationDetail.evaluationDate}):
${entry.evaluationDetail.classroomScore}'
        ],
      </c:forEach>
    ]),
    'Personal Slides',
    buildColors(['#00f', '#bbf'])
  );
}
```

Figura 39. Código de la generación del gráfico *Personal Slides* del Activity Graph.

En el código de la figura 39 se ve que se utilizan algunas funciones Javascript misceláneas al momento de crear el gráfico. Estas funciones cumplen un papel menor de formatear ciertas opciones del gráfico para el uso de Google Charts.

El fragmento importante del código es lo que se encuentra entre los bloques JSTL <c:forEach>; es aquí donde leyendo las filas de entradas entregadas por el controlador se construye el arreglo final que será dibujado en el gráfico.

Se genera un arreglo y se hace una llamada a crear un gráfico por cada gráfico en la visualización. Esto implica la creación de 9 arreglos distintos de datos para generar cada gráfico en esta vista, 4 por los gráficos agregados y 5 por los gráficos desagregados.

5.2. Implementación “Class Activity Graph”

Esta visualización se genera de manera similar a la del Activity Graph pero es en su totalidad más sencilla de generar ya que se ocupa solamente un gráfico en toda la vista.

La estrategia de utilización del selector de asignatura y de los botones de ordenamiento es la misma que la usada en los selectores del Activity Graph, usando funciones callback de Javascript al momento en el que se seleccionan opciones distintas que redireccionan al usuario a la misma visualización pero con nuevos parámetros específicos de la vista.

Como vimos en el diseño de los servicios con los que se comunica el controlador de este gráfico, el ordenamiento de las entradas de él se realiza en la capa de servicios. Por ende al presionar el botón de ordenamiento de en este gráfico simplemente se agrega el criterio de ordenamiento como parámetro en la URL, el cuál es entregado del controlador al servicio respectivo y se obtienen las columnas en el orden deseado.

Consecuentemente, la construcción del gráfico en la vista consiste en generar el arreglo requerido por Google Charts para dibujar el gráfico, el cuál nuevamente se obtiene usando una iteración <c:forEach>, como se puede ver en la figura 40.

```
var data = new google.visualization.DataTable();

<c:forEach var="entry" items="${entries}">
  data.addRow([
    '${entry.user}',
    Math.round(${entry.personalSlides}),
    Math.round(${entry.groupWork}),
    Math.round(${entry.teacherSlides}),
    ${max}*${entry.evaluations}/7.0,
    '${entry.user}: ' + ${entry.evaluations},
    '' + ${entry.evaluations}
  ]);
</c:forEach>
```

Figura 40. Código de creación de arreglo de datos para usar en gráfico de Google Charts.

5.3. Implementación “Learning Network”

La implementación de la Learning Network difiere con las demás en que se ocupa la librería halfviz.js para dibujar el grafo [30].

La estrategia de construcción del grafo consiste mayoritariamente del proceso que ocupan las demás visualizaciones, teniendo un selector de asignaturas y filtros que son manejados por Javascript al igual que el Activity Graph y el Class Activity Graph, en donde la selección de una nueva opción de filtro envía al usuario a la misma vista pero con distintos parámetros, los cuales son interpretados por el controlador para notificar al servicio de Learning Network que se utilicen los diferentes valores escogidos en la construcción del grafo.

El grafo por su parte es dibujado usando la librería de grafos halfviz.js, la cual se encarga de hacer el dibujo en un canvas y de las interacciones y animaciones de los elementos en él. Para construir el grafo que se desea dibujar, halfviz parsea texto en un cierto formato que indica los nodos y enlaces que debe dibujar. Este texto es construido como un String de Javascript. En la figura 41 se puede ver un ejemplo de texto en el formato de halfviz para construir un grafo. La definición de cómo funciona el parseador de texto de halfviz se puede ver en [30].

```
1 -- 2
1 -- 3
1 -- 4
1 -- 5
1 -- 6

2 -- 7
2 -- 8
2 -- 9

3 -- 10
3 -- 11
```

Figura 41. Ejemplo de texto en el formato que parsea halfviz para construir un grafo.

Luego la estrategia para construir el grafo consiste en iterar sobre los datos obtenidos del controlador e irlos concatenando a un string Javascript contenedor del texto a parsear por halfviz y finalmente asignar este texto al parseador de halfviz para ver el resultado.

5.4. Implementación “Attendance”

En el caso de esta visualización no se utiliza Google Charts para dibujar la tabla. En cambio la tabla se genera con HTML simple iterando sobre las filas de la tabla obtenida desde el controlador.

Al igual que en las visualizaciones anteriores, la implementación de los selectores y de los botones de ordenamiento se logra usando Javascript y los parámetros son agregados a la URL. La lógica de ordenamiento de las filas es realizada en el servicio encargado de recuperar los datos de la visualización por lo que a este nivel solamente se procesa cada fila obtenida en el orden recibido y se agregan las celdas una a una al código HTML resultante de la página.

```
<c:forEach var="row" items="${tableContent}">
  <tr>
    <c:forEach var="cell" items="${row}">
      <c:choose>
        <c:when test="${'user' eq cell.type}">
          <th>${cell.value}</th>
        </c:when>
        <c:when test="${'evaluation' eq cell.type}">
          <td class="miss evaluation">
            <c:if test="${cell.present}">${cell.value}</c:if>
          </td>
        </c:when>
        <c:when test="${'activity' eq cell.type}">
          <c:if test="${cell.present}">
            <c:set var="val" value="${100*(maxActivityVal-
cell.value)/(maxActivityVal-minActivityVal)}"></c:set>
            <td class="hit" style="background-color:
rgb(255,<fmt:formatNumber type="number" pattern="##0" value="${val +
150}"/>,<fmt:formatNumber type="number" pattern="##0"
value="${val}"/>);"></td>
          </c:if>
          <c:if test="${not cell.present}">
            <td class="miss"></td>
          </c:if>
        </c:when>
      </c:choose>
    </c:forEach>
  </tr>
</c:forEach>
```

Figura 42. Generación de celdas de la visualización Attendance.

5.5. Implementación “Actions v/s Evaluations”

Este es una de las visualizaciones más simples de dibujar en términos de código en este nivel ya que nuevamente la mayor parte del trabajo de filtrado de los selectores ocurre en los servicios y controladores y la visualización consiste de un único gráfico de burbujas obtenido usando Google Charts.

La estrategia de implementación de uso de los selectores de filtrado es de la misma forma lograda usando Javascript y redireccionamiento.

A continuación se muestra el código usado para generar la tabla de datos usada por Google Charts para dibujar este gráfico de burbujas.

```
var data = new google.visualization.DataTable();

data.addColumn('string', 'Student');
data.addColumn('number', 'Evaluation score');
data.addColumn('number', 'Number of Actions');
data.addColumn('string', 'Name');
data.addColumn('number', 'Attendances');

<c:forEach var="entry" items="${entries}">
  data.addRow([
    '${entry.student}',
    ${entry.evaluations},
    ${entry.actions},
    '${entry.student}',
    ${entry.attendance}
  ]);
</c:forEach>
```

Figura 43. Generación de tabla de datos usada para dibujar gráfico de Actions v/s Evaluations.

VIII. Pruebas de Usuario

Se realizó una prueba de usuario en una clase auxiliar del curso de Algoritmos y Estructuras de Datos con la intención de validar la interfaz de usuario de la aplicación, presentando a los alumnos problemas contextuales a la materia que veían en el curso a la fecha: Balanceo de Árboles Binarios AVL e inserción y eliminación en Árboles 2-3.

A la clase auxiliar asistieron alrededor de 20 alumnos y se realizó el día 16 de Mayo de 2014 en el Auditorio DCC, sala que cuenta con un proyector y un computador que permiten desplegar contenido en una pizarra.

En la primera hora de clase los auxiliares realizaron clases normales y se usaron los últimos 30 minutos de la clase para hacer la prueba de usuario. Se les explicó a los alumnos en qué consistía la aplicación, el contexto en el que sería usada y una breve explicación de la motivación del proyecto. También se les explicó en qué consistiría la prueba de usuario y qué se esperaba obtener de feedback. Los alumnos se mostraron interesados y con buena disposición de colaborar con la actividad.

Antes de comenzar a trabajar los problemas, se le dio a los alumnos instrucciones generales de cómo utilizar la aplicación, explicando la interfaz de usuario y las herramientas disponibles para dibujar entre otras cosas. Luego se explicó la metodología en la que se haría la actividad, la cuál consistió en, usando el proyector, mostrar un problema en la pizarra a través de la aplicación, explicar en qué consistía el problema y luego pedir que un voluntario lo resolviera usando un iPad que fue dispuesto para la actividad. Los problemas presentados fueron problemas que debían ser resueltos gráficamente, de modo que la idea era que los alumnos dibujarán los pasos necesarios para resolver el problema y dibujaran el estado final del árbol en la misma diapositiva donde el problema se presentaba.

La actividad resultó ser exitosa. Los alumnos entendieron rápidamente cómo usar la aplicación y no tardaron en dibujar las soluciones de los problemas. En la figura 44 se puede ver uno de los problemas resueltos usando la aplicación. El resto de los problemas se pueden ver en el Anexo 1.

Al empezar a resolver los ejercicios, algunos de los voluntarios que no tenían claro cómo resolver el problema gatillaron espontáneamente una discusión entre todos los alumnos sobre cómo había que resolver el problema y entre todo el grupo desarrollaron una solución al tiempo que el voluntario con el iPad la dibujaba. Luego al final de cada problema los auxiliares explicaron la solución correcta, que a veces fue la misma a la que habían llegado los alumnos, y esto ayudó a que todos quedaran con una idea clara de cómo resolver correctamente el problema.

Árboles 2 3

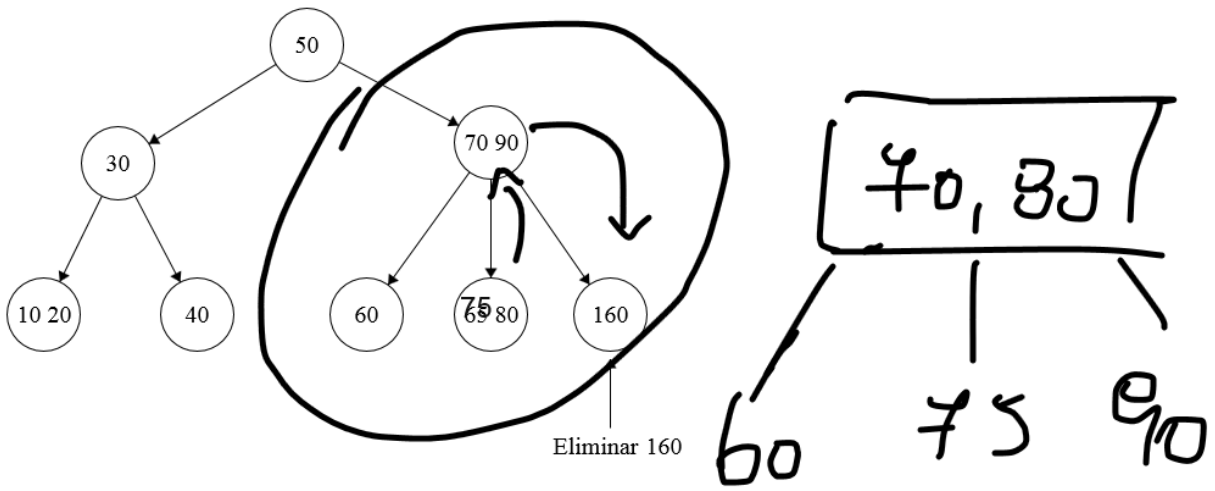


Figura 44. Problema de eliminación en árboles 2-3.

Al final de la clase algunos alumnos se acercaron y dejaron una hoja con comentarios de feedback, estos se pueden ver en el Anexo 2. En general a los alumnos le gustó la actividad y la encontraron útil para aprender la materia.

En definitiva, incluso usando solo un iPad en la clase, la actividad funcionó eficientemente, se logró la enseñanza que se esperaba y descubrimos que el uso de la aplicación fomentó la espontánea discusión entre los alumnos. Además se obtuvo feedback útil respecto la aplicación y respecto a la interfaz de usuario que no había sido considerada previamente.

En el momento en el que se realizó esta prueba no se encontraba terminada la funcionalidad de Learning Analytics y por ende no se pudo tener feedback respecto a esta dimensión de la aplicación.

IX. Conclusiones y Trabajo futuro

Como aplicación colaborativa de apoyo a la educación presencial, Sketchpad logra promover la interactividad entre usuarios y se considera que en efecto permite el desarrollo de clases colaborativas. En este sentido el desarrollo de la aplicación se completó exitosamente de acuerdo al propósito para el que fue diseñada.

Por otra parte, las herramientas de Learning Analytics que se idearon para Sketchpad también fueron implementadas exitosamente en la forma en la que fueron diseñadas.

Como trabajo futuro se espera estudiar el resultado que se obtenga al transcurrir un semestre de clases en el que estudiantes hagan uso de la aplicación frecuentemente en sesiones de clases regulares en las que haya espacio para el trabajo colaborativo. La sola utilización de la aplicación de esta forma proveerá valiosos datos recopilados por el sistema de Analytics de Sketchpad que permitirá analizar si las visualizaciones desarrolladas para el Analytics Dashboard ayudan a educadores y estudiantes a encontrar tendencias o relaciones entre las actividades hechas en clases y el desempeño académico.

Dado que no se realizaron pruebas de usuarios luego de que las funcionalidades del Analytics Dashboard fueron completadas, no se pudo validar si los datos recopilados en el proceso de Analytics son efectivamente datos útiles para los usuarios como métricas de desempeño en clases o como correlativos de desempeño académico. De la misma forma no se pudo validar que las visualizaciones desarrolladas son las más eficientes para encontrar tendencias entre estudiantes o para correlacionar resultados.

Sin embargo, cualquiera sea el resultado de hacer dicha validación será valioso para el desarrollo de estudios respecto a Learning Analytics en aplicaciones de trabajo colaborativo ya que, con una significativa cantidad de datos, se podrá confirmar o descartar el uso de los gráficos y métricas implementadas en Sketchpad para este tipo de aplicaciones.

La arquitectura y el diseño de la aplicación resultaron cumplir exitosamente su propósito de desarrollar un software cuyos componentes tienen bajo acoplamiento y ordenadas dependencias. Por su parte la metodología de desarrollo usada permitió un ágil desarrollo de nuevas componentes y funcionalidad y una rápida evaluación y aprobación de estas por parte del cliente. En este sentido, la arquitectura ocupada permitió que las iteraciones fuesen óptimamente desarrolladas y que cada nueva funcionalidad implementada fuese creada de forma atómica, es decir, con poco

acoplamiento a otras funcionalidades y generalmente consistiendo de un componente propio independiente del resto de componentes. De esta manera y consecuentemente la implementación de la aplicación final se completó exitosamente al ocupar estas buenas prácticas de desarrollo de Software.

De esta manera, se considera que si bien faltó hacer pruebas de usuario para validar ciertos aspectos de la aplicación, el uso de la aplicación entregará información de Analytics valiosa para estudiantes y educadores al mismo tiempo de ser una viable herramienta para apoyar la educación presencial y potenciar el trabajo colaborativo.

X. Bibliografía

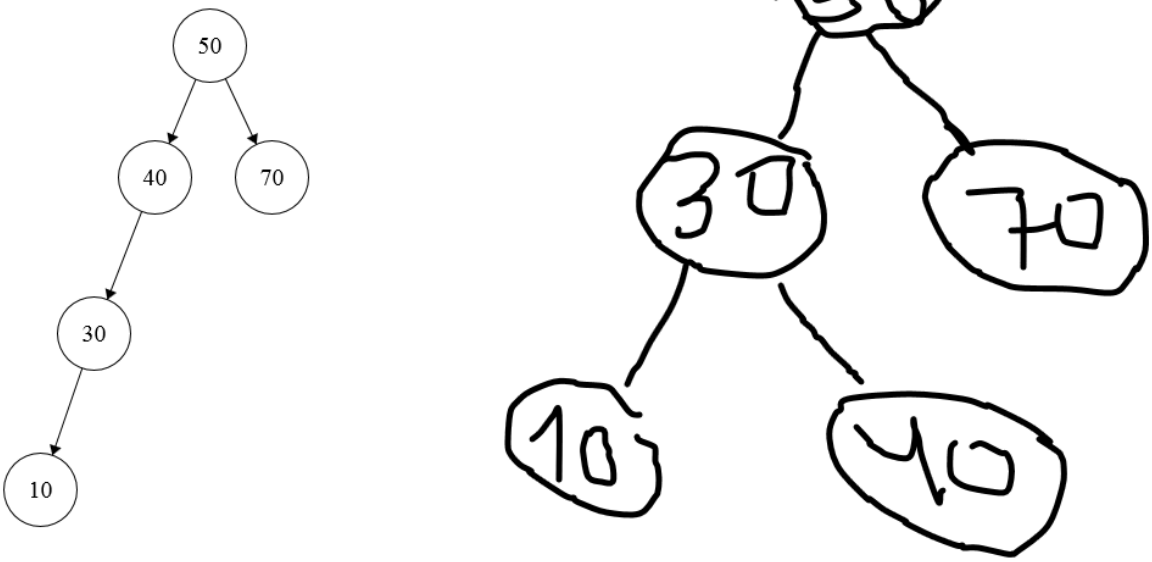
1. M. Ratto, R. Shapiro, T. Truong, W. Griswold, “The activeclass project: Experiments in encouraging classroom participation”. En CSCL 2003.
2. S. Kopf, N. Scheele, W. Effelsberg, “The Interactive Lecture: Teaching and Learning Technologies for Large Classrooms”, Department for Mathematics and Computer Science, University of Mannheim, Mannheim, GDR 2005.
3. Siemens, G., & Long, P. (2011). “Penetrating the fog: Analytics in learning and education.” *Educause Review*, *46*(5), pp. 30-32.
4. 1st International Conference on Learning Analytics and Knowledge, Banff, Alberta, Febrero-Marzo 2011, https://tekri.athabascau.ca/analytics/.
5. J. Heines, W. Liang, “Combining, Storing, and Sharing Digital Ink,” Proceedings of SIGSE’07, Covington, Kentucky, EEUU. Marzo 7-10, 2007.
6. D. Nikola, “MCPresenter: Herramienta Colaborativa Móvil de Apoyo a Diversas Prácticas Pedagógicas y Gestión del Conocimiento”, memoria de título, Departamento de Ciencias de la Computación, Universidad de Chile 2009.
7. R. Dufresne, W. Gerace, W. Leonard, P. Mestre, L. Wenk, “Classtalk: A classroom communication system for active learning”, Journal of Computing in Higher Education, 7, 1996, pp. 3-47.
8. Brynjolfsson, E., Hitt, L., & Kim, H. (2011). “Strength in numbers: how does data-driven decisionmaking affect firm performance?”. *Available at SSRN 1819486*.
9. Anderson, C. (2008). “The end of theory”. *Wired magazine*, *16*.
10. Ali, L., Hatala, M., Gašević, D., & Jovanović, J. (2012). A qualitative evaluation of evolution of a learning analytics tool. *Computers & Education*, *58*(1), pp. 470-489.
11. Ferguson, R. (2012). Learning analytics: drivers, developments and challenges. *International Journal of Technology Enhanced Learning*, *4*(5), pp. 304-317.
12. RJ. Anderson, R. Anderson, T. Vandegrift, S. Wolfman, K. Yasuhara, “Promoting interaction in large classes with computer-mediated feedback”, CSCL 2003.
13. Mazza, R., & Dimitrova, V. (2007). “CourseVis: A graphical student monitoring tool for supporting instructors in web-based distance courses. *International Journal of Human-Computer Studies”*, *65*(2), pp. 125-139.
14. Mazza, R., & Milani, C. (2005). “Exploring usage analysis in learning systems: Gaining insights from visualisations”. *AIED’05 workshop on Usage analysis in learning systems* (pp. 65-72).
15. Kosba, E., Dimitrova, V., & Boyle, R. (2005). “Using student and group models to support teachers in web-based distance education”. In *User Modeling 2005* (pp. 124-133). Springer Berlin Heidelberg.

16. Zinn, C., & Scheuer, O. (2006). “Getting to know your student in distance learning contexts. In *Innovative Approaches for Learning and Knowledge Sharing”* (pp. 437-451). Springer Berlin Heidelberg.
17. Scheuer, O., & Zinn, C. (2007). “How did the e-learning session go? The Student Inspector”. *Frontiers in Artificial Intelligence and Applications*, *158*, 487.
18. Golub, E. (2004). “Supporting faculty goals during student presentations via electronic note-taking”. *Frontiers in Education, 2004. FIE 2004. 34th Annual* (pp. F4E-13). IEEE.
19. G. Zurita, N. Baloian, F. Baytelman, “Supporting rich interaction in the classroom with mobile devices", Fifth IEEE International Conference on Wireless, Mobile, and Ubiquitous Technology in Education (WMUTE), pp. 115-122, 23-26, 2008
20. A. Schmidt, M. Lau, M. Beigl, \Handheld C.S.C.W.," Workshop on handheld, Proceedings of the Computer Supported Collaborative Work, CSCW'98 Noviembre, Seattle, 1998.
21. C. Liu, L. Kao, “Handheld Devices with Large Shared Display Groupware: Tools to Facilitate Group Communication in One-to-One Collaborative Learning Activities," IEEE International Workshop on Wireless and Mobile Technologies in Education (WMTE'05), pp. 128-135, 2005.
22. Siemens, G., Gasevic, D., Haythornthwaite, C., Dawson, S., Shum, S. B., Ferguson, R., ... & Baker, R. S. J. D. (2011). Open Learning Analytics: an integrated & modularized platform. *Proposal to design, implement and evaluate an open platform to integrate heterogeneous learning analytics techniques*.
23. Wise, A. F., Zhao, Y., & Hausknecht, S. N. (2013, April). Learning analytics for online discussions: a pedagogical model for intervention with embedded and extracted analytics. In *Proceedings of the Third International Conference on Learning Analytics and Knowledge* (pp. 48-56). ACM.
24. Phillips, R., Maor, D., Preston, G., & Cumming-Potvin, W. (2012). Exploring learning analytics as indicators of study behaviour.
25. Boulanger, D., Seanosky, J., Kumar, V., Kinshuk, Panneerselvam, K., Selvi, T., “Smart Learning Analytics”, Athabasca University, Athabasca, Canada.
26. Seanosky, J., Boulanger, D., Kumar, V., Kinshuk, (2014). “Unfolding learning analytics for big data”, International conference on smart learning environment (ICSLE 2014), Hong Kong, China.
27. Johnson, P. M., Kou, H., Agustin, J. M., Zhang, Q., Kagawa, A., & Yamashita, T. (2004, August). Practical automated process and product metric collection and analysis in a classroom setting: Lessons learned from Hackystat-UH. In *Empirical Software Engineering, 2004. ISESE'04. Proceedings. 2004 International Symposium on* (pp. 136-144). IEEE.

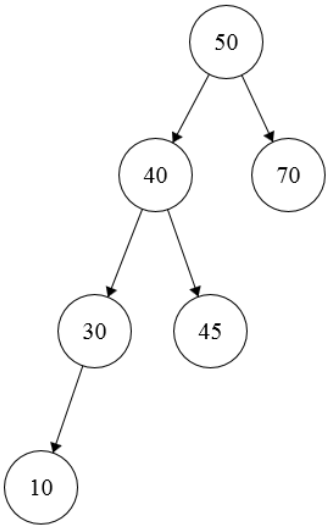
28. Johnson, P. M. (2007, September). Requirement and Design Trade-offs in Hackystat: An In-Process Software Engineering Measurement and Analysis System. In *ESEM* (Vol. 7, pp. 81-90).
29. MultipartFile (Spring Framework 4.1.5 RELEASE API). http://docs.spring.io/spring/docs/current/javadoc-api/org/springframework/web/multipart/MultipartFile.html
30. arbor.js. http://arborjs.org/halfviz/
31. Chapter 14. HQL: The Hibernate Query Language. https://docs.jboss.org/hibernate/orm/3.3/reference/en/html/queryhql.html
32. IDE Eclipse. http://eclipse.org
33. Java Empresarial (JEE). http://www.oracle.com/technetwork/java/javaee/overview/index.html
34. Spring Web MVC Framework. http://docs.spring.io/spring/docs/current/spring-framework-reference/html/mvc.html

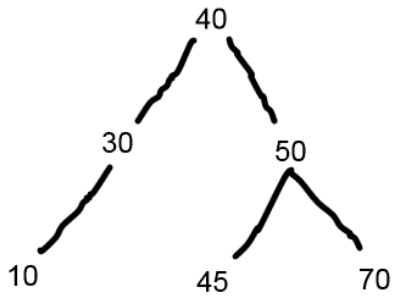
XI. Anexos

1. Anexo 1: Problemas resueltos en Auxiliar de Algoritmos y Estructuras de Datos



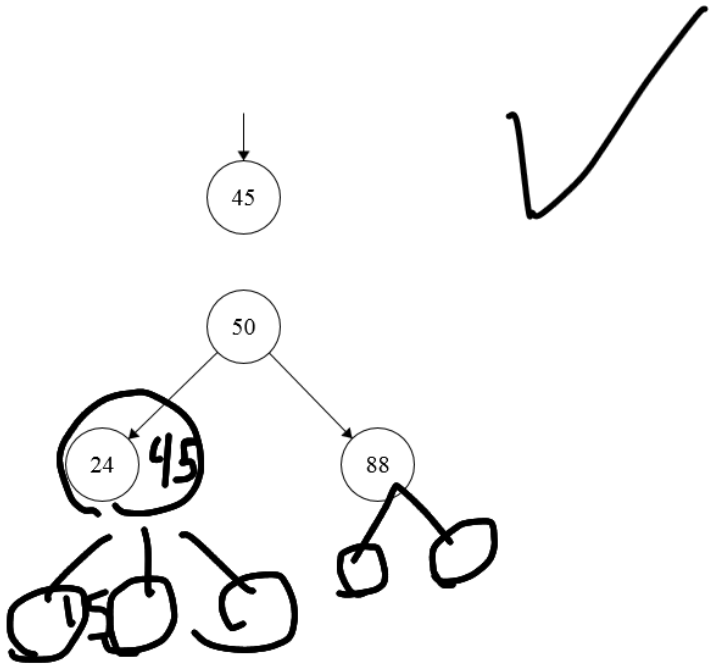
Problema 1



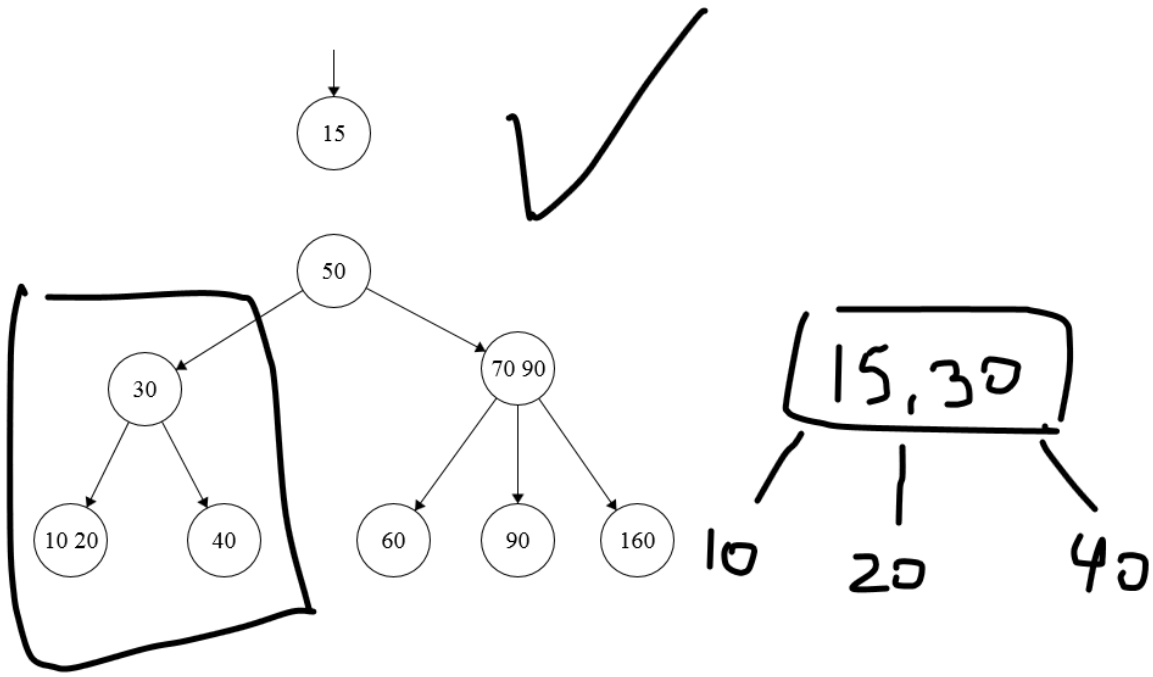


Problema 2

Árboles 2 3

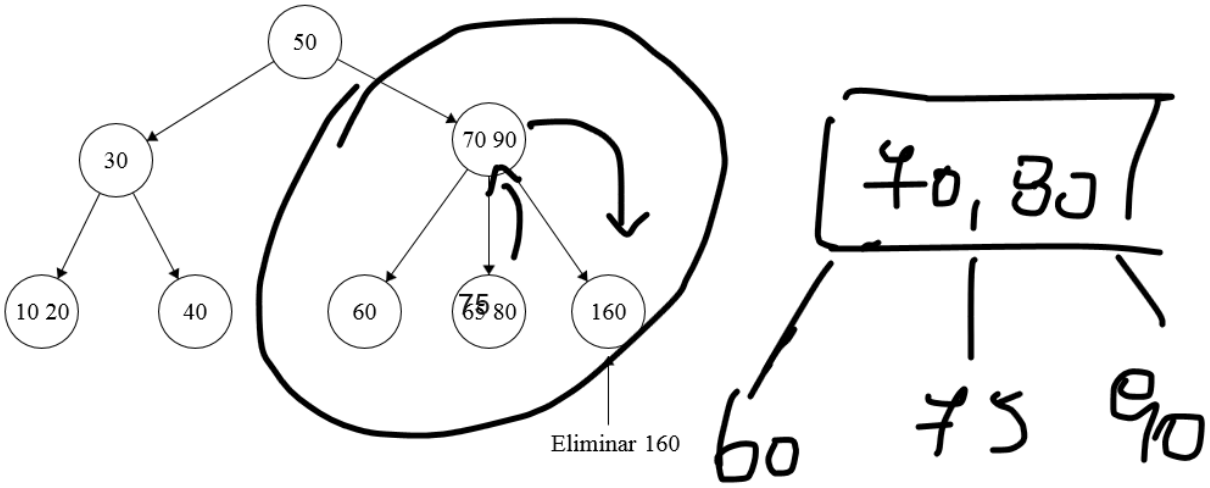


Problema 3.

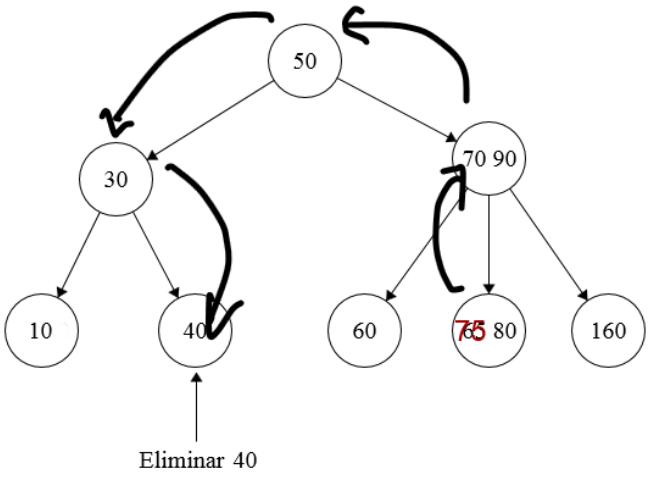


Problema 4.

Árboles 2 3



Problema 5.



Problema 6.

2. Anexo 2: Feedback alumnos curso Algoritmos y Estructuras de Datos

2.1. Alumno 1

- Mostrar el nombre del usuario en la barra superior, para evitar confusiones
- Agregar un botón para ir al slide siguiente
- Posibilidad de eliminar los cambios hechos por un alumno o todos
- Ocultar un slide, para poder plantear un problema y tener la solución en la slide siguiente sin que los alumnos la vean
- Botón para eliminar los permisos de todos los alumnos de dibujar en caso de que ocurra una confusión

2.2. Alumno 2

- Implementar figuras predefinidas
- Podría abrir pdf, ppt, formatos de imágenes y editar sobre ellos

2.3. Alumno 3

- Agregar figuras predefinidas