Universidad Autónoma Metropolitana Unidad Azcapotzalco

División de Ciencias Básicas e Ingeniería

Licenciatura en Ingeniería en Computación

Gestión de calificaciones de cursos mediante servicios Web

Diseño del sistema

Elaboró:
Avendaño Méndez Sergio Enrique 205304956

Trimestre: 12-O

Fecha:
10 de Diciembre de 2012

Asesores:
Dra. Beatriz Adriana González Beltrán

Dra. Lizbeth Gallardo López

Índice

1. Introducción

El objetivo del proyecto es desarrollar un sistema de *gestión de calificaciones de cursos mediante servicios Web,* un profesor podrá realizar las funciones necesarias para controlar sus grupos, a saber: definir los instrumentos de evaluación (exámenes, proyectos, tareas, etc); y de acuerdo a los criterios definidos por él mismo (escala de calificación y reactivos) obtener las calificaciones de los alumnos. La aplicación deberá ser flexible ante posibles cambios en los instrumentos de evaluación y ante cambios en los alumnos inscritos. Para construir la aplicación, se desarrollarán servicios Web que realicen todas las funciones que le permitan gestionar las calificaciones de los cursos. Dichos servicios podrán ser invocados desde una aplicación cliente (para PC).

En este documento se presentan los artefactos de diseño que se utilizaron para modelar el comportamiento del Sistema de Evaluación de Alumnos (SEA), dichos artefactos son:

- Requerimientos del sistema
- Diagrama de casos de uso general
- Escenarios de uso
- Diagrama de clases
- Análisis de robustez
- Diagrama de la arquitectura del sistema
- Diagrama de la base de datos

2. Especificación del sistema

2.1 Requerimientos

A continuación, se listan los requerimientos funcionales del sistema, así como su prioridad y descripción.

Requisitos	Prioridad	Descripción
Gestionar modalidad de evaluación de los cursos	Alta	La aplicación debe proporcionar al profesor las funcionalidades necesarias para definir y modificar una modalidad de evaluación. Esta información deberá guardarse en una base de datos para su posterior uso. Una modalidad de evaluación podría componerse de la siguiente manera: los instrumentos de evaluación pueden ser tareas, exámenes y programas. Como criterios tendríamos que las tareas tuvieran un peso del 20% de la calificación final, los exámenes un peso de 50% y los programas un peso de 30%.
Gestionar la evaluación de los alumnos	Alta	La aplicación debe proporcionar al profesor las funciones necesarias para el registro de calificaciones de los alumnos asociados a un curso, esto de acuerdo a los instrumentos y criterios de evaluación definidos en la modalidad de evaluación. También, deberá proporcionar la función para calcular la calificación final. Las calificaciones deberán guardarse en una base de datos para su consulta.
Gestión de grupos	Alta	La aplicación deberá proporcionar al profesor las funciones necesarias para crear y modificar los grupos a los que imparte clases. Cada grupo tiene las siguientes propiedades: lista de alumnos, clave de grupo, trimestre, horario y tipo de evaluación. Estas propiedades deberán guardarse en la base de datos. El profesor podrá dar de alta y dar de baja alumnos según lo requiera.
Gestión de materias	Alta	La aplicación de software deberá proporcionar al profesor las funciones necesarias para el registro de las materias que el profesor va a impartir. Para cada materia deberán registrarse los datos siguientes: clave, nombre y número de créditos, los cuáles, serán guardados en la base de datos. Una materia deberá estar asociada a un grupo de alumnos.

2.2 Roles de personas involucradas en la evaluación de alumnos.

- **Administrador:** Este actor es el encargado de manipular completamente el sistema.
  1) Gestionar modalidad de evaluación de los cursos
  2) Gestionar la evaluación de los alumnos
  3) Gestionar grupos
  4) Gestionar materias

2.3 Requerimientos no funcionales

- Que la aplicación emplee la menor cantidad de recursos (memoria y tiempo de procesador) posible.
- Que las funciones que proporcione la aplicación sean lo más entendibles y fáciles de usar.

3. Diagrama de casos de uso general

Del documento de requerimientos se pudo determinar al actor principal, el profesor; y a los casos de uso, con los cuales, el actor busca producir un resultado de valor agregado. El diagrama de casos de uso para este proyecto quedó de la siguiente manera.

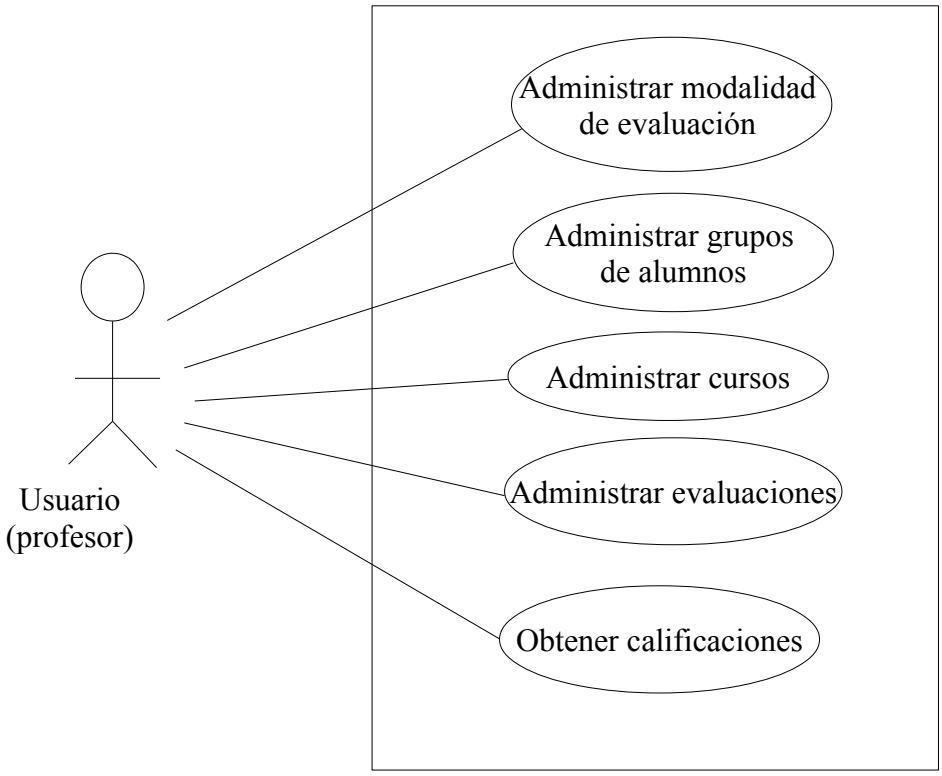


Figura 1. Diagrama de casos de uso

4. Escenarios de uso

A continuación, se listan los escenarios de uso mas importantes del sistema:

4.1 Escenario de uso: Crear modalidad de evaluación

Descripción:
Esta operación permitirá definir los instrumentos de evaluación y sus pesos correspondientes que el profesor vaya a utilizar para evaluar a los alumnos de un grupo a lo largo del curso. Una vez definida la modalidad, ésta deberá asociarse al curso donde vaya a utilizarse.

Actores y sus intereses:
El actor principal (profesor) está interesado en crear una modalidad de evaluación, la cual, utilizará para evaluar a los alumnos de un curso que va a impartir.

Disparador:
La operación inicia cuando el profesor elige la opción "crear modalidad de evaluación".

Pre-condición:
Es necesario haber definido al menos un curso previamente.

Post-condición:
Al terminar el escenario, la modalidad de evaluación estará registrada en la base de datos del sistema y estará asociada al curso que le corresponde.

Flujo principal:
1. El profesor selecciona la opción "crear modalidad de evaluación".
2. El profesor introduce uno a uno el nombre de los instrumentos con sus pesos correspondientes.
3. El sistema verifica que la suma de los pesos de los instrumentos sea igual a 100. En caso negativo se repite el proceso desde el paso 2.
4. El profesor elige el curso que trabajará con esta modalidad de evaluación.
5. El sistema registra en la base de datos la nueva modalidad.
6. El profesor elige si desea crear más modalidades de evaluación para otros cursos. En caso positivo se repite el proceso desde el paso 2.

4.2 Escenario de uso: Alta de grupo de alumnos

Descripción:
Consiste en dar de alta en la base de datos un grupo de alumnos con todos sus datos (clave, cupo, salón, horario). Un grupo puede tener asociados uno o más alumnos.

Actores y sus intereses:
El actor principal (profesor) desea dar de alta un grupo para asociarle alumnos.

Disparador:
La operación inicia cuando el profesor elige la opción “alta de grupo de alumnos”.

Post-condición:
Al terminar la operación, toda la información del grupo quedará registrada en la base de datos para su posterior uso.

Flujo principal:
1. El profesor elige la opción “alta de grupo de alumnos”.
2. El profesor introduce todos los datos del grupo a registrar.
3. El sistema verifica que la información del grupo a registrar no exista en la base de datos.
4. El sistema da de alta la información del grupo.
5. Si el profesor desea dar de alta más grupos, se repite el proceso desde el paso 2.

4.3 Escenario de uso: Alta de alumno

Descripción:
Consiste en dar de alta a un alumno asociado a un grupo en la base de datos. Los datos a almacenar son nombre(s), apellido paterno, apellido materno, matricula y correo electrónico.

Actores y sus intereses:
El actor principal (profesor) desea dar de alta a un alumno en un grupo.

Disparador:
La operación da inicio cuando el profesor elige la opción “Alta alumno”.

Pre-condición:
Es necesario haber definido un grupo previamente para poder dar de alta alumnos.

Post-condición:
Al terminar la operación, la información del alumno quedará registrada en el sistema.

Flujo principal:

1. El profesor elige la opción “Alta alumno”.
2. El profesor ingresa al sistema los datos del alumno (nombre(s), apellido paterno, apellido materno, matrícula y correo electrónico).
3. El sistema guarda en la base de datos la información del alumno.
4. Si el profesor desea dar de alta más alumnos, el proceso se repite desde el paso 2.

4.4 Escenario de uso: Alta de curso

Descripción:
En este escenario de uso, se pretende registrar en la base de datos la información relacionada al curso que el profesor está por impartir. En particular, se tiene que registrar el nombre del curso (nombre de la uea), clave del curso, créditos, trimestre y el horario en el cual se impartirá.

Actores y sus intereses:
El actor principal (profesor) desea dar de alta un curso que está por impartir.

Disparador:
La operación inicia cuando el profesor elige la opción “Alta curso”.

Post-condición:
Al terminar la operación, toda la información del curso será dada de alta en el sistema.

Flujo principal:

1. El profesor elige la opción “Alta curso”.
2. El profesor introduce la información del curso a registrar (nombre del curso, clave del curso, créditos, trimestre, y horario).
3. El sistema registra en la base de datos la información proporcionada por el profesor.
4. Si el profesor desea dar de alta otro curso, se repite el proceso desde el paso 2.

4.5 Escenario de uso: Calificar alumnos

Descripción:
Ésta operación consiste en calificar a los alumnos pertenecientes a un grupo, en base a los instrumentos de evaluación definidos previamente.

Actores y sus intereses:
El actor principal (profesor) desea calificar a sus alumnos en base a un instrumento de evaluación que definió al inicio del curso.

Disparador:
Ésta operación da inicio cuando el profesor elige la opción “calificar alumnos”.

Pre-condición:
Para que se pueda cumplir ésta operación, es necesario que el profesor haya definido previamente al menos un curso, un grupo, alumnos y los instrumentos de evaluación que el profesor utilizará para calificar a sus alumnos.

Post-condición:
Al finalizar éste escenario, las calificaciones emitidas por el profesor estarán registradas en el sistema

Flujo principal:
1. El profesor elige la opción “calificar alumnos”.
2. El sistema muestra la lista de cursos existentes.
3. El profesor elige el curso donde desea evaluar.
4. El sistema muestra la lista de grupos de alumnos correspondientes al curso.
5. El profesor elige el grupo de alumnos donde desea evaluar.
6. El sistema muestra la lista de instrumentos de evaluación definidos para evaluar al grupo.
7. El profesor elige el instrumento con el cual va a evaluar a los alumnos.
8. El sistema muestra la lista de alumnos que pertenecen al grupo.
9. El profesor introduce el nombre de la evaluación (ej, tarea 1, examen 1) y la calificación correspondiente (de 1 a 10) de cada alumno.
10. El sistema registra las calificaciones en la base de datos.
11. Si el profesor desea realizar otra evaluación, el proceso se repite desde el paso 2.

4.6 Escenario de uso: Obtener promedio total

Descripción:
Éste escenario de uso consiste en obtener el promedio total de un alumno, en base a todos los instrumentos de evaluación definidos por el profesor al inicio del curso.

Actores y sus intereses:
El actor principal (profesor) desea conocer los promedios totales correspondientes a los alumnos de algún curso.

Disparador:
El escenario de uso da inicio, cuando el actor elige la opción “Obtener promedio total”.

Pre-condición:
Para que ésta operación se lleve a cabo con éxito, el profesor debe haber realizado evaluaciones

previamente.

Post-condición:
Al finalizar ésta operación, el actor obtendrá el promedio total correspondiente a los alumnos de un grupo.

Flujo principal:
1. El actor elige la opción “Obtener promedio total”.
2. El sistema muestra la lista de cursos existentes.
3. El actor elige el curso, del cual, desea saber el promedio.
4. El sistema muestra la lista de grupos de alumnos correspondientes al curso.
5. El actor elige el grupo de alumnos, del cual, desea saber el promedio.
6. El sistema muestra la lista de alumnos que pertenecen al grupo junto con los promedios totales correspondientes.
7. Si el actor desea obtener más promedios, el proceso se repite desde el paso 2.

5. Diagrama de clases

Para obtener las entidades clave del SEA, se aplicó la técnica de “extracción de sustantivos” sobre el documento de requerimientos y sobre el documento de escenarios de uso. Con esta técnica se pudo obtener una lista de sustantivos candidatos, de la cual fue necesario eliminar aquellas entidades que no resultaron ser clave. La mayoría de las candidatas a entidad se eliminó de la lista, debido a que resultaron ser atributos de otra entidad.

De esta manera, quedaron solo 7 entidades clave:

- Curso
- Escala
- Instrumento
- Evaluación
- Grupo
- Alumno
- Calificación

Una vez elegidas las entidades clave, se definieron sus atributos, responsabilidades y colaboraciones. Para ello se utilizó la técnica de Diseño de clases empleando tarjetas CRC.

5.1 Entidad Curso

La entidad `curso` se refiere a la UEA que va a impartir el profesor. Los atributos de `curso` son: ID, nombre del curso, clave del curso, créditos del curso y trimestre. Se determinó una colaboración entre las entidades `Curso` y `Grupo.`

5.2 Entidad Escala

La entidad `Escala` permite pasar de una nota cuantitativa a una nota cualitativa basada en las letras: NA, S, B, MB. Un profesor podría determinar su escala como se muestra en la figura 2.

Rangos de calificación	Letra
x>=0 y x<6	NA
x>=6 y x<7.5	S
x>=7.5 y x<9	B
x>=9 y x<=10	MB

Tabla 1. Ejemplo de escala determinada por un profesor al inicio de un curso

Cuenta con los atributos ID, Letra, límite_inferior y límite_superior. En los límites inferior y superior se determinará el rango correspondiente a una letra. La entidad `Escala` colaborará con la entidad `Grupo.`

5.3 Entidad Instrumento

La entidad `instrumento` se refiere a los aspectos tales como tareas, exámenes, proyectos, etc. que el profesor determina para evaluar a los estudiantes de un curso en particular. Se definieron únicamente los atributos: ID, nombre del instrumento y peso del instrumento. El peso del instrumento representará el porcentaje de la calificación final que tiene dicho instrumento. Se definieron colaboraciones con la entidad `Grupo` y la entidad `Evaluación`, debido a que un profesor define para un grupo evaluaciones, a partir de uno o más instrumentos de evaluación.

5.4 Entidad Evaluación

La entidad `Evaluación` corresponde a la especificación de un instrumento particular; por ejemplo, para el instrumento de exámenes el profesor puede especificar 3 parciales; para el instrumento de tareas puede especificar 5 tareas diferentes a los largo de curso. Para esta entidad, se definieron los siguientes atributos: ID, nombre de la evaluación y descripción de la evaluación. Se determinó que la entidad `Evaluación` colabora directamente con la entidad `Instrumento` (como ya se explicó anteriormente) y con la entidad `Calificación`.

5.5 Entidad Grupo

La entidad `Grupo` se refiere al conjunto de alumnos inscritos a un cursos particular. Se definieron los atributos: ID, clave de grupo, cupo, salón, alumnos inscritos y horario. Esta entidad va a colaborar con otras tres entidades. Colaborará con la entidad `Curso`, con la entidad `Alumno` y con la entidad `Instrumento`.

5.6 Entidad Alumno

La entidad `Alumno` guarda los datos personales de un alumno inscrito a un grupo particular; y cuenta con los atributos: ID, nombre de alumno, matrícula, email y calificación final. La entidad colaborará con la entidad `Grupo` y con la entidad `Calificación.`

5.7 Entidad Calificación

La entidad `Calificación` asocia a la entidad `Alumno` con la entidad `Evaluación`, definiendo así una calificación de un alumno para una evaluación particular. Sus atributos son ID y calif.

Al final, el diagrama de clases quedó de la siguiente manera:

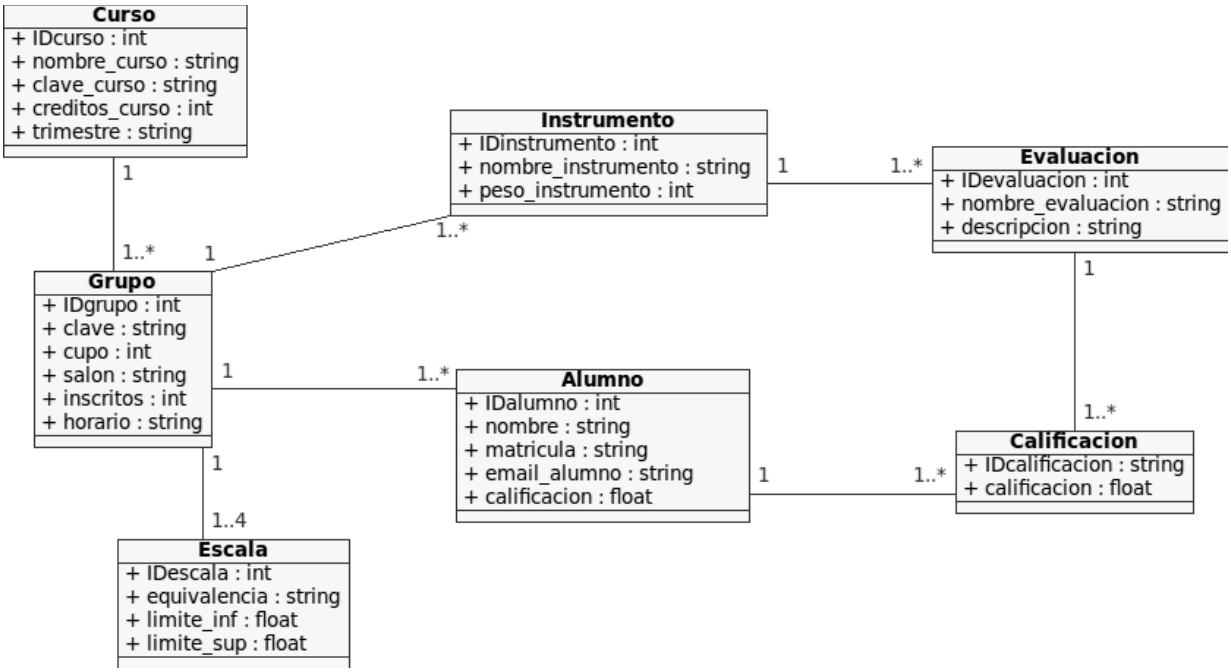
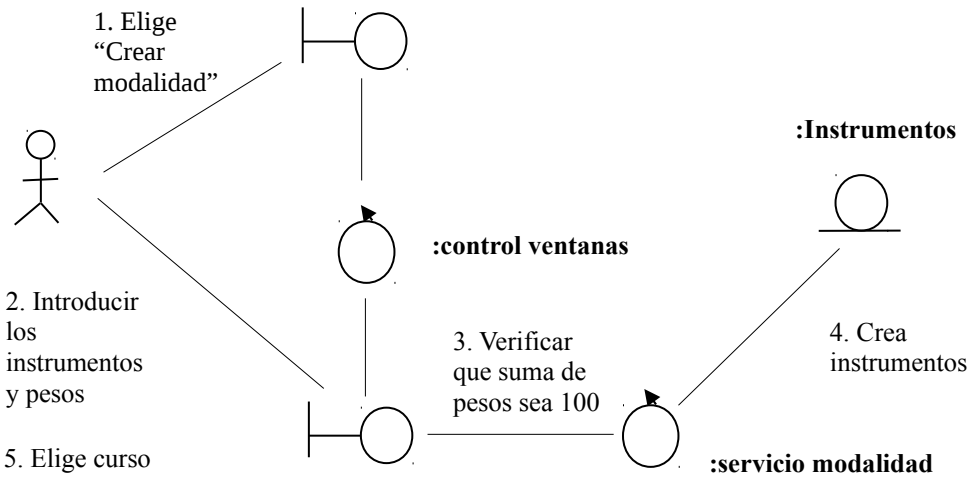


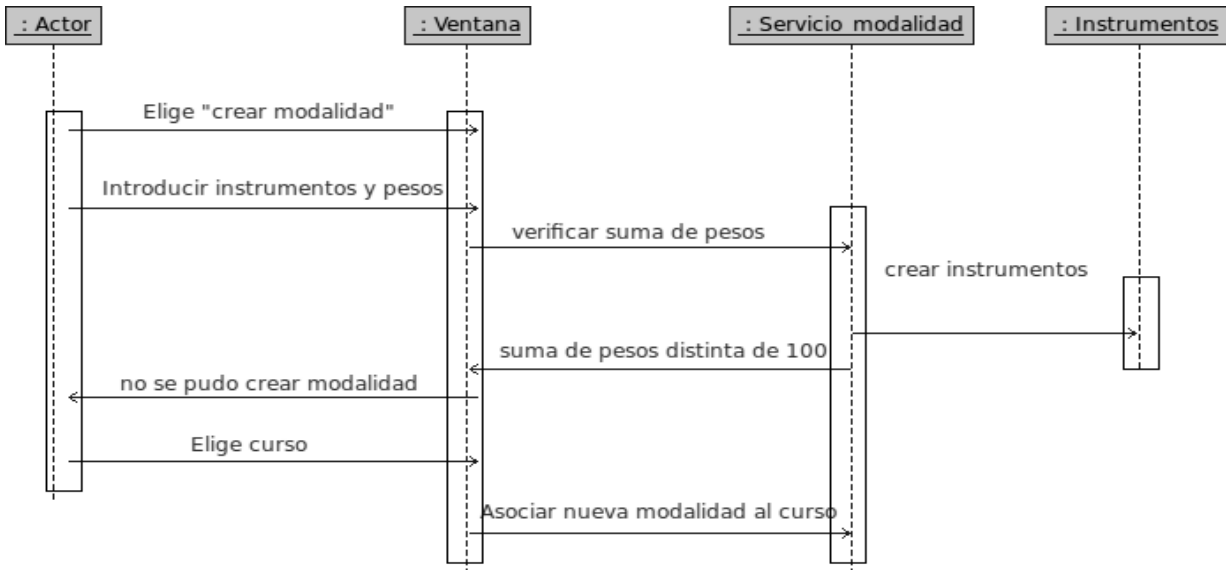
Figura 2. Diagrama de clases

Todas las clases cuentan con métodos set y métodos get, con los cuáles, se pueden definir y recuperar los atributos de cada una de las instancias de la clase. De esta forma estas clases pasan a funcionar como clases Java Bean.

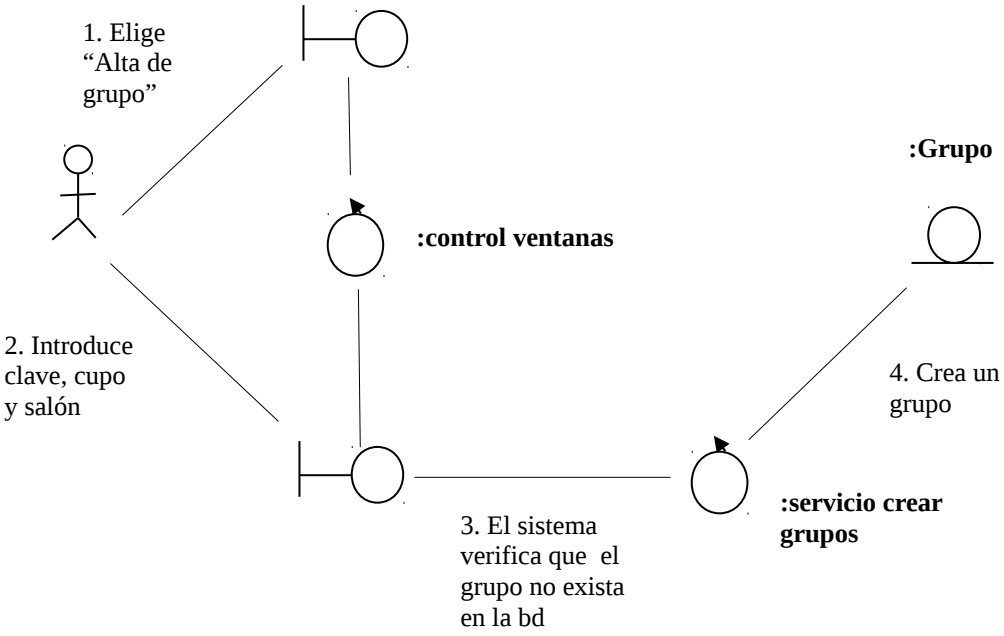
6. Análisis de robustez

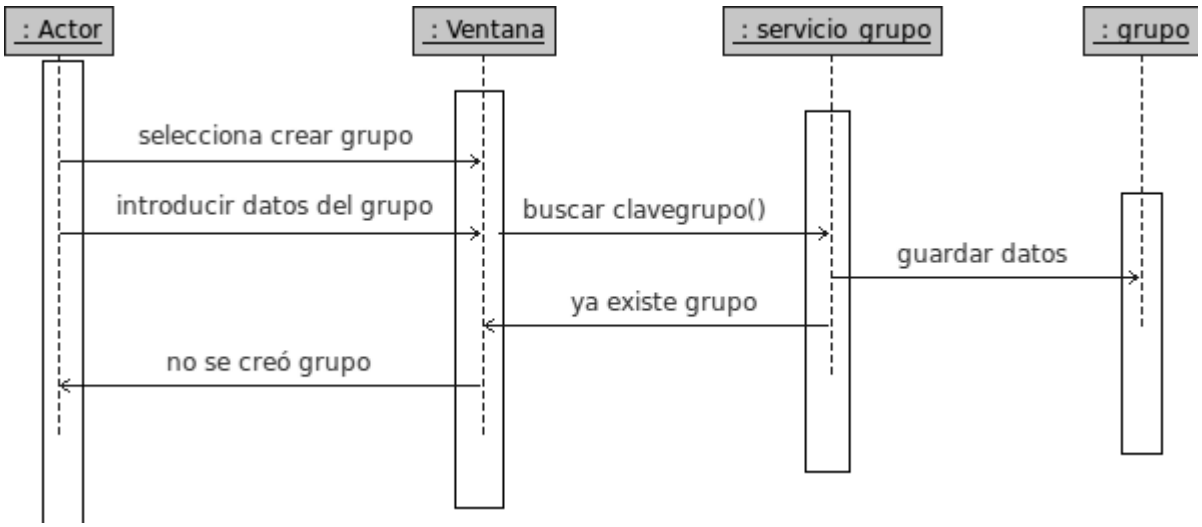
6.1 Escenario de uso: Crear modalidad de evaluación



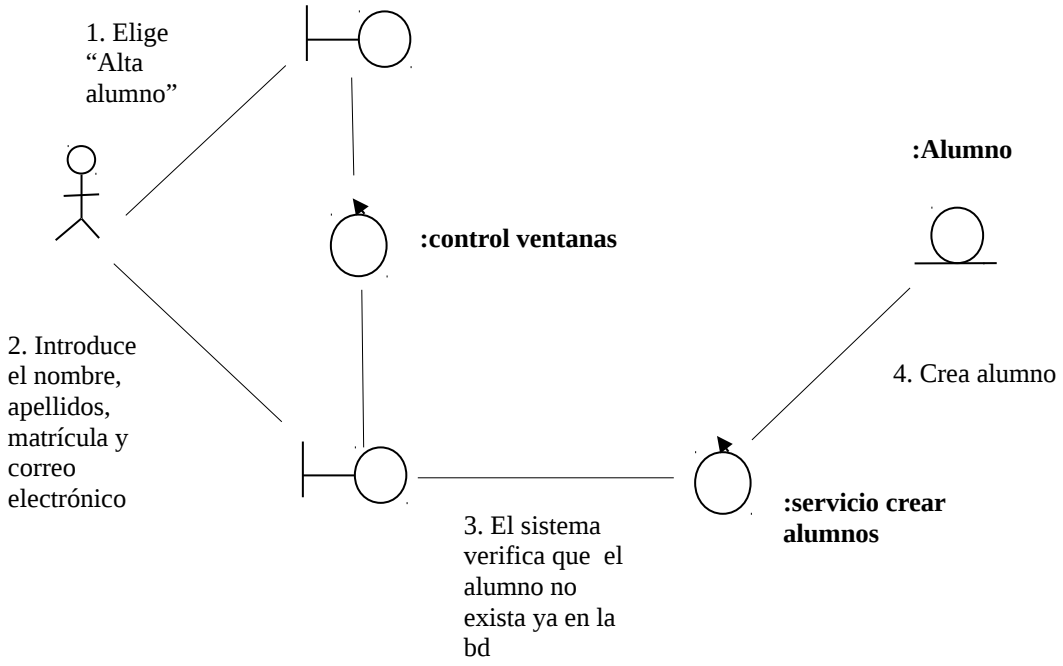


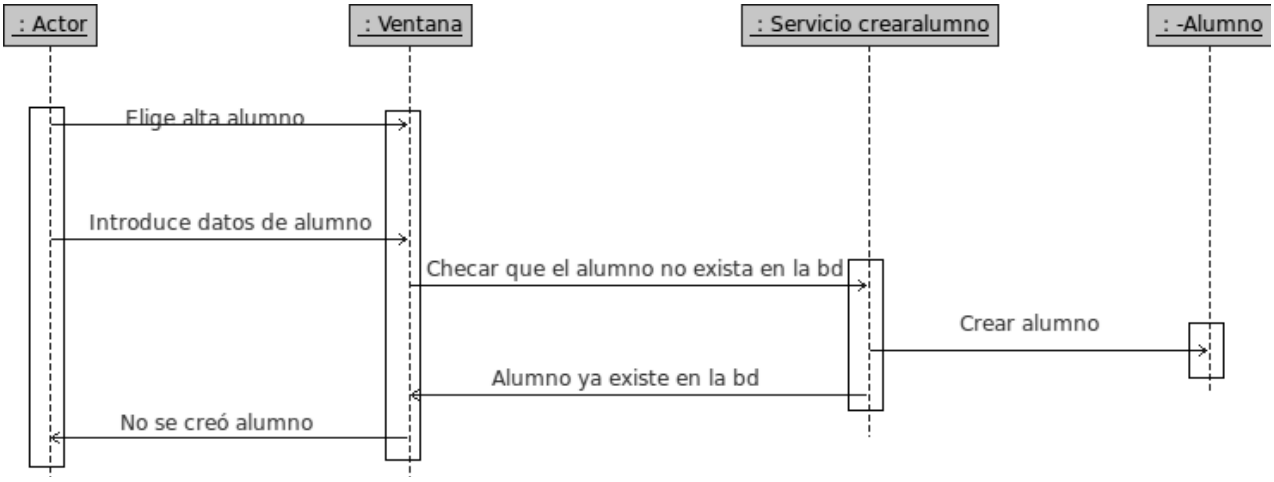
6.2 Escenario de uso: Alta de grupo de alumnos



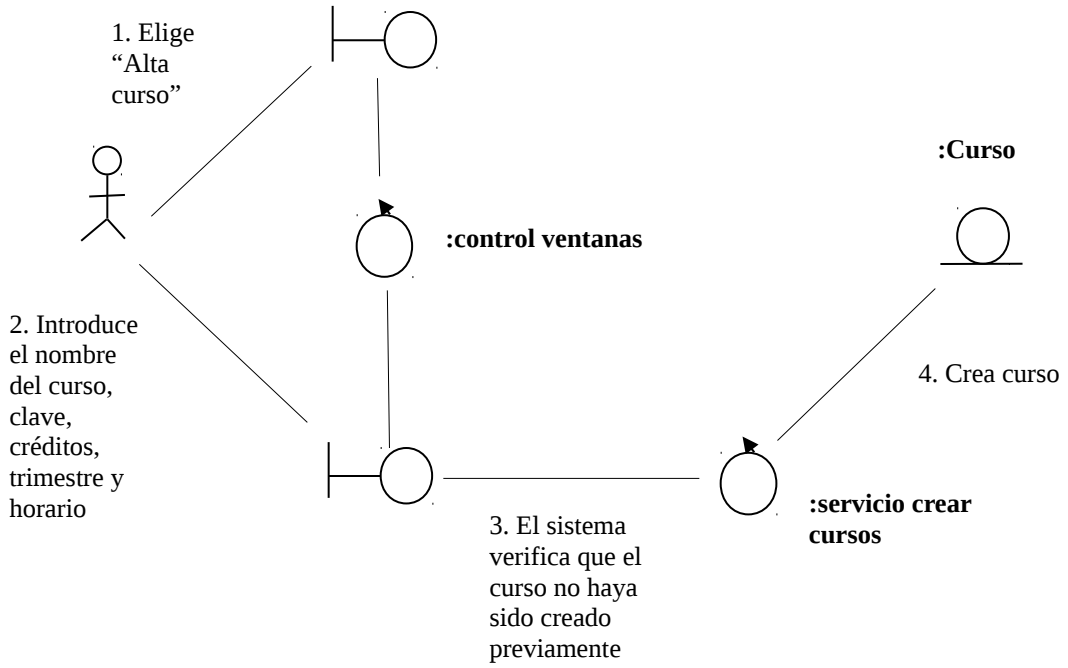


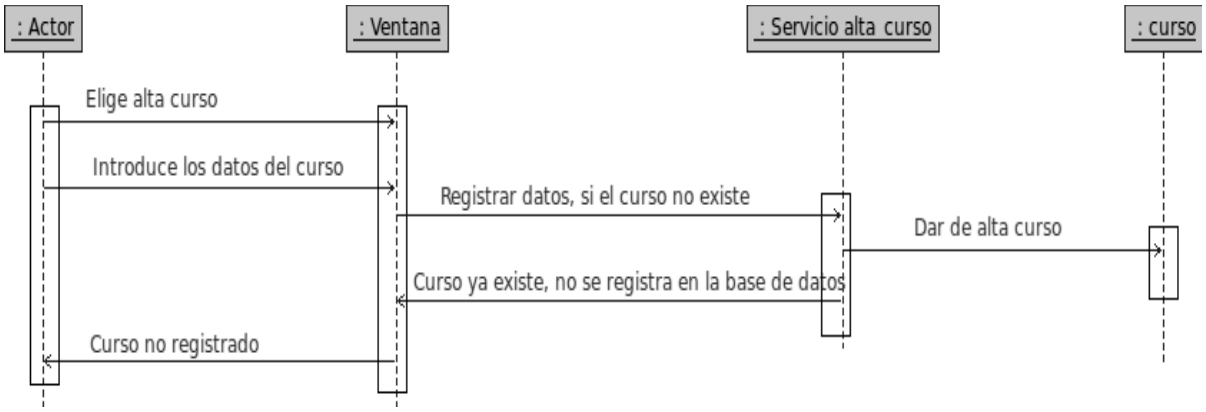
6.3 Escenario de uso: Alta de alumno



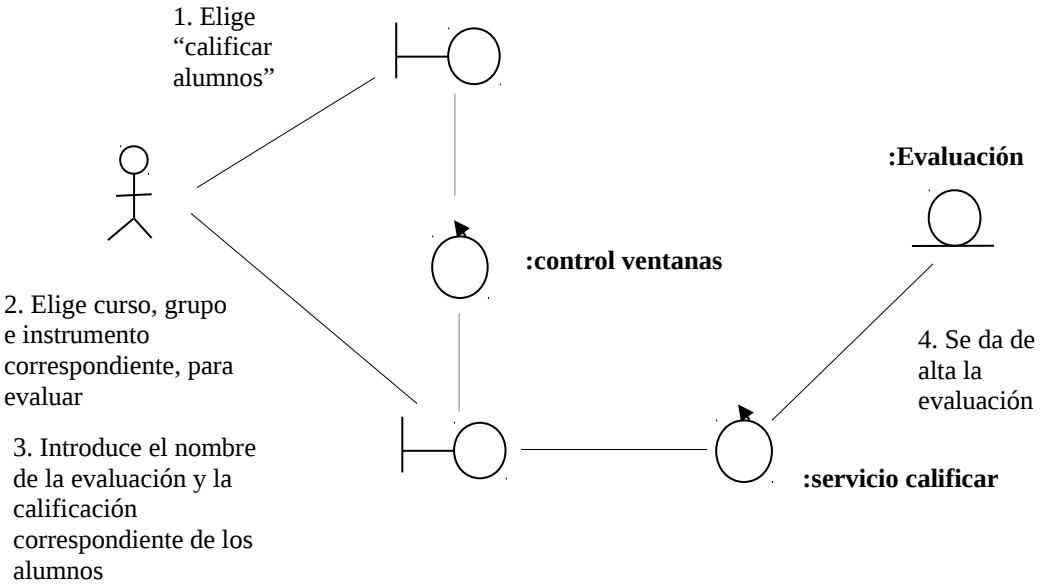


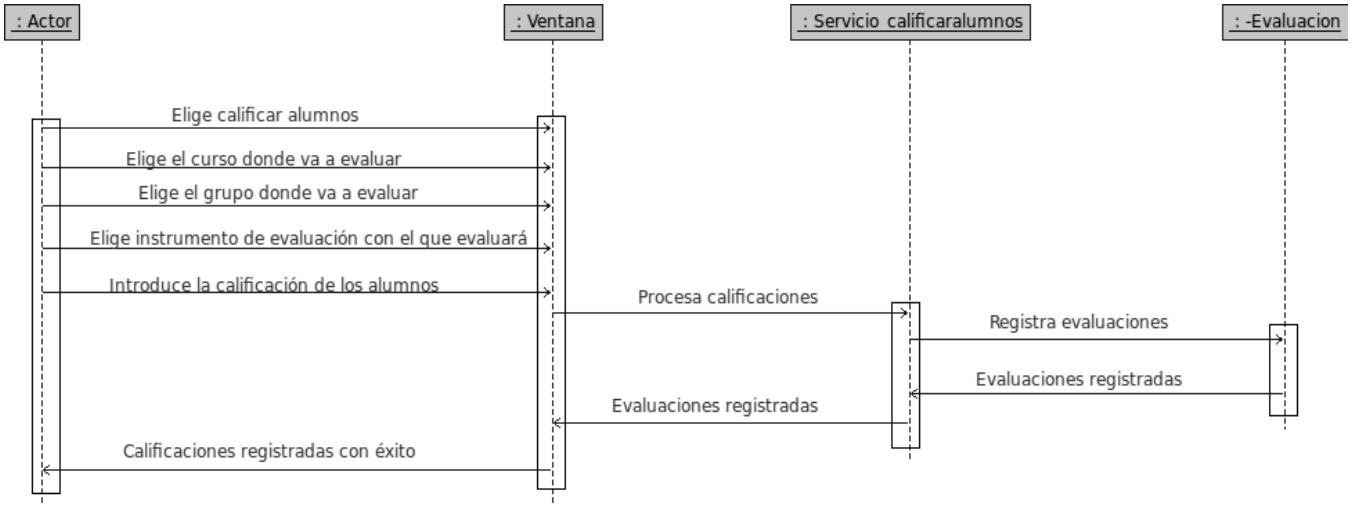
6.4 Escenario de uso: Alta de curso



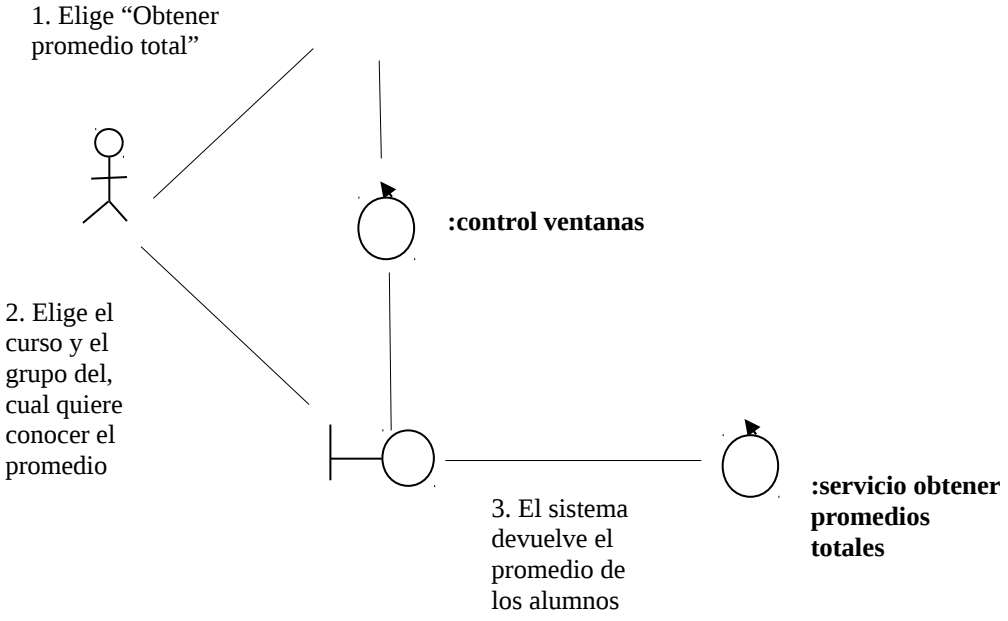


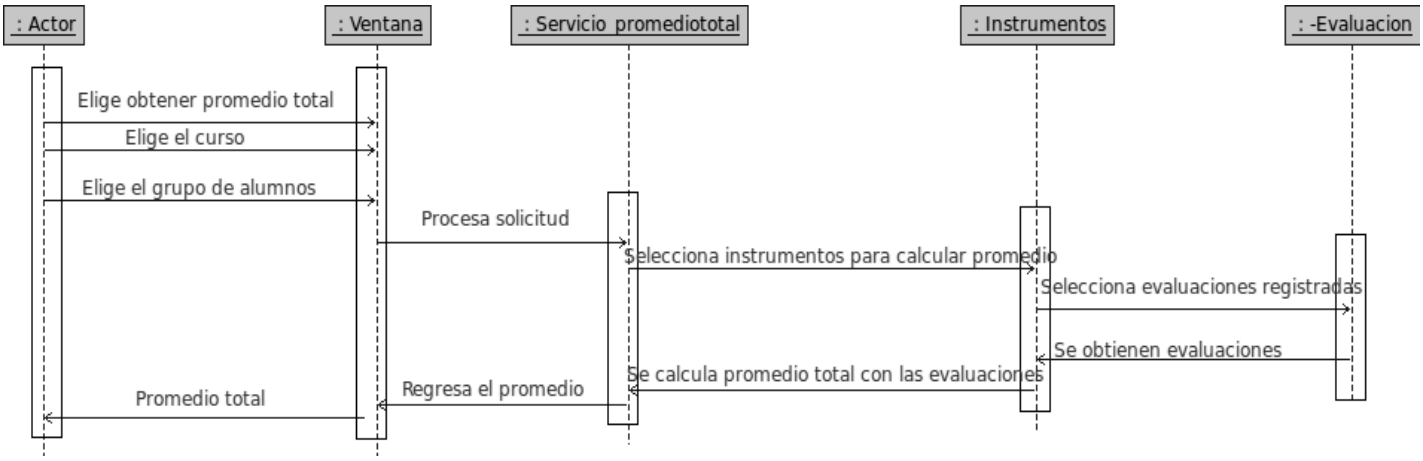
6.5 Escenario de uso: Calificar alumnos





6.6 Escenario de uso: Obtener promedio total





7. Arquitectura del sistema

Se definió una arquitectura de 3 capas; su principal ventaja es que se logra una aplicación cuyos módulos tienen un bajo acoplamiento. De esta manera, cuando se requirió implementar un cambio importante, solo se trabajó en la capa correspondiente: datos, aplicación ó presentación.

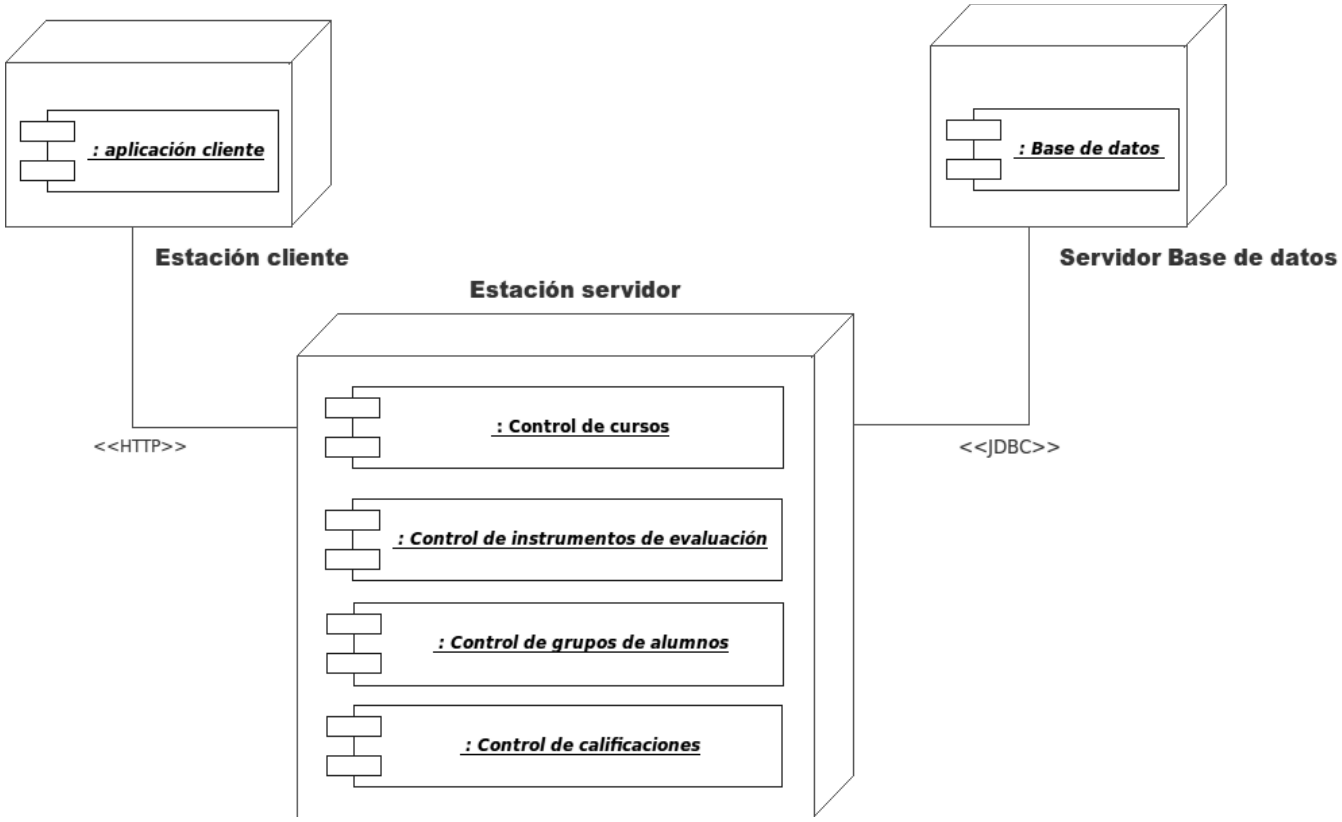


Figura 3. Arquitectura del sistema

La arquitectura a nivel de hardware es de tipo cliente ligero, donde el software se distribuye en dos o más computadoras; se dice que es de cliente ligero porque en el dispositivo del usuario, solo es necesario contar con una parte de la aplicación cuya función principal hacer peticiones al servidor y mostrar los resultados; la lógica del negocio de la aplicación y su procesamiento se lleva a cabo en el servidor.

7.1 Capa de datos

Aquí es donde residen los datos. Para este proyecto, se utilizó el Sistema de Gestión de Base de Datos Relacional MySQL.

7.2 Capa de aplicación

En esta capa residen los recursos y servicios Web que los clientes van a utilizar para realizar sus actividades. Aquí se reciben las peticiones de los usuarios y se envían respuestas tras procesar dichas peticiones. Dependiendo de la petición del usuario a través de la aplicación cliente, se proporcionan en esta capa las funciones necesarias para acceder a los recursos solicitados. Esta capa se comunica con la capa de Presentación para recibir solicitudes y presentar resultados, y con la capa de datos para solicitar al gestor de la base de datos almacenar o recuperar datos de él.

7.3 Capa de presentación

También recibe el nombre de estación cliente. Es la capa que ve el usuario finalmente. Presenta el sistema al usuario, le comunica la información y captura los datos del usuario en un mínimo proceso. Esta capa se comunica únicamente con la capa de aplicación.

8. Esquema de la base de datos

Para mapear los datos de las entidades del modelo de clases, se utilizó para este proyecto un modelo de base de datos relacional, en la cual, todas las tablas son accesibles sin necesidad de acceder a sus tablas padres (como ocurre en las bases de datos jerárquicas).

Para la creación de la base de datos relacional, se tomó como referencia el diagrama de clases del proyecto. En primer lugar, se transformó a cada entidad del modelo de clases en una tabla. Después se definió una llave principal para cada tabla. Y por último, se determinaron asociaciones entre cada tabla.

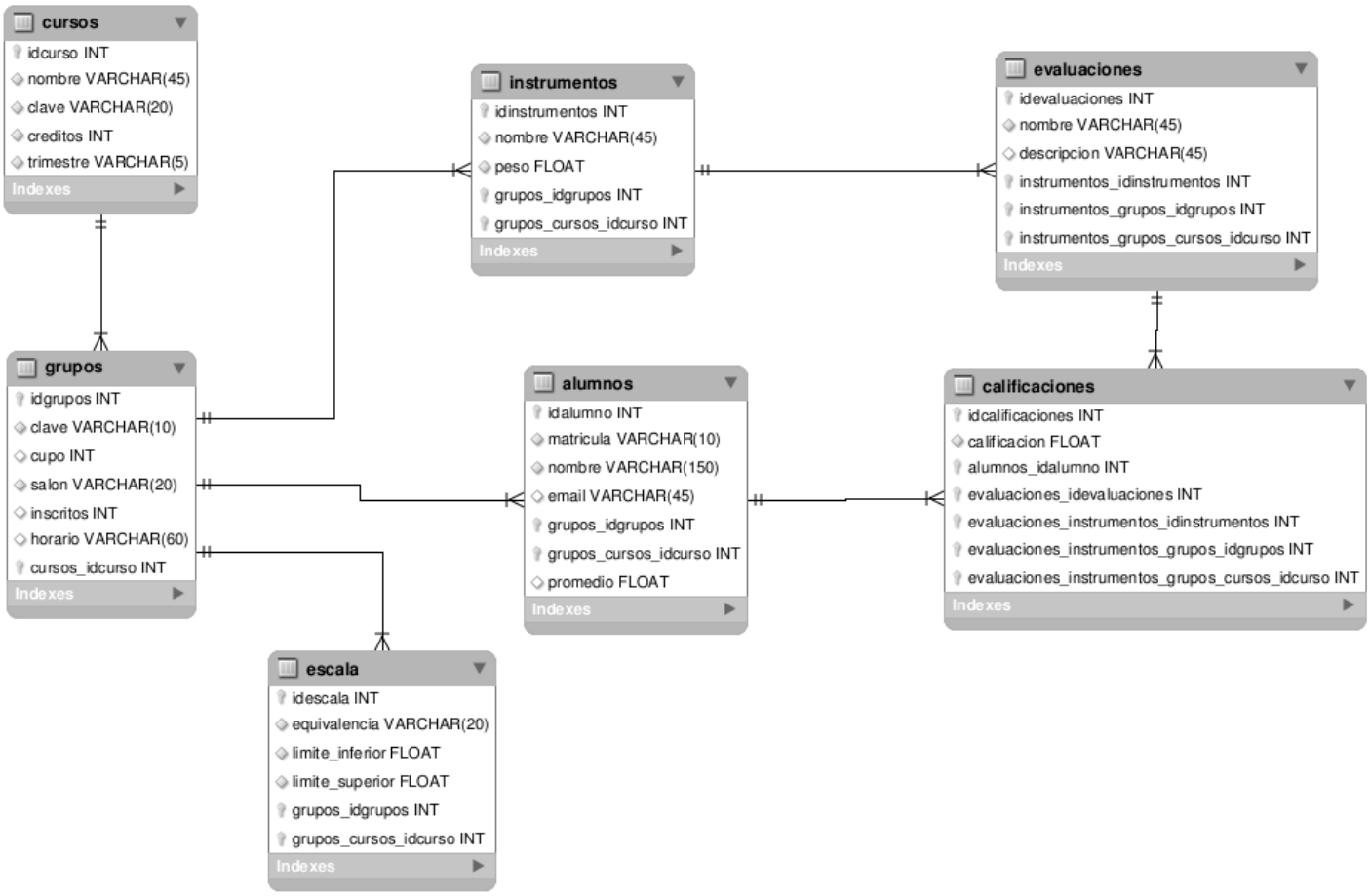


Figura 4. Esquema de la base de datos

8.1 Tabla Cursos

Almacena los datos correspondientes a los cursos que el profesor va a impartir.

Columnas:

- idcursos: llave primaria de la tabla, tipo int, autoincrement
- nombre: tipo varchar
- clave: tipo varchar
- créditos: tipo int
- trimestre: tipo varchar

8.2 Tabla Grupos

Almacenan los datos correspondientes a los grupos de alumnos. Cuenta con una llave foránea "cursos_idcurso" con la cual se establece una relación con la tabla `cursos`.

Columnas:

- idgrupos: llave primaria de la tabla, tipo int, autoincrement
- clave: tipo varchar
- cupo: tipo int
- salon: tipo varchar
- inscritos: tipo int
- horario: tipo varchar
- cursos_idcurso: llave foránea, tipo int

8.3 Tabla instrumentos

Almacena los registros correspondientes a los instrumentos de evaluación que el profesor va a definir para evaluar a un grupo. Cuenta con 2 llaves foráneas para establecer referencias a las tablas `grupos` y `cursos.`

Columnas:

- idinstrumentos: llave primaria de la tabla, tipo int, autoincrement
- nombre: tipo varchar
- peso: tipo float
- grupos_idgrupos: llave foránea, tipo int
- grupos_cursos_idcurso: llave foránea, tipo int

8.4 Tabla escalas

Esta tabla almacena los registros correspondientes a las escalas de calificación para un grupo de alumnos. Cuenta con 2 llaves foráneas para establecer referencias con registros de las tablas `Grupos` y `Cursos`.

Columnas:

- idescala: llave primaria de la tabla, tipo int, autoincrement
- equivalencia: tipo varchar
- limite_inf: tipo float
- limite_sup: tipo float
- grupos_idgrupos: llave foránea, tipo int
- grupos_cursos_idcurso: llave foránea, tipo int

8.5 Tabla alumnos

Almacenan todos los registros correspondientes a los alumnos de un determinado grupo. Cuenta con 2 llaves foráneas para establecer referencias con registros de las tablas `grupos` y `cursos`.

Columnas:

- idalumnos: llave primaria de la tabla, tipo int, autoincrement
- matricula: tipo varchar
- nombre: tipo varchar
- email: tipo varchar
- grupos_idgrupos: llave foránea, tipo int
- grupos_cursos_idcurso: llave foránea, tipo int
- promedio: tipo float

8.6 Tabla evaluaciones

Guarda todos los registros correspondientes a las evaluaciones con las que el profesor calificará a sus alumnos. Se tienen en esta tabla 3 llaves foráneas para establecer referencias con registros de las tablas `instrumentos, grupos` y `cursos`.

Columnas:

- idevaluaciones: Llave primaria de la tabla, tipo int, autoincrement
- nombre: tipo varchar
- descripcion: tipo varchar
- instrumentos_idinstrumentos: llave foránea, tipo int
- instrumentos_grupos_idgrupos: llave foránea, tipo int
- instrumentos_grupos_cursos_idcurso: llave foránea, tipo int

8.7 Tabla calificaciones

Contiene todos los registros correspondientes a las calificaciones emitidas, de acuerdo a los instrumentos y evaluaciones definidos con anterioridad. Se tienen 5 llaves foráneas para establecer referencias con las tablas `alumnos, instrumentos, evaluaciones, cursos,` y `grupos.`

Columnas:

- idcalificaciones: Llave primaria de la tabla, tipo int, autoincrement
- calificacion: tipo float
- alumnos_idalumno: llave foránea, tipo int
- evaluaciones_idevaluaciones: llave foránea, tipo int
- evaluaciones_instrumentos_idinstrumentos: llave foránea, tipo int
- evaluaciones_instrumentos_grupos_idgrupos: llave foránea, tipo int
- evaluaciones_instrumentos_grupos_cursos_idcurso: llave foránea, tipo int

Universidad Autónoma Metropolitana Unidad Azcapotzalco

División de Ciencias Básicas e Ingeniería

Licenciatura en Ingeniería en Computación

Gestión de calificaciones de cursos mediante servicios Web

Manual de usuario

Elaboró:
Avendaño Méndez Sergio Enrique 205304956

Trimestre: 12-O

Fecha:
10 de Diciembre de 2012

Asesores:
Dra. Beatriz Adriana González Beltrán

Dra. Lizbeth Gallardo López

Índice

1. Introducción

Este manual sirve como guía para el usuario de la aplicación que permite la *gestión de calificaciones de cursos mediante servicios Web*. Se explica el funcionamiento de las tres principales secciones de la aplicación:

- Gestión de cursos
- Gestión de grupos
- Consultar historial

A continuación, vamos a mostrar cómo funcionan algunos módulos de la aplicación SEA, partiendo de la pantalla principal; luego, se mostrarán las funcionalidades disponibles de la sección gestión de cursos, gestión de grupos y consulta de historial, las cuales ayudarán al profesor a llevar una adecuada gestión de las calificaciones de sus alumnos.

2. Pantalla inicial

En la figura 1 se muestra el menú principal de la aplicación, el cual consta de tres opciones para llevar a cabo las operaciones de gestión de cursos, gestión de grupos de alumnos y consultar historial. Al elegir una opción se desplegará la ventana correspondiente a la sección elegida.

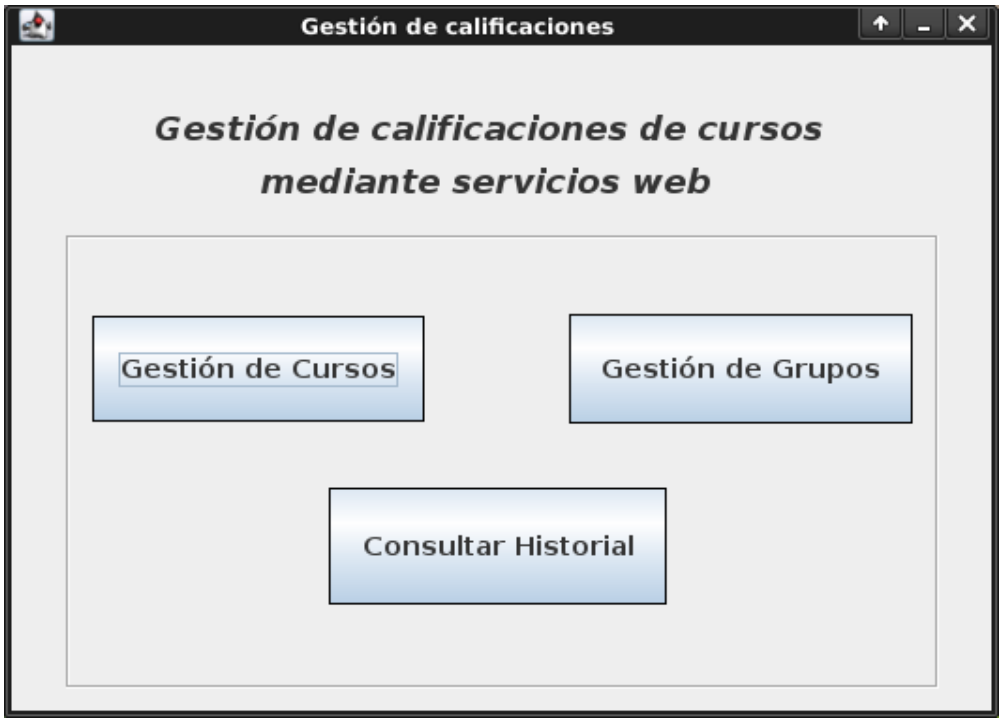


Figura 1. Pantalla inicial de la aplicación

3. Sección gestión de cursos

La sección de cursos (ver figura 2) cuenta con opciones que permiten visualizar los cursos existentes en el sistema, dar de alta un nuevo curso, modificar la información de un curso ya existente, deshabilitar un curso y eliminar un curso.

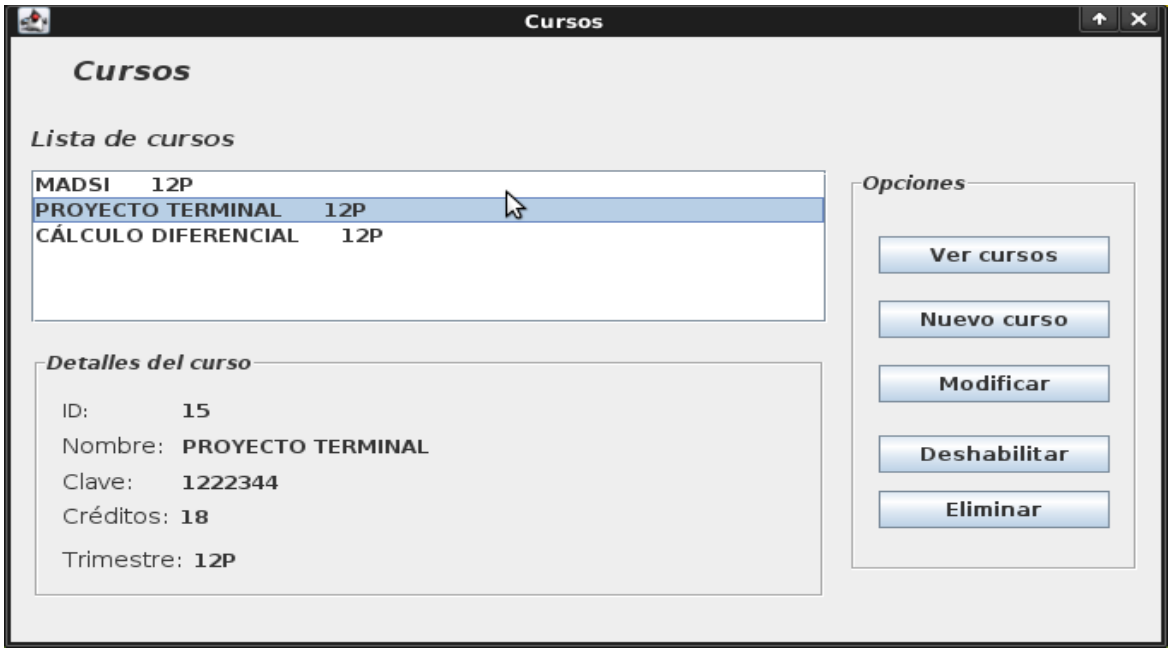


Figura 2. Ventana principal de la sección cursos

3.1 Opción ver cursos

Al dar clic sobre esta opción, la aplicación muestra una lista de los cursos existentes en el sistema. Al elegir cada uno de los cursos de la Lista, se muestran sus datos en el recuadro Detalles del curso.

3.2 Opción nuevo curso

Esta opción permite al usuario dar de alta un nuevo curso en el sistema. Al elegir esta opción, se muestra una nueva pantalla *Alta curso* (figura 3), en la cual el usuario va a poder escribir los datos del curso correspondiente. El usuario puede dar de alta en esta ventana todos los cursos que desee.

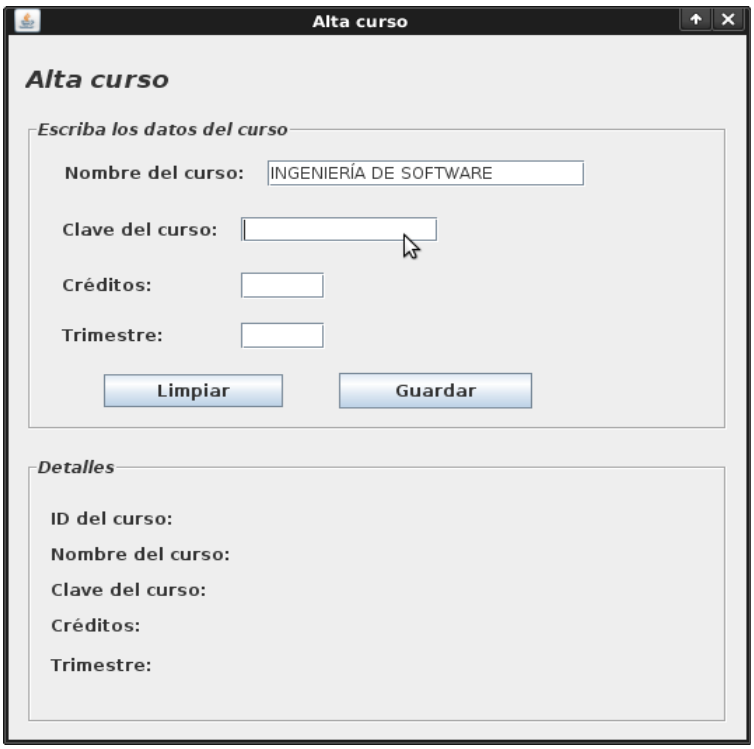


Figura 3. Ventana Alta curso

3.3 Opción modificar

Con esta opción se abre una nueva ventana *Actualiza curso* (ver figura 4), con la cual el usuario puede modificar la información de un curso que ya existe en la base de datos del sistema. El usuario debe seleccionar previamente un curso de la lista para poder utilizar esta opción.

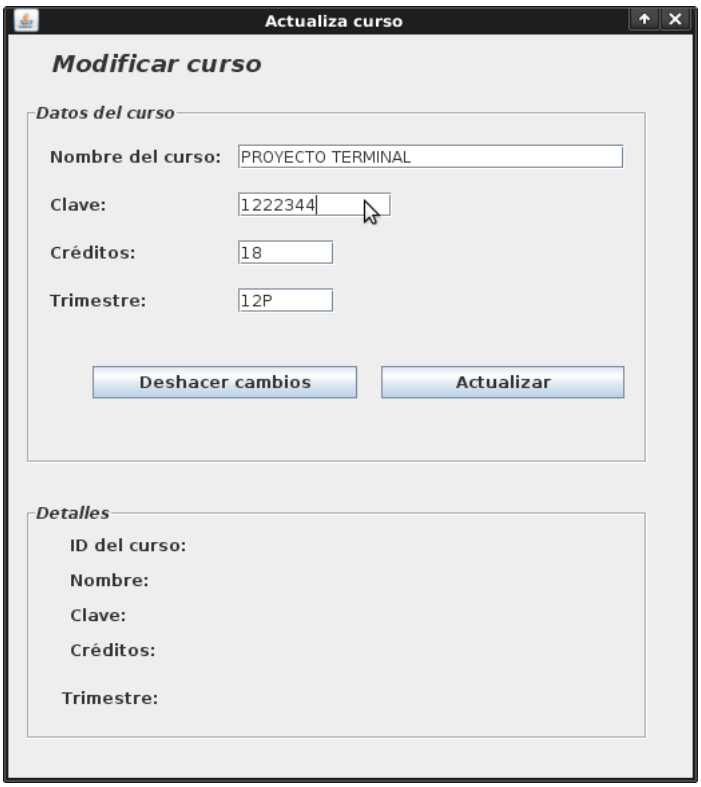


Figura 4. Ventana modificar curso

3.4 Opción deshabilitar

Esta opción permite al usuario deshabilitar el curso seleccionado, de esta manera, ya no se podrán efectuar operaciones sobre este curso ni con grupos de alumnos asociados a éste. Sin embargo, la información del curso estará disponible en la sección *Consultar Historial* de esta aplicación.

3.5 Opción eliminar

Permite al usuario eliminar la información de un curso existente. El sistema eliminará la información del curso completamente solo si no se han definido para este curso grupos de alumnos, instrumentos y evaluaciones. Si ya hay información asociada al curso, entonces el sistema no podrá eliminar la información del curso. Esto se hace para garantizar la integridad del sistema y para cuidar la información que el profesor utiliza para calificar a sus alumnos.

4. Sección gestión de grupos

En esta sección se encuentran todas las opciones que permiten gestionar los grupos de alumnos (ver figura 5). Entre éstas opciones se encuentran: ver grupos, nuevo grupo, modificar grupo y eliminar grupo. Todas ellas operan de manera similar a las opciones de la sección anterior.

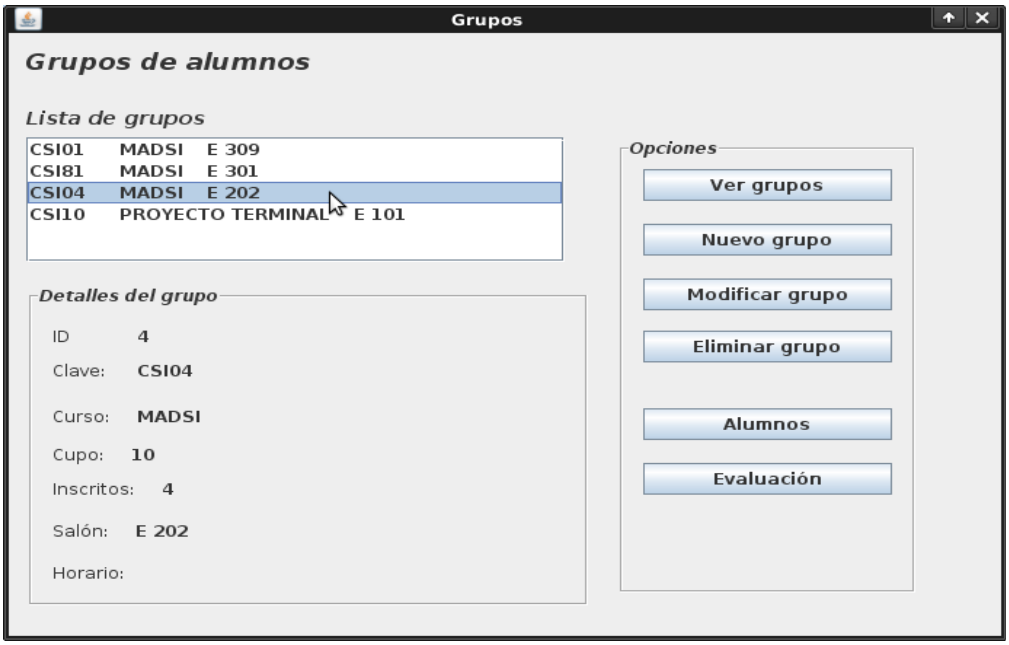


Figura 5. Ventana principal de la sección grupos de alumnos

En la sección grupos se incluyen 2 opciones más. La opción *Alumnos*, que brinda las funciones necesarias para la gestión de alumnos y la opción *Evaluación*, la cual tiene las funciones que permiten definir la modalidad de evaluación y calificar a los alumnos del grupo en base a dicha modalidad.

4.1 Ventana Alumnos

Figura 6. Ventana alumnos

En la ventana Alumnos (ver figura 6) existen 2 opciones para dar de alta alumnos en el sistema. La primera de ellas es *Alta de alumno*, la cual permite al usuario dar de alta un solo alumno a la vez. Para ello la aplicación despliega un formulario para que el usuario escriba la información del alumno.

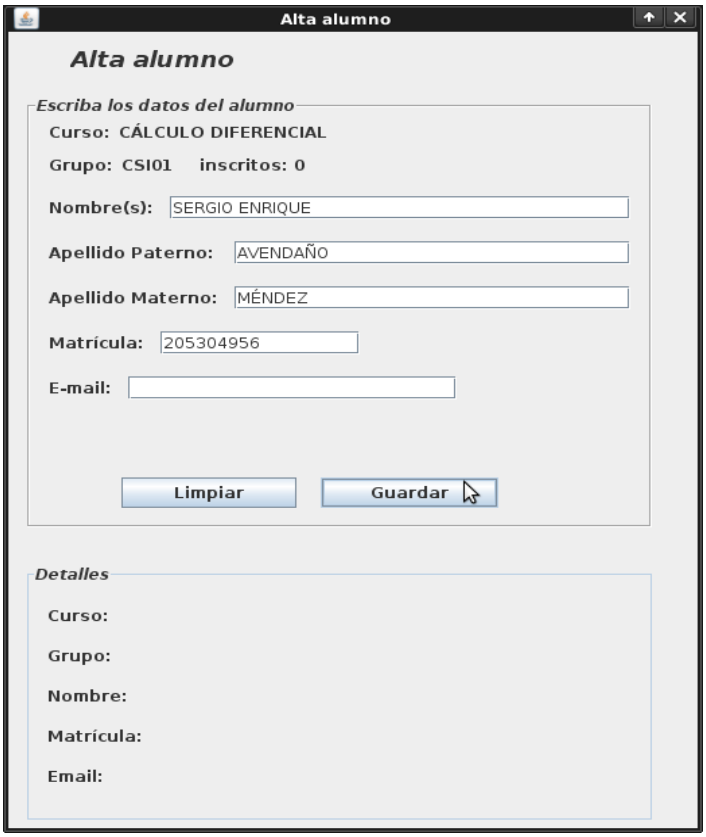


Figura 7. Ventana Alta alumno

La segunda opción es *Cargar lista*. Esta opción permite dar de alta varios alumnos a la vez, provenientes de un archivo de texto.

Figura 8. Cargar lista de alumnos

Para que la aplicación pueda obtener la información de los alumnos adecuadamente, el archivo de texto con la lista debe tener el siguiente formato:

	ALUMNOS	
No.	Nombre	Matrícula
1	NOMBRE	MATRICULA
2	NOMBRE	MATRICULA
3	NOMBRE	MATRICULA
4	NOMBRE	MATRICULA
.		
.		
.		

Tabla 1. Formato del archivo con la información de los alumnos

4.1.1 Opción modificar

Con esta opción el usuario puede modificar la información de alumnos ya existentes en el sistema.

4.1.2 Opción eliminar

Permite al usuario eliminar alumnos del sistema.

4.2 Ventana evaluación

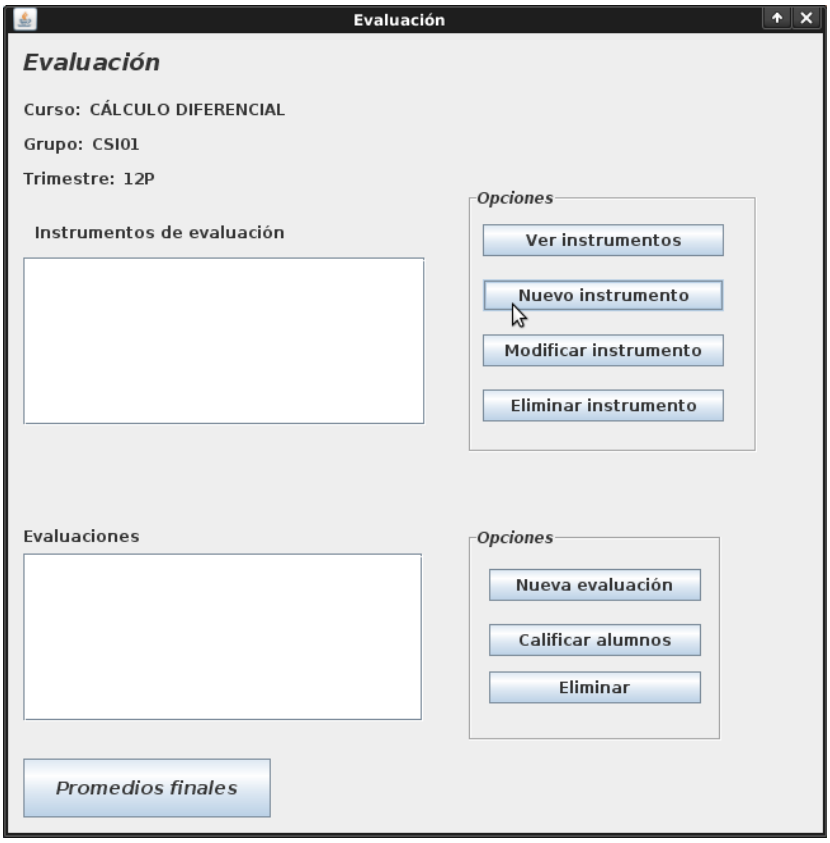


Figura 9. Ventana evaluación

En la ventana *evaluación* (ver figura 9), se define la modalidad de evaluación con la que se calificará a los alumnos del grupo. Es aquí donde se pueden definir los instrumentos de evaluación y las evaluaciones asociadas.

Las opciones *ver instrumentos, nuevo instrumento, modificar instrumento y eliminar instrumento*, trabajan de manera similar que en los apartados anteriores.

Una vez que se han definido instrumentos de evaluación junto con sus pesos de la calificación final, se determinan las evaluaciones asociadas a estos. Por ejemplo, si el usuario define para evaluar a sus alumnos un instrumento de evaluación llamado *Exámenes*, se podrían definir evaluaciones como *Primer Parcial*, *Segundo Parcial*, *Tercer Parcial*.

Para asociar un instrumento, con una evaluación, se selecciona el instrumento en cuestión, y se da clic en la opción *Nueva Evaluación* de la misma ventana. En la figura 10 se pueden observar 2 instrumentos de evaluación definidos por el usuario (*Exámenes* y *tareas*) y una evaluación asociada a uno de éstos (La evaluación examen parcial 1 está asociada al instrumento *Exámenes*).

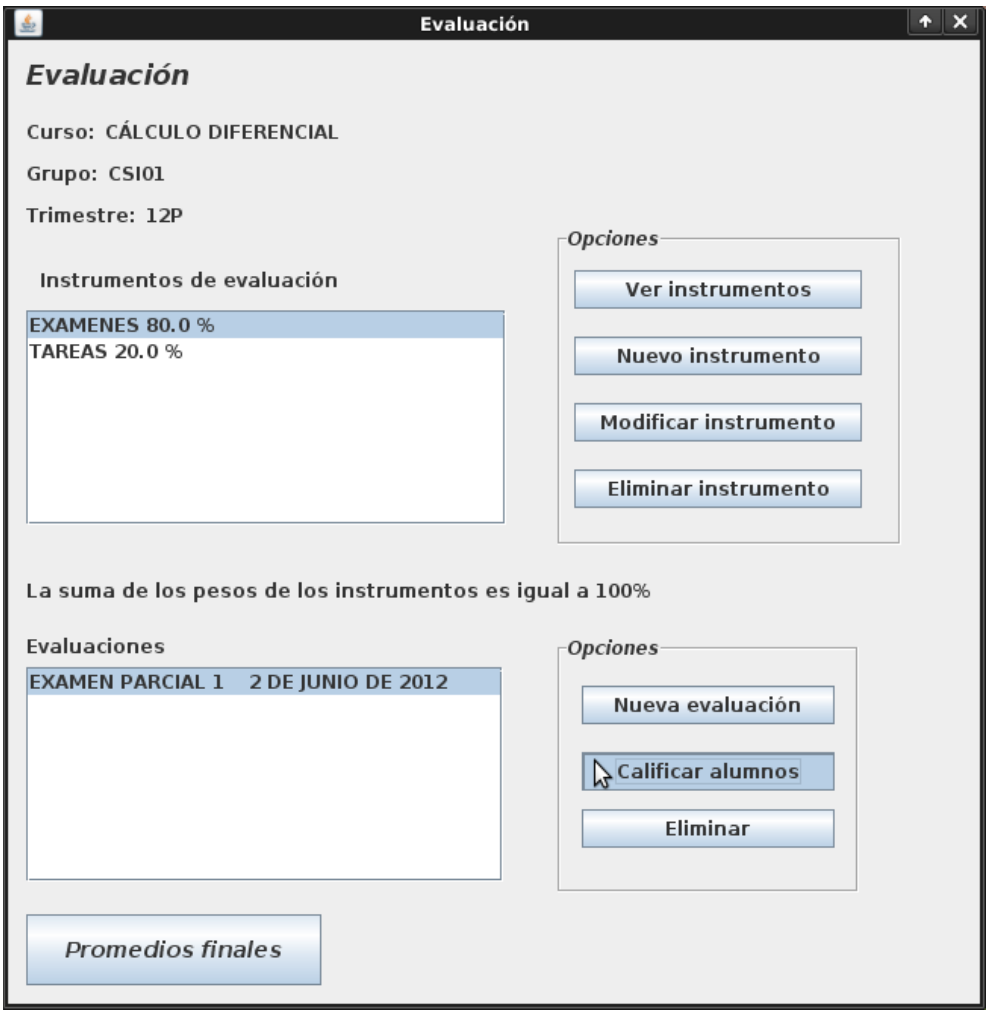


Figura 10. La evaluación Examen Parcial 1 está asociada al instrumento Exámenes

En la tabla 2 tabla se mostrará un escenario real, donde se explican los pasos a seguir para definir en el sistema los instrumentos, criterios y evaluaciones con los que un profesor podrá calificar a sus alumnos.

El usuario (profesor) ya ha dado de alta en el sistema un curso, un grupo y varios alumnos a los cuales, deberá evaluar una vez termine el trimestre.

El profesor decide que para obtener la calificación final de sus alumnos utilizará los siguientes instrumentos de evaluación:

- Tareas 20 %
- Examen 80 %

Para definir la información de la evaluación del grupo en el sistema, el profesor debe seguir los

siguientes pasos:

1. Para dar de alta esta información en el sistema, el profesor debe acceder a la ventana *Evaluación* y dar clic en el botón *Nuevo Instrumento*. De esta manera podrá introducir en el sistema los instrumentos *Tareas* y *Exámenes*, con su peso correspondiente.

2. Para el curso que va a impartir, el profesor considera que 3 exámenes y 5 tareas son suficientes para evaluar a los alumnos inscritos. Para dar de alta esta información el profesor selecciona el instrumento tareas de la lista de instrumentos definidos y luego da clic en la opción *nueva evaluación.* De esta forma el profesor puede introducir el nombre de las tareas que va a emplear para evaluar al grupo.

3. Cuando terminar de definir las tareas, emplea los mismos pasos para definir los 3 exámenes que va a aplicar a los alumnos.

4. Si durante el transcurso del trimestre decide agregar una evaluación más, el profesor puede emplear los mismos pasos para seguir definiendo evaluaciones.

5. Cuando ya están dadas se han dado de alta las evaluaciones, el profesor puede comenzar a calificar a sus alumnos.

Tabla 2. Escenario real para definir las evaluaciones de un curso.

Cuando ya se han definido instrumentos de evaluación y evaluaciones asociadas a éstos, ya es posible calificar a los alumnos del grupo en base a estos criterios.

Al elegir la opción calificar (ver figura 11) alumnos aparecerá una pantalla con la lista de los alumnos registrados en el grupo junto con un espacio para colocar la calificación correspondiente a cada uno de ellos.

Figura 11. Calificar alumnos

Las calificaciones quedarán guardadas en el sistema al elegir la opción *Guardar calificaciones*.

4.3 Promedios finales

La opción *Promedios* (ver figura 12) de la ventana Evaluación, permite calcular los promedios finales de todos los alumnos en base a los instrumentos de evaluación y criterios definidos anteriormente.

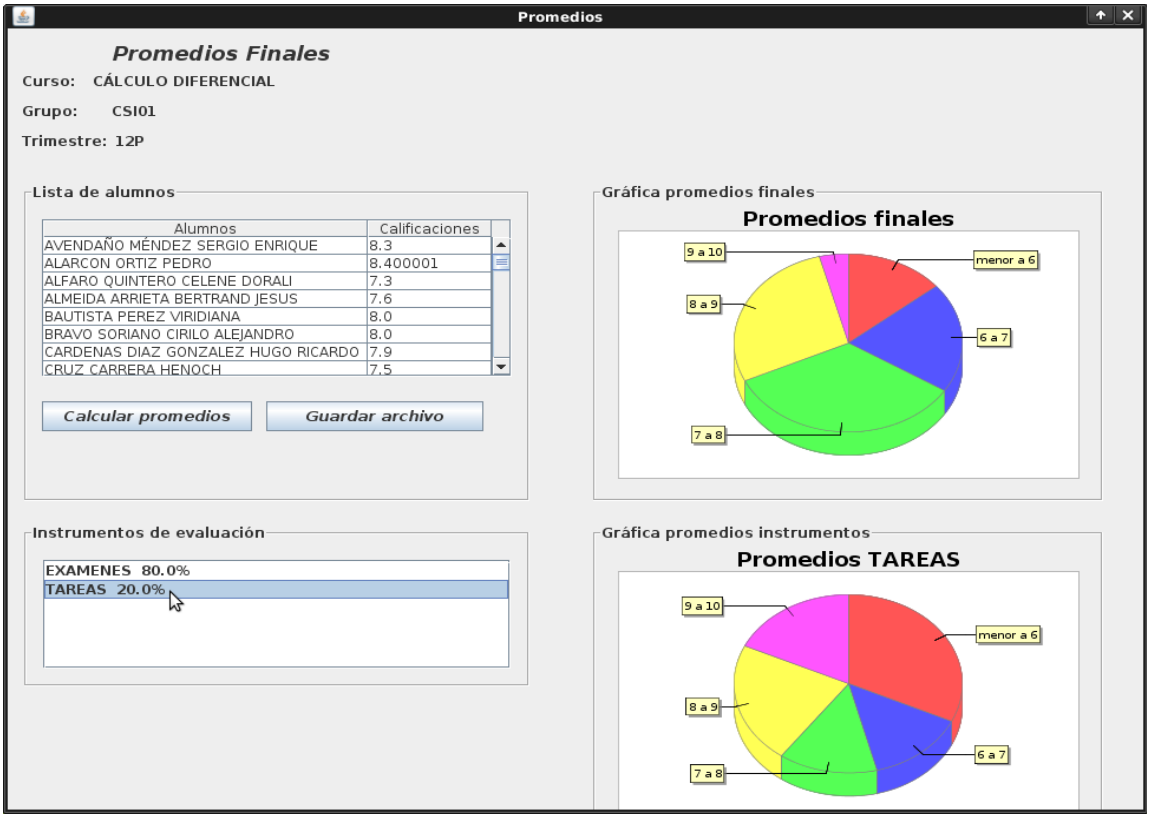


Figura 12. Ventana promedios

Al finalizar la operación *calcular promedios* se pueden visualizar gráficas, en las cuales se puede observar como estuvieron distribuidos los promedios finales y los promedios de los instrumentos entre los alumnos. En la ventana Promedios, existe una opción *Guardar archivo*, la cual permite guardar en un archivo de texto la lista de alumnos del grupo, junto con las calificaciones finales que les corresponden.

5. Sección Consultar Historial

En esta sección, se recuperan los datos de los grupos existentes en el sistema..

Se recupera la lista de alumnos inscritos en el grupo, los promedios finales, la modalidad de evaluación utilizada para evaluar al grupo y una gráfica que muestra la distribución de las calificaciones finales.

En la ventana principal de esta sección(ver figura 13), aparecerá la lista de grupos de alumnos existentes en el sistema. Para consultar su historial, basta con elegir uno de estos grupos y dar clic en el botón consultar.

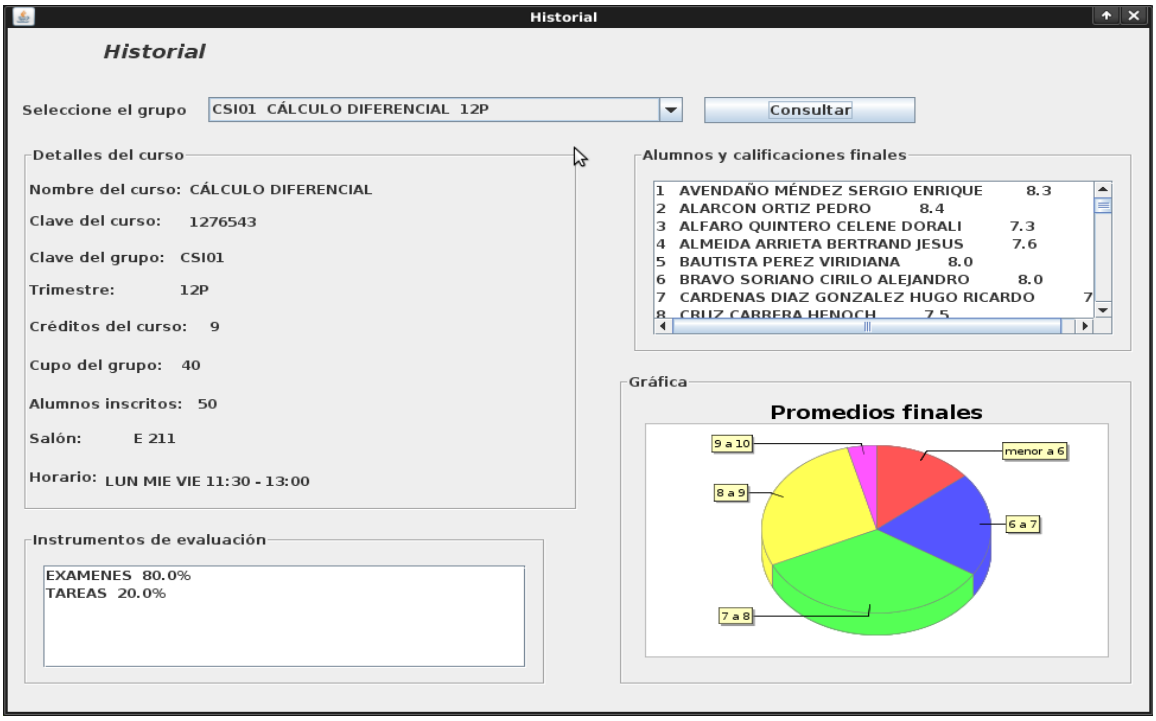


Figura 13. Historial de grupos

Universidad Autónoma Metropolitana Unidad Azcapotzalco

División de Ciencias Básicas e Ingeniería

Licenciatura en Ingeniería en Computación

Gestión de calificaciones de cursos mediante servicios Web

Manual técnico

Elaboró:
Avendaño Méndez Sergio Enrique
205304956

Trimestre: 12-O

Fecha:
10 de Diciembre de 2012

Asesores:
Dra. Beatriz Adriana González Beltrán

Dra. Lizbeth Gallardo López

Índice

1. Introducción

El objetivo principal del proyecto fue desarrollar un sistema que permitiera a un profesor llevar el control adecuado de las calificaciones de sus alumnos en los diferentes cursos que imparte. Este sistema (SEA- Sistema de Evaluación de Alumnos) funciona con servicios Web basados en REST, los cuales son invocados desde una aplicación cliente.

El manual técnico tiene como objetivo mostrar la tecnología que se utilizó durante el proyecto; así como la forma de instalar la aplicación en el servidor y en el cliente.

2. Hardware utilizado

La aplicación fue desarrollada e instalada en una Laptop Compaq Presario C700 con los siguientes recursos:

- Memoria RAM de 1 Gb
- Disco duro de 120 Gb
- Procesador Intel(R) Celeron (R) a 1.74 Ghz
- Sistema Operativo Ubuntu 10.04 LTS

3. Software utilizado

El software utilizado para el desarrollo del proyecto es licencia libre. A continuación se describe cada uno de ellos.

3.1 NetBeans IDE

Es una herramienta para escribir, compilar, depurar y ejecutar programas. Está escrito en Java, sin embargo, puede servir para programar en cualquier lenguaje de programación. Hubo dos razones importantes para elegir este IDE; en primer lugar, porque es un producto de libre distribución, por lo cual no hay restricciones en su uso. En segundo lugar, porque contiene una gran cantidad de componentes de software, con los cuales, se pueden construir aplicaciones Web. La versión utilizada para desarrollar el proyecto fue la 6.9.1.

3.2 Servidor de aplicaciones GlassFish

Es un servidor de aplicaciones que implementa la plataforma Java EE6, por lo que soporta las últimas versiones de tecnologías como: JSP, JSF, Servlets, EJBs, Java API para servicios Web (JAX-WS), entre otras tecnologías. La versión de GlassFish utilizada para desarrollar el proyecto fue la versión 3.0.1, debido a que para el proyecto, se utilizaron algunas características de Java EE6[1].

1 Java Plataform Enterprise Edition. Es una plataforma de programación para desarrollar y ejecutar software de aplicaciones en el lenguaje de programación Java. Permite utilizar arquitecturas de N capas distribuidas y se apoya en componentes de software modulares ejecutándose sobre un servidor de aplicaciones.

3.3 Jersey

Es una implementación propuesta por Sun para JAX-RS (API para servicios Web basados en REST). Se utilizó Jersey para desarrollar servicios Web que respondan al protocolo HTTP. Con esta API[2] se logró determinar que clases actuarán como servicios Web. La versión utilizada de Jersey para este proyecto fue la 1.1.2.

3.4 Hibernate

Es una herramienta de Mapeo-Objeto-Relacional (ORM) para la plataforma java, el cual facilita el mapeo de los atributos de una base de datos relacional hacia el modelo de objetos de una aplicación; utilizando para el mapeo, archivos de configuración XML. Hibernate es también un software de libre distribución. Se utilizó esta herramienta para el mapeo entre las tablas de la base de datos hacia objetos de Java, lo cual permitió reducir el tiempo de desarrollo porque disminuye la carga de la escritura de consultas SQL; de lo contrario, la obtención y guardado de los datos en la base de datos habría tenido que realizarse manualmente, empleando SQL y JDBC. La versión de hibernate utilizada para este proyecto fue la 3.3.1.

3.5 Manejador de base de datos MySQL

Es un sistema de gestión de bases de datos relacionales, multihilo y multiusuario. Es ideal para entornos donde la lectura de datos puede llegar a ser intensiva. MySQL es ofrecido bajo la licencia GNU GLP para cualquier uso compatible con esta licencia. La versión utilizada para el desarrollo del proyecto fue la 5.1.63.

3.6 MySQL Workbench

Es una herramienta visual de diseño de bases de datos que integra: diseño, creación y gestión de bases de datos MySQL. Para usar esta herramienta, es necesario tener instalado previamente, el manejador de base de datos MySQL. La versión utilizada para este proyecto fue la 5.2.35.

2 Interfaz de programación de aplicaciones (API – *Application Programming Interface*). Conjunto de funciones o métodos que exponen las clases de alguna biblioteca para ser utilizado por otro software.

3.7 Lenguaje de programación

El código de la aplicación cliente como de los servicios Web, fue implementado con el lenguaje de programación orientado a objetos Java.

3.8 Sistema operativo

Todas las tecnologías mencionadas anteriormente, fueron montadas sobre el Sistema Operativo Linux Ubuntu en la versión 10.04 LTS.

3.9 REST en Netbeans

NetBeans cuenta con una funcionalidad llamada *TEST RESTFUL WEB SERVICES*, con la cual se puede probar el funcionamiento de un servicio Web basado en REST, utilizando un navegador Web; donde el navegador Web actúa como cliente (ver figura 1). De esta manera el navegador Web invoca al servicio que le devolverá como resultado un documento XML con la información de un alumno.

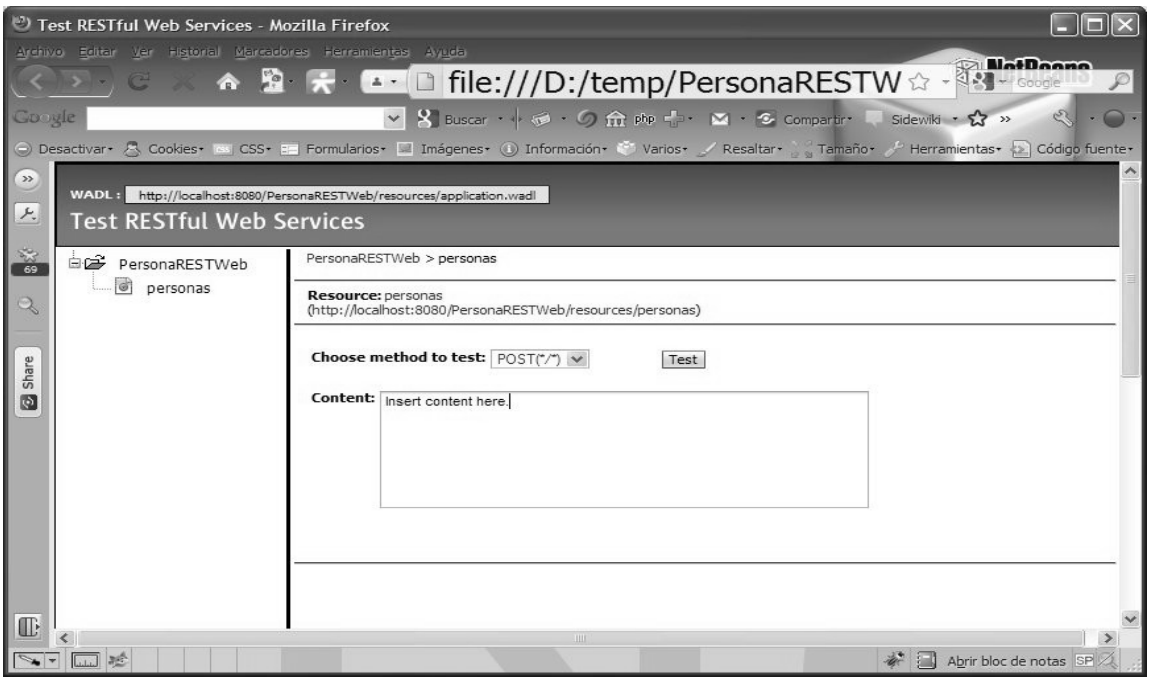


Figura 1. Funcionalidad Test RESTful Web Services del NetBeans

Utilizando NetBeans y el navegador Web como cliente se pudo probar la funcionalidad de cada servicio Web que se desarrolló para el SEA, sin tener que programar una aplicación cliente para ello.

4. Estructura del directorio de la aplicación

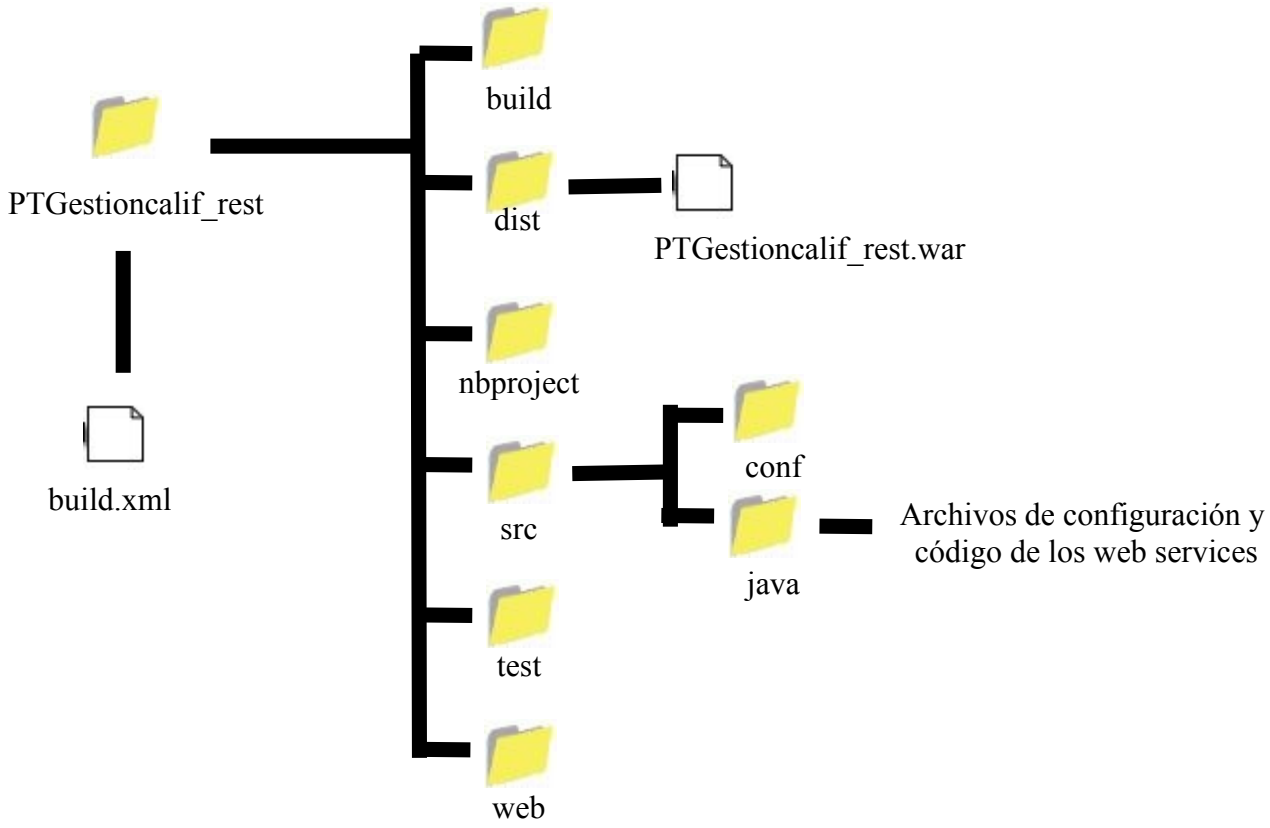


Figura 2. Estructura del directorio que contiene los servicios Web

El directorio *PTGestioncalif_rest* contiene todo el código correspondiente a los servicios Web que permitirán la gestión de calificaciones de cursos. En la figura 2 se puede observar la ubicación del archivo *PTGestioncalif_rest.war* (directorio dist), el cual es muy importante a la hora de instalar los servicios Web en el servidor de aplicaciones GlassFish.

En el directorio *java* se encuentran localizados los archivos con el código de los servicios Web y los archivos de configuración para la base de datos. Si la base de datos va a alojarse en una máquina remota es necesario cambiar algunas líneas de los archivos de esta carpeta. Mas adelante se explicará como hay que realizar estos cambios.

En la figura 3 se muestra la estructura del directorio que corresponde a la aplicación cliente.

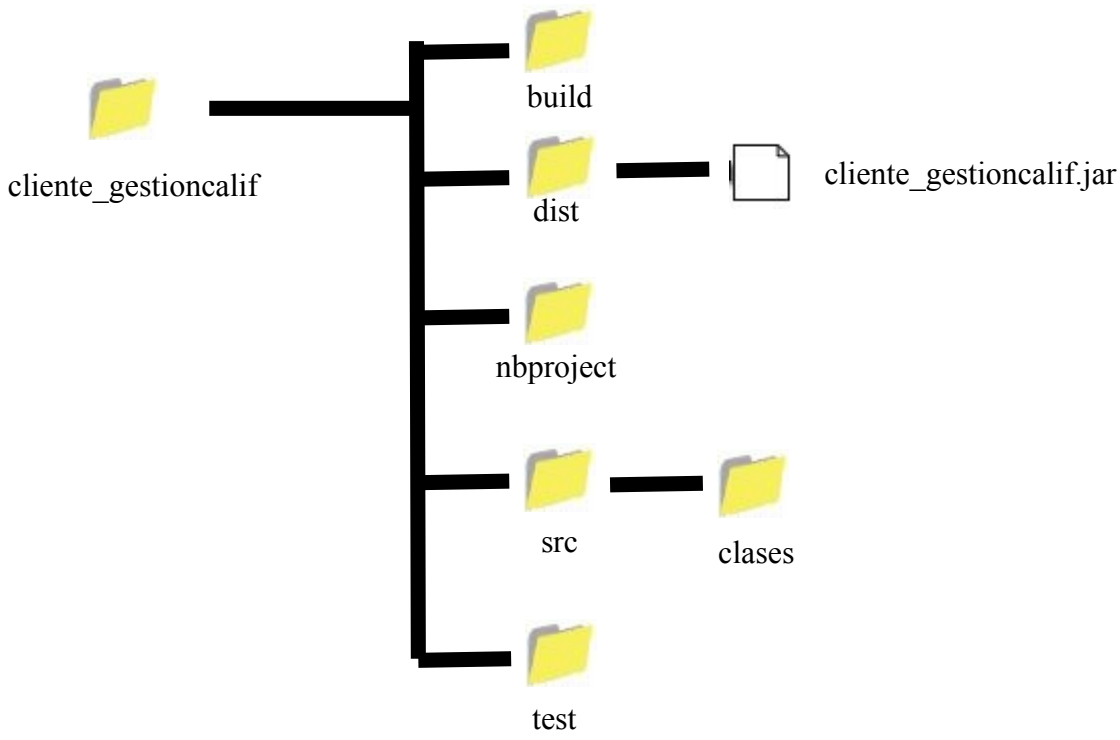


Figura 3. Estructura del directorio de la aplicación cliente

El directorio *cliente_gestioncalif* contiene el código correspondiente al cliente que va a solicitar los servicios para llevar a cabo la gestión de calificaciones. En este directorio se encuentra el código de las interfaces que van a interactuar con el usuario. Para desplegar la aplicación cliente lo único que hay que hacer es ejecutar el comando *java -jar cliente_gestioncalif.jar* desde la carpeta *dist*, ó si se trata de una máquina con windows basta con dar doble clic sobre él.

En la carpeta *src* se ubican el código correspondiente a la interfaz de usuario de la aplicación, también se ubica el código necesario para invocar los servicios Web indicados para gestionar las calificaciones de los alumnos.

5. Proceso de instalación de la aplicación

Para instalar el sistema correctamente debemos asegurarnos de que la máquina servidor donde serán alojados los servicios Web, deberá tener acceso a internet o a una red local; además, deberá tener instalado y configurado adecuadamente un servidor de aplicaciones Tomcat o GlassFish, para que puedan responder a las peticiones de la aplicación cliente. Del mismo modo, los clientes deben tener acceso a internet o a la red local donde se encuentre el servidor para poder enviar sus peticiones y recibir respuestas, con el objetivo de hacer una correcta gestión de calificaciones de los alumnos. Los pasos a seguir para la instalación son:

5.1 Instalar servicios Web en la máquina servidor

Para instalar los servicios Web en un servidor se deberán instalar y configurar adecuadamente el servidor de aplicaciones Glassfish. Luego se debe copiar el archivo:

\PTGestioncalif_rest\dist\PTGestioncalif_rest.war en
\GlassFishv3\GlassFish\domains\domain1\autodeploy

Al colocar el archivo .war en esta carpeta, el servidor GlassFish automáticamente desplegará en la máquina servidor los servicios Web necesarios para llevar a cabo la gestión de calificaciones, y estarán listos para recibir las peticiones de los clientes.

5.2 Instalar la base de datos

El siguiente paso es cargar la base de datos en la cual se alojarán todos los registros relacionados con la gestión de calificaciones. La información de la base de datos para el Sistema de Evaluación de Alumnos es:

Nombre: PTGestioncalif_rest
Usuario: root
Contraseña: root
 # de tablas: 7
Script: PTGestioncalif_rest.sql

El script generador de la base de datos se localiza en la carpeta de la aplicación *PTGestioncalif_rest* en la siguiente dirección: *PTGestioncalif_rest\src\conf*.

La base de datos puede ser alojada en la misma máquina servidor donde se ubican los servicios Web; o bien, puede ser alojada en una máquina remota.

Para que los servicios Web puedan interactuar con la base de datos, se deberán indicar las

propiedades de la base en el archivo de configuración *hibernate.cfg.xml* ubicado en el directorio: *PTGestioncalif_rest\src\java*

5.3 Configurar los clientes

Una vez que se han instalado los servicios Web y la base de datos, es posible interactuar con ellos usando una aplicación cliente. Esta aplicación cliente puede estar alojada en una máquina remota, sin embargo el cliente debe estar configurado para enviar sus peticiones a la máquina servidor donde se instalaron los servicios Web. Por esta razón, es necesario indicar en la aplicación cliente la dirección del servidor con el que va a interactuar. Para ello, se deberá colocar la dirección IP del servidor en los archivos cliente de la carpeta *cliente_gestioncalif\src\clases*. Los archivos son los siguientes:

- *clientealumno.java*
- *clienteborraralumno.java*
- *clienteborrarcurso.java*
- *clienteborrargrupo.java*
- *clientecalificacion.java*
- *clientecurso.java*
- *clienteescala.java*
- *clienteevaluacion.java*
- *clientegrupo.java*
- *clienteinstrumento.java*

Esto se hará únicamente en el caso de que el servidor se encuentre en una máquina remota. En caso contrario, si el servidor está alojado en la misma máquina donde se ubica el cliente, entonces no se cambiará ninguna línea de estos archivos. Una vez configurados los clientes, estos serán capaz de solicitar recursos a través de los servicios Web ubicados en la máquina servidor, con lo cual, podrán realizar operaciones para la gestión de calificaciones de cursos.

6. Pruebas sobre el SEA

En la figura 4 se muestra una prueba de funcionalidad del servicio Web que devuelve la información de un curso. En la imagen, el navegador invoca al recurso `Curso,` que está ubicado en el servidor. El navegador obtiene como respuesta un documento XML con la información de los cursos existentes en el sistema.

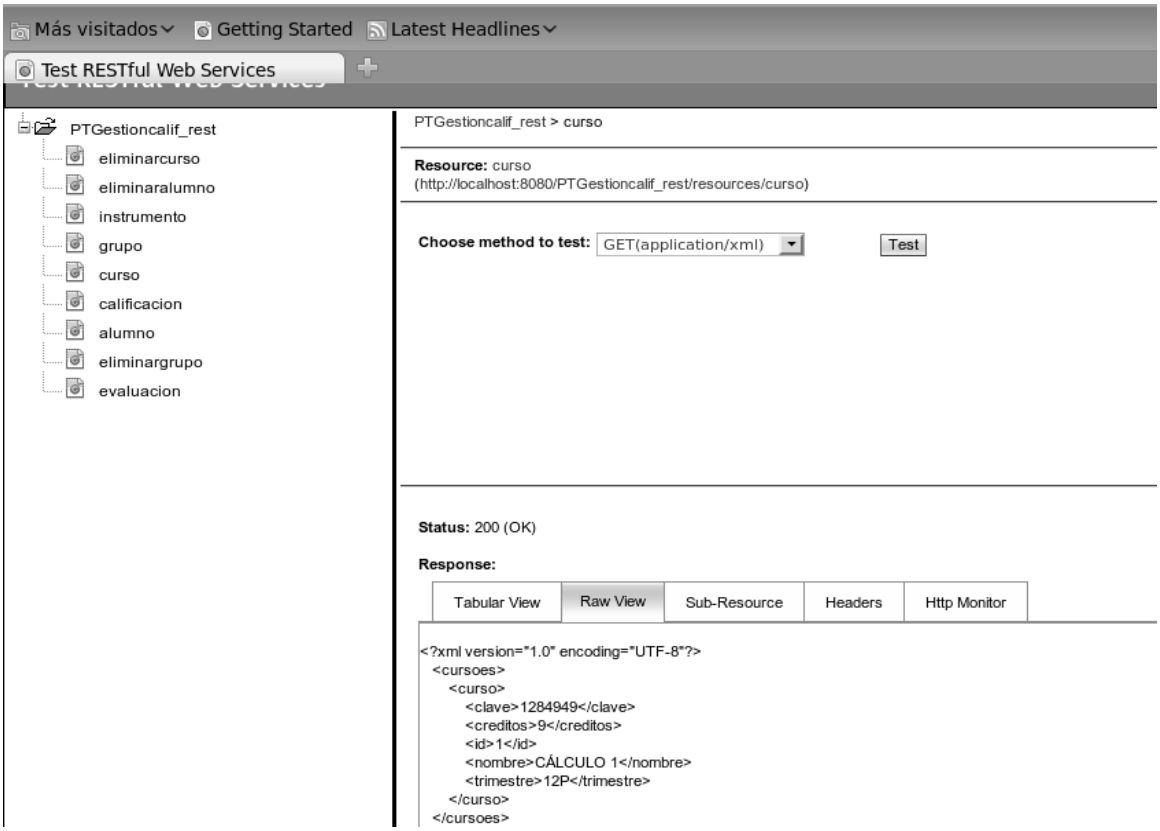


Figura 4. Servicio Web que devuelve un objeto de tipo curso

No solo se comprobó el funcionamiento unitario de los servicios Web. También se probó su funcionamiento en conjunto, es decir se hicieron pruebas con operaciones que involucraban 2 ó más servicios Web. Por ejemplo, el proceso para dar de alta a un alumno es el siguiente:

1. Dar de alta un curso en la base de datos del sistema
2. Dar de alta un grupo de alumnos en la base de datos del sistema
3. Dar de alta un alumno

De esta manera se puede observar que para dar de alta un alumno, se requiere el correcto funcionamiento de varios servicios.

Una vez que se probó el buen funcionamiento de cada uno de los servicios, se comenzó a construir una aplicación cliente, la cual consumiría todos los servicios Web necesarios para llevar a cabo la gestión de calificaciones correctamente. La aplicación cliente se construyó utilizando la biblioteca gráfica swing.

En la figura 5 se muestra la ventana cursos de la aplicación cliente. En ella se lleva a cabo la gestión de los cursos que va a impartir un profesor. Al dar clic en el botón ver cursos la aplicación cliente invoca un servicio Web, el cual le devolverá como respuesta (en un archivo XML) la lista de cursos existentes en la base de datos del sistema. Al obtener la lista de cursos por parte del servidor, la aplicación cliente muestra el resultado obtenido en el apartado Lista de cursos. Si el usuario elige la opción Modificar entonces la aplicación cliente invocará otro servicio Web, el cual le permitirá actualizar el registro de un curso en la base de datos.

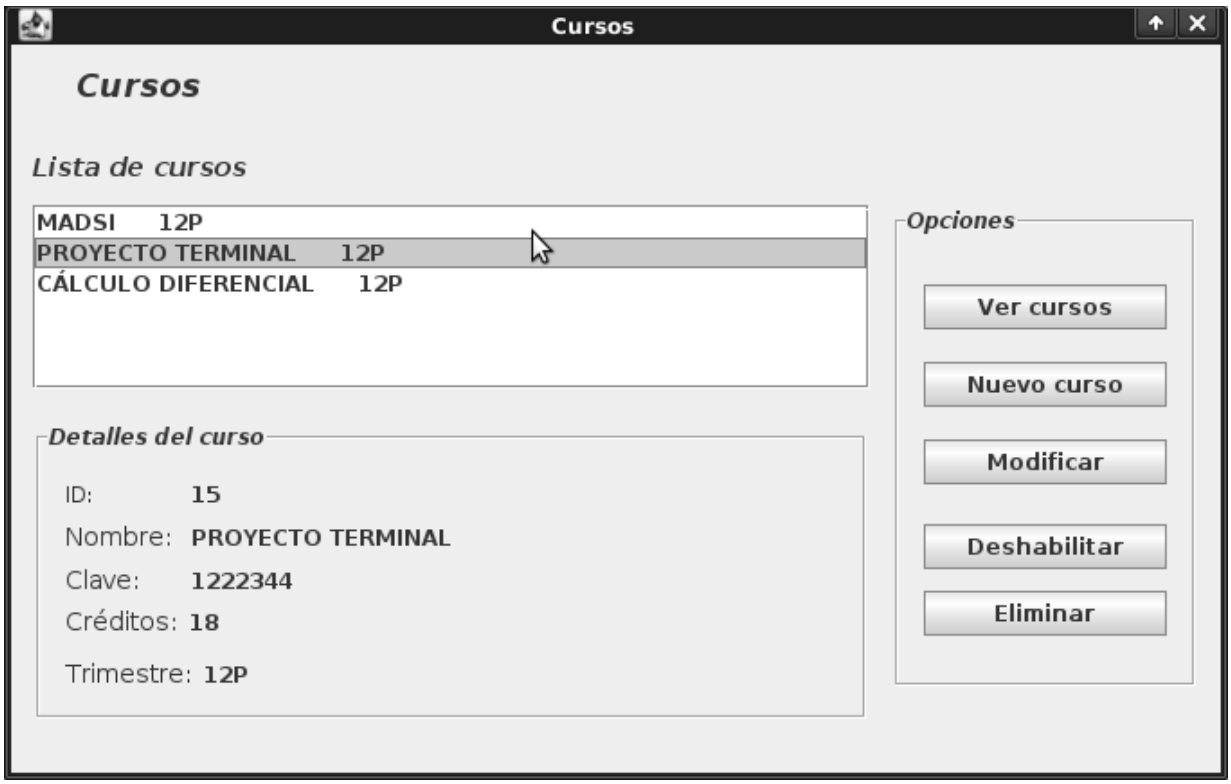


Figura 5. Ventana cursos

En la figura 6 se muestra la ventana *promedios* de la aplicación cliente, mediante la cual, es posible obtener los promedios finales de los alumnos de un grupo. Esta ventana cuenta con opciones para guardar un archivo de texto con los promedios finales y para visualizar gráficas con la distribución de las calificaciones finales.

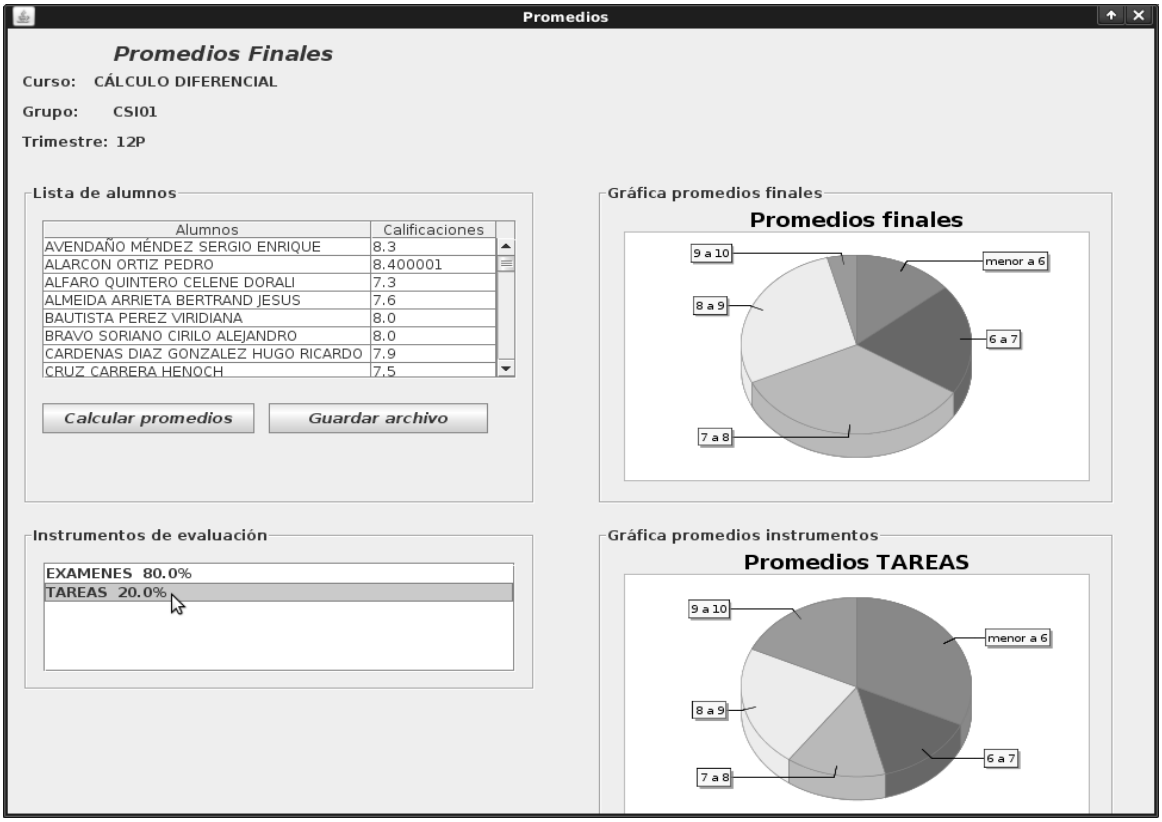


Figura 6. Ventana promedios

Desde la aplicación cliente que se implementó se pueden gestionar cursos, grupos de alumnos, alumnos, instrumentos de evaluación, evaluaciones, calificaciones y promedios, con lo cual se cumple el objetivo de gestionar las calificaciones de los cursos que un profesor puede llegar a impartir.

Para que la aplicación cliente pueda trabajar correctamente, es necesario que los servicios Web ubicados en el servidor, estén implementados correctamente en GlassFish, de otra manera la aplicación cliente no podrá solicitar recurso y por ende no podrá realizar ninguna operación de la gestión de calificaciones de cursos.

7. Problemas durante el desarrollo

Se registraron varios problemas durante el desarrollo de los servicios Web y de la aplicación cliente. La mayoría de los problemas tuvo que ver con la manera en que tanto los servicios Web

como la aplicación cliente intercambian mensajes. En ocasiones, los servicios Web no podían procesar una solicitud de la aplicación cliente debido a que el cliente no enviaba los mensajes en el formato correcto, En otra ocasión hubo problemas porque el cliente no podía establecer contacto con el servidor.

Otro problema durante el desarrollo del proyecto fue el modelado de la base de datos. Se gastó mucho tiempo en encontrar un modelo de base de datos que pudiera manejar adecuadamente los datos de la gestión de calificaciones. Al cabo de varias semanas, y de varias pruebas se encontró un modelo que en primera instancia, manejó bien la información de alumnos, cursos, grupos, instrumentos de evaluación, calificaciones y evaluaciones.

Algunas operaciones tomaban mucho tiempo en realizarse como por ejemplo, el cálculo de los promedios finales. Primero se pensó que podía ser por el mal estado de la red local donde fue implementado el proyecto, pero después se supo que el problema estaba en como se recuperaba la información de la base de datos. Se encontró que se hacían operaciones de más, lo cual retrasaba el cálculo de calificaciones finales. Para solucionar el problema se tuvieron que revisar a fondo las operaciones de consulta en la capa de datos para encontrar donde se hacían operaciones innecesarias.

Universidad Autónoma Metropolitana Unidad Azcapotzalco

División de Ciencias Básicas e Ingeniería

Licenciatura en Ingeniería en Computación

Gestión de calificaciones de cursos mediante servicios Web

Reporte Final

Elaboró:
Avendaño Méndez Sergio Enrique
205304956

Trimestre: 12-O

Fecha:
10 de Diciembre de 2012

Asesores:
Dra. Beatriz Adriana González Beltrán

Dra. Lizbeth Gallardo López

Índice

1. Introducción

El objetivo principal del proyecto Gestión de calificaciones de cursos mediante servicios Web, es el desarrollo de servicios Web que permitan a un profesor llevar el control adecuado de las calificaciones de sus alumnos en los diferentes cursos que pueda impartir. Dichos servicios, podrán ser invocados desde una aplicación cliente (como puede ser un navegador, una aplicación para un teléfono móvil o una aplicación para una PC), lo cuál, permitirá al usuario (profesor) llevar a cabo la gestión de calificaciones a través de cualquiera de estos medios.

Los servicios Web deben proporcionar al usuario las funcionalidades necesarias para definir y modificar una modalidad de evaluación. Una modalidad de evaluación puede contener uno o varios instrumentos de evaluación (Exámenes, tareas, proyectos, etc), que a su vez, pueden estar sometidos a algunos criterios definidos por el profesor (peso de la calificación final).

También, se deben proporcionar funciones para la gestión de alumnos y cursos impartidos, para que de esta manera, el usuario pueda registrar en el sistema la información de sus cursos que va a impartir, información de los grupos de alumnos e información de los alumnos que tomarán un curso.

Por último, los servicios Web deben proporcionar al profesor las funciones necesarias para el registro de calificaciones de los alumnos que están asociados a un curso, de acuerdo a los instrumentos y criterios de evaluación definidos previamente. En base a las calificaciones obtenidas por los alumnos, se debe proporcionar una función que calcule los promedios finales de los alumnos del curso.

Se prevé el desarrollo del Sistema de Evaluación de Alumnos (SEA) el cual hará uso de servicios Web para proporcionar las funcionalidades descritas anteriormente. La implementación de estos Servicios Web estarán basados en la arquitectura REST[1].

1 Transferencia de Estado Representacional (REST - Representational State Tranfer). Define un conjunto de principios para diseñar servicios Web haciendo foco en los recursos del sistema, incluyendo como se accede al estado de dichos recursos y como se transfieren hacia los clientes.

2 Objetivos

2.1 Objetivo general

Diseñar e implementar servicios Web que permitan gestionar las calificaciones para los cursos impartidos por un profesor.

2.2 Objetivos específicos

- Diseñar e implementar los servicios Web que permitan administrar los grupos de alumnos.
- Diseñar e implementar los servicios Web que permitan administrar las materias.
- Diseñar e implementar los servicios Web que permitan definir las modalidades de evaluación.
- Diseñar e implementar los servicios Web que permitan registrar las calificaciones de los alumnos.
- Diseñar e implementar los servicios Web que permitan obtener los grupos de un profesor.
- Diseñar e implementar los servicios Web que permitan obtener las materias registradas por un profesor.
- Diseñar e implementar los servicios Web que permitan obtener las modalidades de evaluación de un grupo.
- Diseñar e implementar los servicios Web que permitan obtener las calificaciones de los alumnos

3 Antecedentes

Existen actualmente proyectos dentro y fuera de la universidad que tienen relación con la gestión de calificaciones de alumnos. A continuación se describen:

3.1 Referencias internas

Sistema de gestión de calificaciones para los cursos impartidos por un profesor [1] . En este proyecto, se busca diseñar un sistema que le permita a un profesor administrar las calificaciones de sus alumnos en los diferentes cursos que imparte. Es una aplicación de escritorio, por lo que no se hace uso de servicios Web ni de recursos en red para llevar a cabo sus funciones.

Aplicación Android para la sincronización de las tareas y el material didáctico de sistemas MOODLE[2] [2] . Este proyecto consiste en desarrollar una aplicación móvil en Android para MOODLE, que permita gestionar los cursos de los alumnos. Esto es, permitirá la inscripción a los cursos, la virtualización de material didáctico y las tareas de cada curso a los que está inscrito un alumno, así como la posibilidad de subir las tareas a sistemas MOODLE desde un *Smartphone*. Aunque en este proyecto se proporcionan algunas funcionalidades para la gestión de grupos y alumnos en un sistema MOODLE no se proporciona, por ejemplo, una funcionalidad para llevar a cabo el control de las calificaciones de los alumnos.

3.2 Referencias externas

MOODLE [3] . Es un sistema de Gestión de cursos de Código Abierto (*Open Source Course Management System*), conocido también como Sistema de Gestión del Aprendizaje (*Learning Management System,* LMS) o como Entorno de Aprendizaje Virtual (*Virtual Learning Environtment*, VLE). Es una aplicación Web gratuita que los educadores pueden utilizar para crear sus sitios de aprendizaje efectivo en línea. MOODLE cuenta con un libro de calificaciones que permiten hacer cualquier operación con las calificaciones obtenidas por los alumnos. Además, se pueden crear escalas personales de calificación.

2 Entorno Modular de Aprendizaje Dinámico Orientado a Objetos (MOODLE – Module Object Oriented Dynamic Learning Environment). Es un sistema de gestión de cursos de distribución libre, que ayuda a los educadores a crear comunidades de aprendizaje en línea.

Atutor [4] . Es un Sistema de Gestión de Aprendizaje de código abierto basado en la Web y diseñado con el objetivo de lograr accesibilidad y adaptabilidad. Los administradores pueden instalar o actualizar ATutor en minutos. Los educadores pueden rápidamente ensamblar, empaquetar y redistribuir contenido educativo, además de llevar a cabo sus clases online. Los estudiantes pueden aprender en un entorno de aprendizaje adaptativo. La versión 1.6.1 de Atutor cuenta con un libro de calificaciones (*Gradebook*) que permite calificar exámenes y tareas. Además, se pueden generar informes con las calificaciones.

Claroline [5]. Es un proyecto de software libre que se distribuye con licencia GNU/GPL. Está escrito en lenguaje de programación PHP, utiliza MySQL como gestor de bases de datos. Está disponible para plataformas y navegadores libres, y plataformas y navegadores propietarios. Entre sus características están el administrar foros de discusión, administrar listas de enlaces, crear grupos de estudiantes, estructurar una agenda con tareas, gestionar los envíos de los estudiantes (documentos, tareas, trabajados, etc) entre otras características. Claroline permite crear grupos de estudiantes para llevar un control de los ejercicios y tareas que se desarrollan durante el curso, sin embargo, no ofrece herramientas para llevar un control sobre las calificaciones.

4. Justificación del proyecto

Los servicios Web son muy prácticos, ya que pueden aportar una gran independencia entre la aplicación que usa el servicio Web (aplicación cliente) y el propio servicio. La aplicación cliente podrá obtener de los servicios Web las funciones necesarias para llevar a cabo la gestión, es decir, obtendrán las funciones para crear un grupo de alumnos, para dar de alta alumnos, para crear una modalidad de evaluación, para calificar alumnos, etc.

Una ventaja importante del uso de servicios Web es que pueden reutilizarse y ser invocados desde cualquier tipo de aplicación cliente, desarrollada en cualquier lenguaje de programación y sobre cualquier plataforma.

Para llevar a cabo la gestión de calificaciones de los alumnos, la implementación de los servicios Web será basado en REST, por razones de rendimiento, ubicuidad y simplicidad, ya

que REST está basado en HTTP[3]. Ésto es una gran ventaja para la implementación de cliente ligeros, pues solo existen 2 condiciones: 1) Tener acceso remoto mediante HTTP y 2) Un analizador básico de XML[4]. Ésto a su vez, permite que la gran mayoría de los lenguajes modernos puedan tener acceso a cualquier servicio Web que se ofrece a través de internet.

5. Arquitectura de los sistemas basados en REST

5.1 Definición de REST

REST define un conjunto de principios arquitectónicos para diseñar servicios Web poniendo énfasis en el uso eficiente de los recursos. Un recurso puede ser una entidad o conjunto de entidades que puede ser accedido por diversos clientes. Una entidad representa un concepto de la lógica del negocio de la aplicación. REST determina la forma de consultar y modificar el estado de un recurso[7] ; así como de establecer como se transfieren los recursos vía HTTP hacia los clientes.

5.2 Principios de REST

Una implementación concreta de un servicio Web REST sigue los siguientes principios de diseño:

- No se publican servicios RPC[5], en REST se publican recursos
- Utiliza los métodos HTTP de manera explícita
- Un protocolo cliente/servidor sin estado
- Cada recurso posee un identificador único y global
- Transfiere XML, JavaScript Object Notation (JSON), o ambos

3 Protocolo de Transferencia de Hipertexto (HTTP – *Hypertext Transfer Protocol*). Define la sintaxis y la semántica que utilizan los elementos de software de la arquitectura Web para comunicarse. Es un protocolo orientado a transacciones, siguiendo el esquema petición-respuesta entre un cliente y un servidor.

4 Lenguaje de Marcado Extensible (XML Extensible Markup Language). Es un estándar para el intercambio de información estructurada entre diferentes plataformas. Puede utilizarse en bases de datos, editores de texto, hojas de cálculo, etc.

5 Llamada a Procedimiento Remoto (RPC. Remote Procedure Call). Son utilizadas dentro del paradigma cliente-servidor. Siendo el cliente el que solicita una función y el servidor devuelve el resultado de la función al cliente.

A continuación se explican a detalle estos principios.

5.3 No se publican servicios RPC, lo que se publican son recursos

En arquitecturas como SOAP[6] se define un archivo WSDL[7] que contiene las firmas de los métodos a los cuales puede acceder un cliente. Este archivo informa al cliente sobre los métodos disponibles; de tal manera que un cliente pueda realizar una llamada a un método remoto (RPC) con el propósito de obtener un resultado específico. En la arquitectura REST, los servicios Web no publican un conjunto de métodos; lo que se publica son recursos. Por ejemplo, en REST no se puede publicar una interfaz `IGestionEmpleados` con métodos: `agregarEmpleado`, `borrarEmpleado` o `buscarEmpleado`. Un recurso para REST sería solamente `EmpleadosDeLaEmpresa` o `Empleado número 12`.

5.4 Utilizar métodos HTTP de manera explícita

Una de las características claves de los servicios Web REST es el uso explícito de los métodos HTTP. Por ejemplo, HTTP GET se define como un método que obtiene un recurso para las aplicaciones cliente. Las operaciones del protocolo HTTP son:

- GET para obtener un recurso del servidor
- POST para crear un recurso en el servidor
- PUT para cambiar el estado de un recurso o actualizarlo
- DELETE para eliminar un recurso

5.5 Un protocolo HTTP cliente/servidor sin estado

En la arquitectura REST, cada mensaje HTTP contiene la información necesaria para comprender la petición; como resultado, ni el cliente ni el servidor necesitan recordar ningún estado de las comunicaciones entre los mensajes. Sin embargo, en otras arquitecturas basadas en HTTP (como SOAP) utilizan cookies para mantener el estado de la sesión (en REST no está permitida esta práctica).

6 SOAP (siglas de *Simple Object Access Protocol*) es un protocolo de intercambio de información basado en XML. Es una herramienta fundamental en los servicios Web que permite intercambio de documentos o llamadas a procedimientos remotos (RPC).

7 WSDL (siglas de *Web Services Description Language*) se trata de un dialecto basado en XML que se utiliza para describir servicios Web. Describe la forma de comunicación, es decir, los requisitos que debe cumplir el cliente para interactuar con los servicios Web.

5.6 Cada recurso posee un identificador único y global

Cada recurso posee un identificador único y global que lo distingue de cualquier otro recurso, inclusive cuando ambos tuvieran los mismos datos. En el caso de `Empleado número 12`, este sería diferente de `Empleado número 20` aunque tuvieran el mismo nombre, sueldo, etc.

5.7 REST debe transferir XML, JSON o ambos

Un cliente puede conocer el estado de un recurso; es decir, los valores que guardan sus atributos a través de un archivo en formato XML o JSON[8] .En la implementación del recurso se decide que información será visible para el exterior, y el tipo de archivo que guardará dicho estado. Una representación de `Empleado número 12` podría ser un documento en formato XML con la información accesible de este.

```
<empleado>
<clave>12</clave>
<nombre>Armando Mendoza</nombre>
<sueldo>12000</sueldo>
</empleado>
```

6. Metodología de base empleada en el proyecto

Se utilizó el Proceso Unificado como metodología de base para el desarrollo del Sistema de Evaluación de Alumnos (SEA). El proceso unificado es un marco de desarrollo de software, el cual se caracteriza por estar centrado en la arquitectura del sistema; por estar dirigido a través de casos de uso, y por llevar el desarrollo de forma iterativa e incremental.

Este proceso está compuesto por fases, las cuales se dividen a su vez en un serie de iteraciones. Por cada iteración, se siguió una serie de actividades del proceso de desarrollo. Dichas actividades fueron: Modelo de negocio, Análisis de requerimientos, Diseño, Implementación y

8 JSON *(JavaScript Object Notation)* es un formato ligero para el intercambio de datos. JSON es un subconjunto de la notación literal de objetos de JavaScript que no requiere el uso de XML. La simplicidad de JSON ha dado lugar a la generalización de su uso, especialmente como alternativa a XML en AJAX.

Pruebas.

Cada iteración realizada durante el proyecto, significó un incremento del producto desarrollado, lo cual añadió mejoras a las funcionalidades del sistema. En cada iteración, se tomó una parte de los casos de uso y escenarios de uso, con los cuales se desarrolló todo el camino a través de las distintas actividades mencionadas anteriormente.

En la iteración de inicio de este proyecto las actividades que tuvieron mayor relevancia fueron: el modelo de negocio, el análisis de requerimientos y el diseño. Describir el modelo de negocio fue importante para conocer el proceso que siguen los profesores para evaluar a sus alumnos y para gestionar las calificaciones. En el análisis de requerimientos se determinó que funciones debía hacer el sistema y con que fin, para ello se elaboró el documento de escenarios de uso en el que se pudieron comprender los requisitos del sistema; así como, la interacción con el usuario. En el diseño se propuso una solución para satisfacer los requerimientos del sistema; para ello, se elaboraron los diagramas de clases, diagramas de secuencia, diagramas de robustez, diagrama de implantación y un diagrama de la base de datos.

En la primera iteración se implementaron los módulos relacionados con el caso de uso `Administración de cursos`. Tomando como referencia los escenarios de uso: `alta de curso, baja de curso y modificar un curso`; además se implementaron los servicios Web necesarios para registrar un curso en la base de datos del sistema, para modificar un registro de un curso y para eliminarlo.

Con la experiencia obtenida en la primera iteración, fue posible implementar el resto de los casos de uso en iteraciones subsiguientes, con lo cual se fue incrementando y mejorando el sistema gestión de calificaciones de cursos.

7. Desarrollo del proyecto

En esta sección se aborda todo el proceso necesario para construir el SEA. La primera parte de esta sección “Diseño” contiene los artefactos que fueron utilizados para modelar el comportamiento del sistema de manera detallada. La segunda parte “Implementación” describe las herramientas utilizadas para desarrollar el sistema, así como los pasos que se siguieron para programar el sistema.

7.1 Diseño

En esta sección se presentarán los artefactos de diseño que se utilizaron para modelar el comportamiento del Sistema de Evaluación de Alumnos (SEA) . Dichos artefactos son:

- Diagrama de casos de uso general
- Escenarios de uso
- Diagrama de clases
- Diagrama de la base de datos
- Diagrama de la arquitectura del sistema

7.1.1 Diagrama de casos de uso general

Del documento de requerimientos se pudo determinar al actor principal (el cual será el profesor) y a los objetivos, con los cuáles el actor busca producir un resultado de valor agregado. El diagrama de casos de uso se muestra en la figura 1.

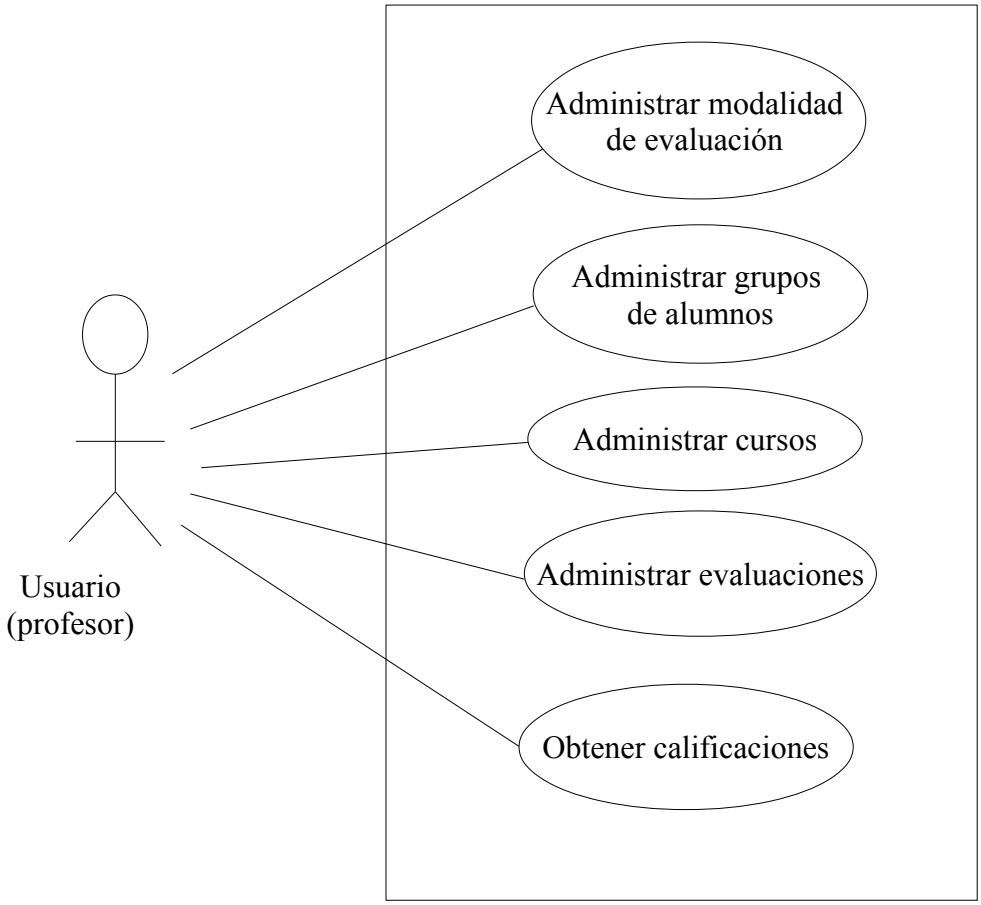


Figura 1. Diagrama de casos de uso

Administrar modalidades de evaluación

Una modalidad de evaluación puede estar constituida por uno o varios instrumentos de evaluación, y éstos a su vez, pueden estar sometidos a varios criterios (como el peso que tiene cada instrumento de evaluación en la calificación final de los alumnos). Una modalidad de evaluación podría componerse de la siguiente manera: Un profesor define como instrumentos de evaluación: Tareas, Exámenes y Programas. El profesor define como criterio que las Tareas tengan un peso del 20% de la calificación final, que los Exámenes tengan un peso del 50% y que los programas tengan un 30%.

Este caso de uso, se traduce en tres escenarios de uso, los cuales son:

- Crear modalidad de evaluación: Consiste en definir los instrumentos de evaluación junto con el porcentaje de la calificación final que el profesor utilizará para evaluar a sus alumnos. Una vez definida la modalidad de evaluación, ésta deberá asociarse al curso correspondiente.
- Modificar modalidad de evaluación: Consiste en modificar el nombre o peso de los instrumentos de evaluación definidos previamente.
- Eliminar modalidad de evaluación: Consiste en eliminar uno o varios instrumentos de evaluación del sistema.

Administrar grupos de alumnos

La administración de grupos de alumnos se traduce en 6 escenarios de uso, los cuáles se listan a continuación:

- Alta de grupo de alumnos: Consiste en dar de alta en la base de datos del sistema un grupo de alumnos con todos sus datos (clave de grupo, cupo, salón).
- Baja de grupo de alumnos: Consiste en dar de baja un grupo de alumnos de la base de datos, solo en caso de que no se tenga asociado: alumnos, instrumentos de evaluación y evaluaciones.
- Modificar grupo de alumnos: Consiste en modificar la información de un grupo ya existente en el sistema.
- Alta de alumno: Consiste en dar de alta un alumno asociado con un grupo en la base de datos. Los datos a almacenar son (nombre(s), apellido paterno, apellido materno, matrícula y correo electrónico). En este escenario, el usuario contará con 2 opciones: 1) Dar de alta a un alumno manualmente: Con esta opción, el usuario podrá escribir

todos los datos del alumno para darlo de alta en el sistema. 2) Dar de alta la lista de alumnos: En esta opción, el usuario tendrá la posibilidad de elegir una lista de alumnos, para que de esta manera, el sistema pueda dar de alta a todos los alumnos que se encuentren en la lista.

- Baja de alumno: Consiste en dar de baja del sistema a un alumno perteneciente a algún grupo.
- Modificar información del alumno: Consiste en modificar la información de un alumno ya existente en el sistema.

Administrar cursos

Un curso puede estar asociado a uno o más grupos de alumnos según sea el caso. La administración de cursos, se traduce en los siguientes escenarios de uso:

- Alta de curso: Consiste en dar de alta en la base de datos del sistema, toda la información relacionada con un curso que vaya a impartir el profesor. Los datos a registrar son: Nombre del curso, clave del curso, créditos y trimestre.
- Modificar curso: Este escenario consiste en modificar la información del curso, previamente registrada en la base de datos.
- Baja de curso. Consiste en eliminar de la base de datos un curso creado previamente, siempre y cuando no tenga asociado un grupo de alumnos.

Administrar evaluaciones

Un profesor, podrá evaluar a sus alumnos de acuerdo a los instrumentos y criterios de evaluación que se hayan definido previamente en el sistema.

La administración de evaluaciones consta de los siguientes escenarios de uso:

- Calificar alumnos: Este escenario de uso consiste en calificar a los alumnos pertenecientes a un grupo en base a un instrumento de evaluación definido previamente
- Corregir calificaciones: Esta operación consiste en modificar alguna calificación previamente registrada.
- Obtener promedios de un instrumento: Este escenario de uso consiste en obtener el promedio de los alumnos correspondiente a un instrumento de evaluación definido en el sistema.
- Obtener promedio total: Este escenario de uso consiste en obtener el promedio total

para cada alumno considerando todos los instrumentos de evaluación definidos al inicio de curso.

Los servicios Web deberán proporcionar a la aplicación cliente todas las funciones necesarias para llevar a cabo todas las actividades de cada uno de los casos de uso mencionados.

En el apéndice A se muestra como ejemplo el escenario de uso "Obtener promedio total"; para mayores detalles de los escenarios de uso remitirse al documento de diseño.

7.1.2 Diagrama de clases

Para obtener las entidades clave del SEA, se aplicó la técnica de “extracción de sustantivos” sobre el documento de requerimientos y sobre el documento de escenarios de uso. Con esta técnica se pudo obtener una lista de sustantivos candidatos, de la cual fue necesario eliminar aquellas entidades que no resultaron ser clave. La mayoría de las candidatas a entidad se eliminó de la lista, debido a que resultaron ser atributos de otra entidad.

De esta manera, quedaron solo 7 entidades clave:

- Curso
- Escala
- Instrumento
- Evaluación
- Grupo
- Alumno
- Calificación

Una vez elegidas las entidades clave, se definieron sus atributos, responsabilidades y colaboraciones. Para ello se utilizó la técnica de Diseño de clases empleando tarjetas CRC.

Entidad Curso

La entidad `curso` se refiere a la UEA que va a impartir el profesor. Los atributos de `curso` son: ID, nombre del curso, clave del curso, créditos del curso y trimestre. Se determinó una colaboración entre las entidades `Curso` y `Grupo.`

Entidad Escala

La entidad `Escala` permite pasar de una nota cuantitativa a una nota cualitativa basada en las letras: NA, S, B, MB. Un profesor podría de terminar su escala como se muestra en la figura 2.

Rangos de calificación	Letra
x>=0 y x<6	NA
x>=6 y x<7.5	S
x>=7.5 y x<9	B
x>=9 y x<=10	MB

Tabla 1. Ejemplo de escala determinada por un profesor al inicio de un curso

Cuenta con los atributos ID, Letra, límite_inferior y límite_superior. En los límites inferior y superior se determinará el rango correspondiente a una letra. La entidad `Escala` colaborará con la entidad `Grupo`.

Entidad Instrumento

La entidad `instrumento` se refiere a los aspectos tales como tareas, exámenes, proyectos, etc. que el profesor determina para evaluar a los estudiantes de un curso en particular. Se definieron únicamente los atributos: ID, nombre del instrumento y peso del instrumento. El peso del instrumento representará el porcentaje de la calificación final que tiene dicho instrumento. Se definieron colaboraciones con la entidad `Grupo` y la entidad `Evaluación`, debido a que un profesor define para un grupo evaluaciones, a partir de uno o más instrumentos de evaluación.

Entidad Evaluación

La entidad `Evaluación` corresponde a la especificación de un instrumento particular; por ejemplo, para el instrumento de exámenes el profesor puede especificar 3 parciales; para el instrumento de tareas puede especificar 5 tareas diferentes a los largo de curso. Para esta entidad, se definieron los siguientes atributos: ID, nombre de la evaluación y descripción de la evaluación. Se determinó que la entidad `Evaluación` colabora directamente con la entidad `Instrumento` (como ya se explicó anteriormente) y con la entidad `Calificación`.

Entidad Grupo

La entidad `Grupo` se refiere al conjunto de alumnos inscritos a un cursos particular. Se definieron los atributos: ID, clave de grupo, cupo, salón, alumnos inscritos y horario. Esta entidad va a colaborar con otras tres entidades. Colaborará con la entidad `Curso`, con la entidad `Alumno` y con la entidad `Instrumento`.

Entidad Alumno

La entidad `Alumno` guarda los datos personales de un alumno inscrito a un grupo particular; y cuenta con los atributos: ID, nombre de alumno, matrícula, email y calificación final. La entidad colaborará con la entidad `Grupo` y con la entidad `Calificación.`

Entidad Calificación

La entidad `Calificación` asocia a la entidad `Alumno` con la entidad `Evaluación`, definiendo así una calificación de un alumno para una evaluación particular. Sus atributos son ID y calif.

El diagrama de clases se presenta en la figura 2

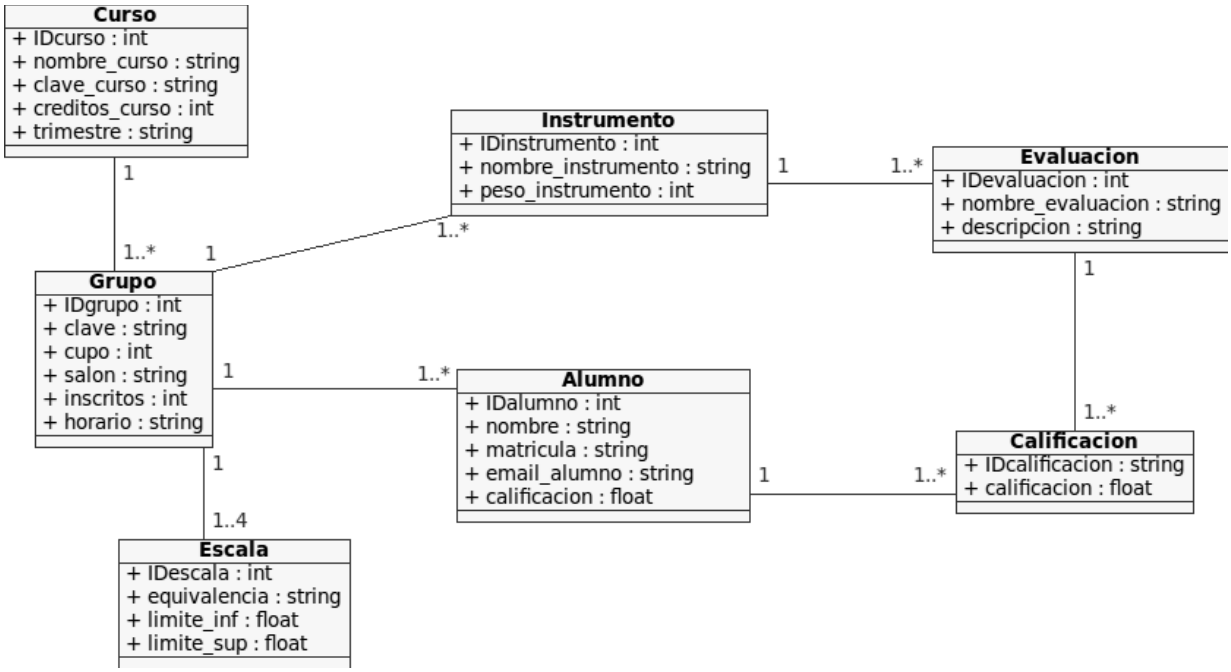


Figura 2. Diagrama de clases

Todas las clases cuentan con métodos set y métodos get, no incluidos en el diagrama, con los cuáles se pueden guardar y recuperar los atributos de cada una de los ejemplares de la clase. De esta forma las clases pasan a funcionar como clases Java Bean[9].

Multiplicidad de las clases

Un objeto del tipo `Curso` puede estar asociado a uno o más grupos de alumnos. Por ejemplo, un profesor podría impartir el curso de `Cálculo Diferencial` a dos grupos de alumnos en diferentes horarios.

Un `Grupo` está formado por uno o n alumnos; y ese grupo pertenece a un solo `Curso`. Un profesor estará interesado en asociar uno o más instrumentos de evaluación para un grupo. Es decir, el profesor podría decidir evaluar a los alumnos de un grupo con un único instrumento

9 Los JavaBeans son un modelo de componentes creado por Sun Microsystems para la construcción de aplicaciones en Java. Es una clase Java que cumple con ciertas normas con los nombres de sus atributos y operaciones. Un JavaBean tiene declarados sus atributos como privados con métodos setter y getter para cada uno de ellos.

`Exámenes` y en otro grupo podría utilizar los instrumentos `Tareas, Exámenes y Participaciones.`

De un `Instrumento` se pueden desprender una o varias `Evaluaciones` según lo decida el profesor. Es decir, el profesor podría definir un instrumento `Exámenes` con las siguientes Evaluaciones `Examen Parcial 1, Examen Parcial 2,` etc.

Un objeto de tipo `Alumno` puede estar asociado a una o más calificaciones, dependiendo de la `Evaluación` que el profesor especificó al inicio del curso.

7.1.3 Estructura de la base de datos

La base de datos del SEA es de tipo relacional, tomando como referencia el diagrama de clases. En primer lugar, se transformó cada entidad del modelo de clases en una tabla; después, se definió una llave principal para cada tabla; por último se realizó la normalización de la base de datos. En la figura 3 se muestra el diagrama de la base de datos.

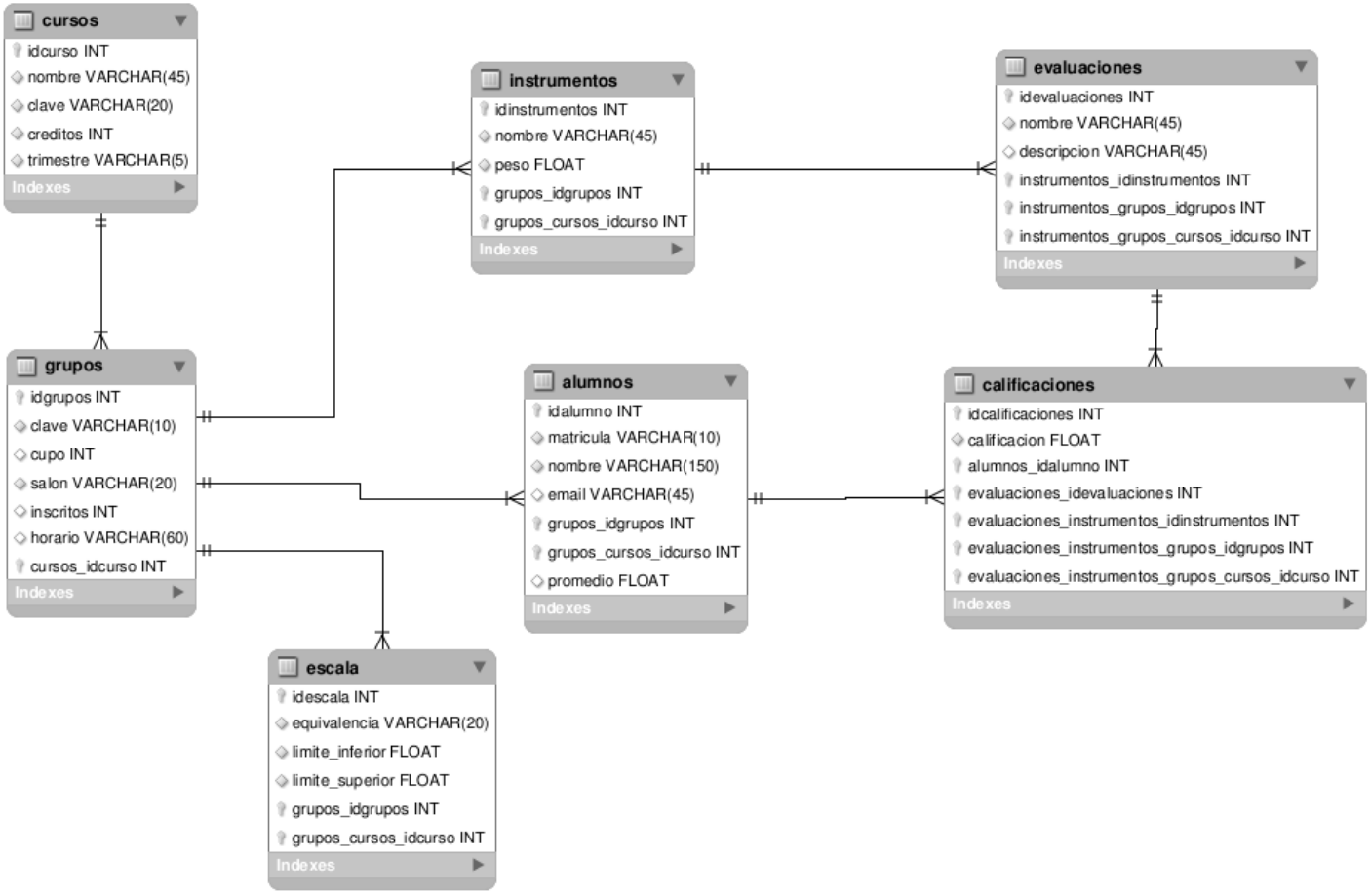


Figura 3. Estructura de la base de datos

Tabla Cursos

Almacena los datos correspondientes a los cursos que el profesor va a impartir.

Columnas:

- idcursos: llave primaria de la tabla, tipo int, autoincrement
- nombre: tipo varchar
- clave: tipo varchar
- créditos: tipo int
- trimestre: tipo varchar

Tabla Grupos

Almacenan los datos correspondientes a los grupos de alumnos. Cuenta con una llave foránea "cursos_idcurso" con la cual se establece una relación con la tabla `cursos`.

Columnas:

- idgrupos: llave primaria de la tabla, tipo int, autoincrement
- clave: tipo varchar
- cupo: tipo int
- salon: tipo varchar
- inscritos: tipo int
- horario: tipo varchar
- cursos_idcurso: llave foránea, tipo int

Tabla instrumentos

Almacena los registros correspondientes a los instrumentos de evaluación que el profesor va a definir para evaluar a un grupo. Cuenta con 2 llaves foráneas para establecer referencias a las tablas `grupos y cursos.`

Columnas:

- idinstrumentos: llave primaria de la tabla, tipo int, autoincrement
- nombre: tipo varchar
- peso: tipo float
- grupos_idgrupos: llave foránea, tipo int
- grupos_cursos_idcurso: llave foránea, tipo int

Tabla escalas

Esta tabla almacena los registros correspondientes a las escalas de calificación para un grupo de alumnos. Cuenta con 2 llaves foráneas para establecer referencias con registros de las tablas

`Grupos` y `Cursos`.

Columnas:

- idescala: llave primaria de la tabla, tipo int, autoincrement
- equivalencia: tipo varchar
- limite_inf: tipo float
- limite_sup: tipo float
- grupos_idgrupos: llave foránea, tipo int
- grupos_cursos_idcurso: llave foránea, tipo int

Tabla alumnos

Almacenan todos los registros correspondientes a los alumnos de un determinado grupo. Cuenta con 2 llaves foráneas para establecer referencias con registros de las tablas `grupos` y `cursos`.

Columnas:

- idalumnos: llave primaria de la tabla, tipo int, autoincrement
- matricula: tipo varchar
- nombre: tipo varchar
- email: tipo varchar
- grupos_idgrupos: llave foránea, tipo int
- grupos_cursos_idcurso: llave foránea, tipo int
- promedio: tipo float

Tabla evaluaciones

Guarda todos los registros correspondientes a las evaluaciones con las que el profesor calificará a sus alumnos. Se tienen en esta tabla 3 llaves foráneas para establecer referencias con registros de las tablas `instrumentos`, `grupos` y `cursos`.

Columnas:

- idevaluaciones: Llave primaria de la tabla, tipo int, autoincrement
- nombre: tipo varchar
- descripcion: tipo varchar
- instrumentos_idinstrumentos: llave foránea, tipo int
- instrumentos_grupos_idgrupos: llave foránea, tipo int
- instrumentos_grupos_cursos_idcurso: llave foránea, tipo int

Tabla calificaciones

Contiene todos los registros correspondientes a las calificaciones emitidas, de acuerdo a los instrumentos y evaluaciones definidos con anterioridad. Se tienen 5 llaves foráneas para establecer referencias con las tablas `alumnos`, `instrumentos`, `evaluaciones`, `cursos`, y `grupos`.

Columnas:

- idcalificaciones: Llave primaria de la tabla, tipo int, autoincrement
- calificacion: tipo float
- alumnos_idalumno: llave foránea, tipo int
- evaluaciones_idevaluaciones: llave foránea, tipo int
- evaluaciones_instrumentos_idinstrumentos: llave foránea, tipo int
- evaluaciones_instrumentos_grupos_idgrupos: llave foránea, tipo int
- evaluaciones_instrumentos_grupos_cursos_idcurso: llave foránea, tipo int

7.1.4 Arquitectura del SEA

El sistema está basado en una arquitectura n-layer. Esta es un arquitectura de cliente ligero, donde el software se puede distribuir en varias máquinas. En la figura 4 se muestra el diagrama de implantación que representa como se distribuyen los componentes del sistema en el hardware.

Capa de datos

Esta capa se encarga de almacenar los datos de las entidades de nuestro sistema en la Base de datos; así como de acceder a ellos, a través de consultas, para manipularlos a través de las operaciones definidas en el sistema.

Capa de aplicación

En esta capa residen los recursos [10] y servicios Web que los clientes van a utilizar para realizar sus actividades. Dependiendo de la petición del usuario, a través de la aplicación cliente, los servicios Web acceden a los recursos y los envía al cliente para responder así a la petición. Esta capa también se comunica con la capa de datos, para almacenar o recuperar datos.

Capa de presentación

También recibe el nombre de estación cliente; y es la capa con la cual interactúa el usuario de la aplicación SEA. Esta capa captura los datos proporcionados por el usuario en un proceso mínimo y luego los envía a la capa de aplicación. En un proceso inverso, esta capa recibe los datos proporcionados por la capa de aplicación y los envía hacia la capa de presentación.

10 Recordemos que un recurso puede ser un objeto entidad o conjunto de objetos entidad, el cual puede ser accedido por diversos clientes. Un objeto entidad representa un concepto de la lógica del negocio de la aplicación

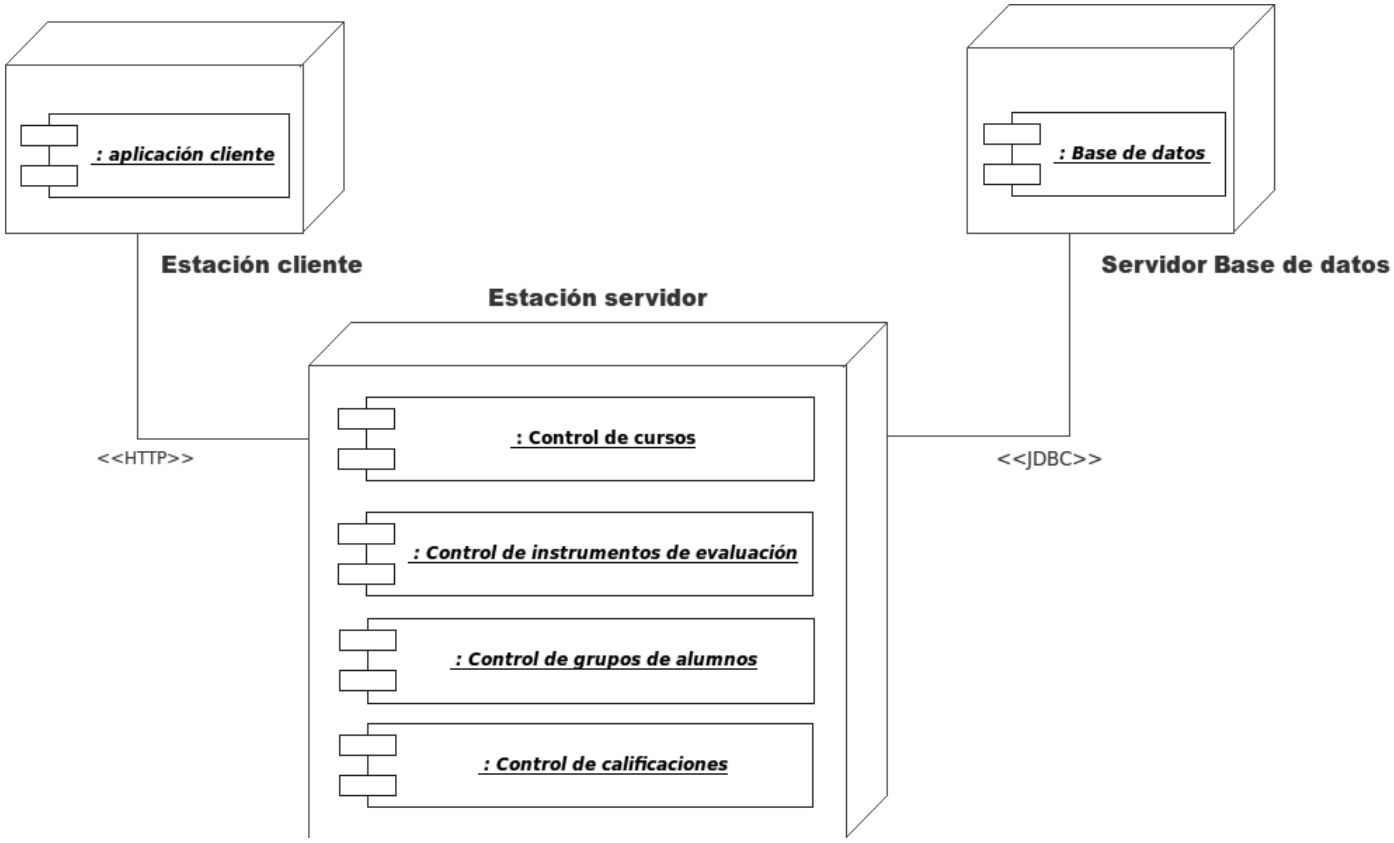


Figura 4. Diagrama de implantación

7.1.5 Arquitectura REST

Para el desarrollo del SEA, se aplicaron los principios de la arquitectura REST de la siguiente manera:

- Se utilizó el protocolo sin estado HTTP para la comunicación entre la aplicación cliente y el servidor.
- Fueron publicados en la estación servidor los siguientes recursos para ser utilizados por la estación cliente: Curso, Grupo, Alumno, Escala, Evaluación, Calificación, Instrumento, Lista de cursos, Lista de grupos, Lista de alumnos, Lista de evaluaciones y Lista de calificaciones, Lista de instrumentos.
- Todos los recursos que residen en la estación servidor tienen un identificador único y global (URI) el cual es utilizado por la estación cliente para acceder a ellos.

- Para trabajar sobre los recursos existentes en el servidor, la estación cliente debe hacer uso de los métodos HTTP adecuados. Los métodos HTTP definidos para trabajar sobre cada recurso son los siguientes:

Recurso	**Métodos HTTP definidos para trabajar sobre ellos**
Curso	GET, POST, PUT, DELETE
Grupo	GET, POST, PUT, DELETE
Alumno	GET, POST, PUT, DELETE
Instrumento	GET, POST, PUT
Evaluación	GET, POST, PUT
Calificación	GET, POST, PUT
Escala	GET, POST, PUT
Lista de cursos	GET
Lista de grupos	GET
Lista de alumnos	GET
Lista de evaluaciones	GET
Lista de calificaciones	GET
Lista de instrumentos	GET

- Para el intercambio de información entre el cliente y el servidor, se utilizaron archivos XML. Si la aplicación cliente solicita al servidor un recurso `Grupo,` entonces el servidor envía al cliente un documento XML con la información del grupo solicitado.

7.2 Implementación

7.2.1 Tecnología empleada: software y hardware

Software

El software utilizado para el desarrollo del proyecto es libre, por lo que no se requirieron licencias. A continuación se describe ese software:

NetBeans IDE

Es una herramienta para escribir, compilar, depurar y ejecutar programas. Está escrito en Java[8] , sin embargo, puede servir para programar en cualquier lenguaje de programación. Hubo dos razones importantes para elegir este IDE; en primer lugar, porque es un producto de libre distribución, por lo cual no hay restricciones en su uso. En segundo lugar, porque contiene una gran cantidad de componentes de software, con los cuales, se pueden construir aplicaciones Web. La versión utilizada para desarrollar el proyecto fue la 6.9.1.

Servidor de aplicaciones GlassFish

Es un servidor de aplicaciones que implementa la plataforma Java EE6, por lo que soporta las últimas versiones de tecnologías como: JSP, JSF, Servlets, EJBs, Java API para servicios Web (JAX-WS), entre otras tecnologías[9] . La versión de GlassFish utilizada para desarrollar el proyecto fue la versión 3.0.1, debido a que para el proyecto, se utilizaron algunas características de Java EE6[11].

Jersey

Jersey es una implementación propuesta por Sun para JAX-RS[8] (API para servicios Web basados en REST). Se utilizó Jersey para desarrollar servicios Web que respondan al protocolo HTTP. Con esta API[12] se logró determinar que clases actuarán como servicios Web. La versión utilizada de Jersey para este proyecto fue la 1.1.2.

Hibernate

Es una herramienta de Mapeo-Objeto-Relacional (ORM) para la plataforma java, el cual facilita el mapeo de los atributos de una base de datos relacional hacia el modelo de objetos de una aplicación; utilizando para el mapeo, archivos de configuración XML. Hibernate es también un software de libre distribución. Se utilizó esta herramienta para el mapeo de las tablas de la base de datos a objetos de Java y visversa, lo cual permitió reducir el tiempo de desarrollo porque no es necesario escribir las consultas SQL; de lo contrario, la obtención y

11 Java Plataform Enterprise Edition. Es una plataforma de programación para desarrollar y ejecutar software de aplicaciones en el lenguaje de programación Java. Permite utilizar arquitecturas de N capas distribuidas y se apoya en componentes de software modulares ejecutándose sobre un servidor de aplicaciones.

12 Interfaz de programación de aplicaciones (API – *Application Programming Interface*). Conjunto de funciones o métodos que exponen las clases de alguna biblioteca para ser utilizado por otro software.

guardado de los datos en la base de datos habría tenido que realizarse manualmente, empleando SQL y JDBC[11] . La versión de hibernate utilizada para este proyecto fue la 3.3.1.

Manejador de base de datos MySQL

MySQL es un sistema de gestión de bases de datos relacionales, multihilo y multiusuario. Es ideal para entornos donde la lectura de datos puede llegar a ser intensiva. MySQL es ofrecido bajo la licencia GNU GLP para cualquier uso compatible con esta licencia [10]. La versión utilizada para el desarrollo del proyecto fue la 5.1.63.

MySQL Workbench

MySQL Workbench es una herramienta visual de diseño de bases de datos que integra: diseño, creación y gestión de bases de datos MySQL. Para usar esta herramienta, es necesario tener instalado previamente, el manejador de base de datos MySQL. La versión utilizada para este proyecto fue la 5.2.35.

Lenguaje de programación

El código de la aplicación cliente como de los servicios Web, fue implementado con el lenguaje de programación orientado a objetos Java.

Sistema operativo

Todas las tecnologías mencionadas anteriormente, fueron montadas en una computadora personal con Sistema Operativo Linux Ubuntu en la versión 10.04 LTS.

Hardware

Para el desarrollo del proyecto fue utilizada una computadora personal con los siguientes recursos:

- Memoria RAM de 1 Gb
- Disco duro de 120 Gb
- Procesador Intel(R) Celeron (R) a 1.74 Ghz
- Sistema Operativo Ubuntu 10.04 LTS

7.2.2 Funcionalidad del sistema

Hasta el momento en este documento solo se han abordado temas de diseño; se mostraron, entre otras cosas, los casos de uso del Sistema de Evaluación de Alumnos, el diagrama de clases y la descripción de las entidades, el modelo de la base de datos con la descripción de las tablas y el diagrama de la arquitectura. A continuación se mostrará una de las funcionalidades del sistema, indicando cómo interactúan tanto la aplicación cliente como el servidor, el cual proporciona los recursos necesarios para la gestión de calificaciones. Para explicar la funcionalidad del sistema se mostrará una pequeña parte de la interfaz gráfica del sistema junto con el código encargado de llevar a cabo las funciones.

Funcionalidad: Alta curso

Con esta funcionalidad, el profesor podrá guardar en el SEA los datos correspondientes a un curso que va a impartir. Los datos a registrar son: Nombre del curso, clave del curso, créditos y trimestre. A continuación se indica el procedimiento necesario para registrar un curso. La aplicación cliente muestra la pantalla de inicio (figura 5) con las 3 opciones principales.

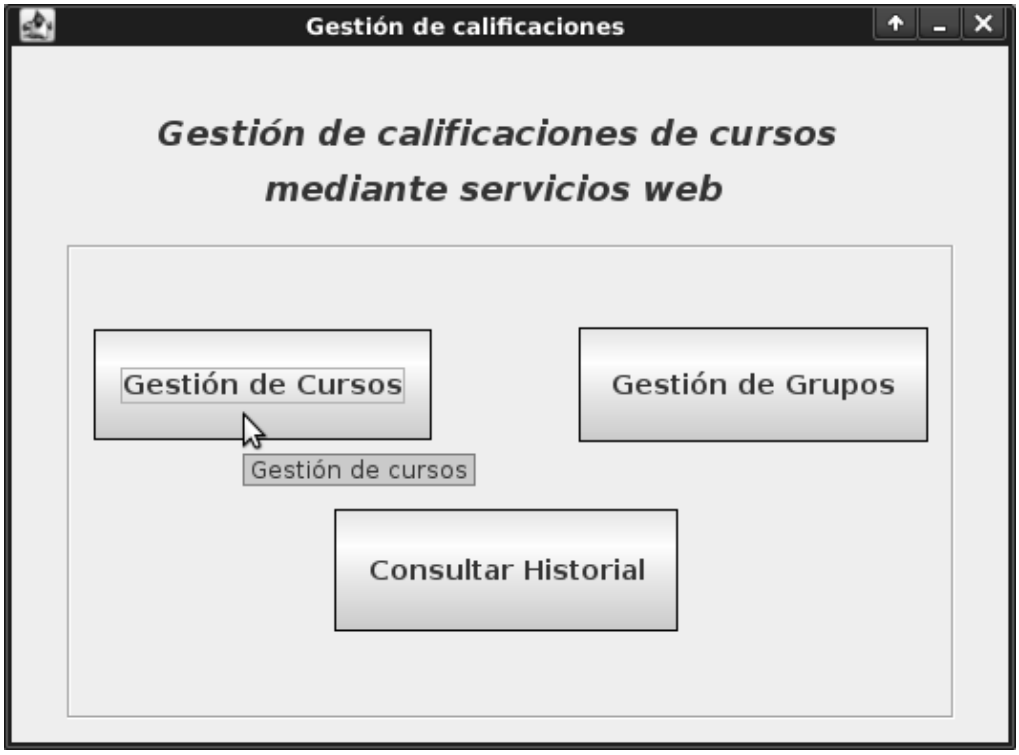


Figura 5. Pantalla principal

Al elegir la opción “Gestión de cursos”, la aplicación abre una nueva ventana donde se encuentran las siguientes opciones para gestionar los cursos: nuevo curso, modificar curso, eliminar curso, activar curso y desactivar curso, ver figura 6

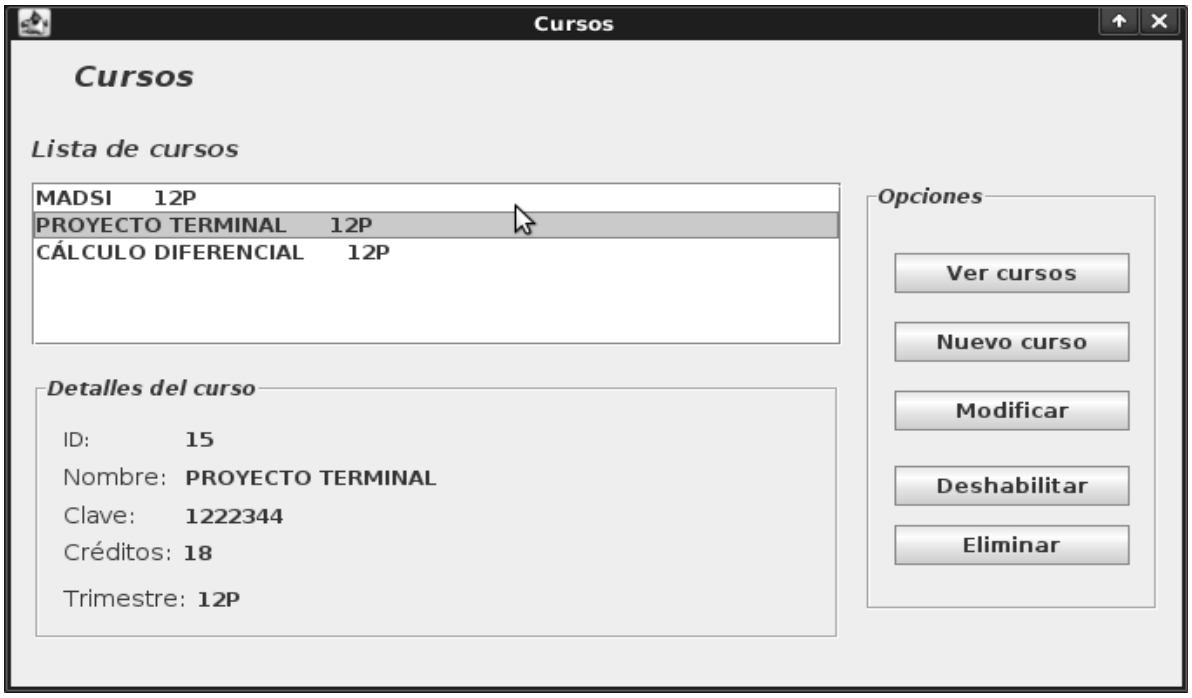


Figura 6. Gestión de cursos

Al elegir la opción Nuevo curso, la aplicación mostrará un formulario (ver figura 6) con los campos necesarios para capturar la información del curso en el sistema.

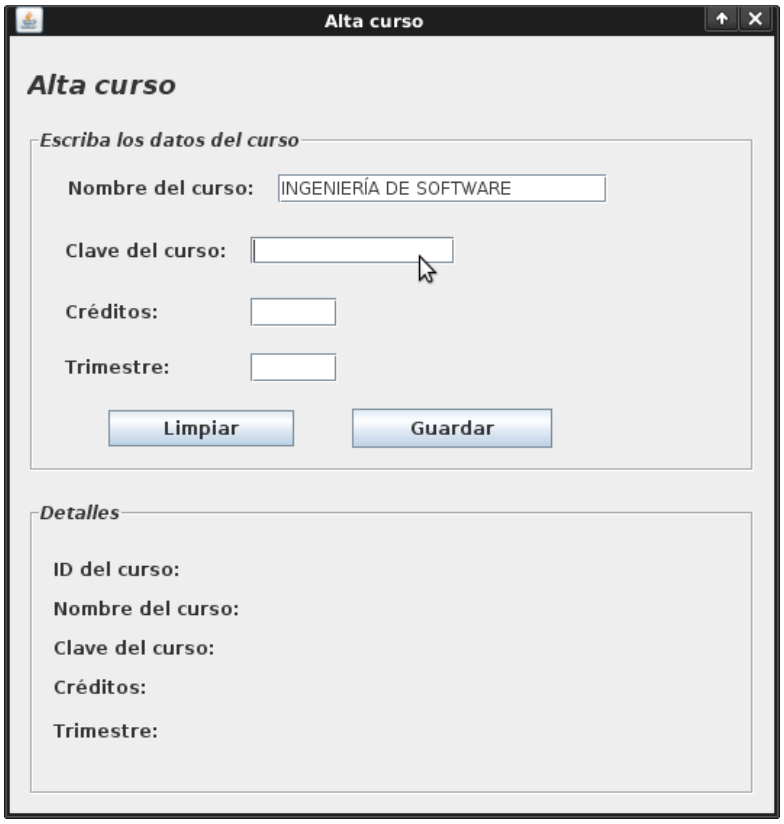


Figura 7. Formulario alta curso

El proceso de crear un curso es la base para posteriormente dar de alta alumnos, instrumentos de evaluación, evaluaciones, etc. las cuales permitirán llevar un control de las calificaciones de los alumnos. Con la información obtenida del formulario mostrado en la figura 6, se crea un objeto de la clase Curso. Una vez que se han capturado en los datos del curso, la aplicación cliente hará uso de un servicio Web, el cual proporcionará el recurso necesario para dar de alta un curso. Para solicitar este recurso al servidor, la aplicación cliente emplea el método de la clase alta curso, el cual se muestra en el código 1.

```
/*Esta función se dispara al momento de dar clic en el botón guardar del formulario mostrado en  la figura 6*/

private void guardarActionPerformed(java.awt.event.ActionEvent evt) {

    /*Antes de guardar un objeto curso en el sistema, se aplica una prueba para asegurarse de que los datos ingresados en los campos del formulario son correctos*/
    if(prueba()==true){

        /*Si los datos de los campos contienen resultados válidos, entonces se procede a guardar en el sistema la información del curso*/

        Curso c = new Curso();

        try{

/*Obtengo los valores capturados por el usuario para crear un objeto Curso, el cual será enviado al servidor*/

            c.setnombre(nombre.getText());
            c.setclave(clave.getText());
            c.setcreditos(Integer.parseInt(creditos.getText()));
            c.settrimestre(trimestre.getText());

            /*Se crea un objeto de la clase clientecurso, el cual contiene la función necesaria para enviar al servidor la información del curso. Esta clase es generada por el IDE Netbeans y su código se explicará mas adelante*/

            clientecurso cliente = new clientecurso();
            c = cliente.guardar(c,Curso.class);
            nombre.setText("");
            clave.setText("");
            creditos.setText("");
            trimestre.setText("");

   /*Se coloca la información del curso creado en las etiquetas destinadas para ello*/
```

```
            idlab.setText(String.valueOf(c.getid()));
            nombrelab.setText(c.getnombre());
            clavelab.setText(c.getclave());
            creditoslab.setText(String.valueOf(c.getcreditos()));
            trimestrelab.setText(c.gettrimestre());

        JOptionPane.showMessageDialog(this, "Información del curso registrada correctamente",
"Aviso", 1, null);

        }catch(Exception e){
            JOptionPane.showMessageDialog(this, "Hubo un error al crear el curso", "Error", 0, null);
        }
}
```

Código 1. Dar de alta un curso

A continuación se muestra un fragmento de la clase clientecurso (generada por el IDE Netbeans), la cual contiene la URL del recurso necesario para crear un `curso`.

```
public class clientecurso {
   private WebResource webResource;
   private Client client;

/*Aquí se especifica la URL del recurso que va a utilizarse para crear el curso
(http://localhost:8080/PTGestioncalif_rest/resources/cursos ). Este recurso debe estar habilitado
en el servidor */

   private static final String BASE_URI = "http://localhost:8080/PTGestioncalif_rest/resources/";


   public clientecurso() {
 com.sun.jersey.api.client.config.ClientConfig config = new
          com.sun.jersey.api.client.config.DefaultClientConfig();
      client = Client.create(config);
```

```
        webResource = client.resource(BASE_URI).path("cursos");
    }

/*El siguiente método (guardar) recibe como parámetro el objeto curso que va a ser registrado en el sistema. Además, se indica que el método HTTP a utilizarse para guardar el curso será POST*/

 public <T> T guardar(Object requestEntity, Class<T> responseType) throws
UniformInterfaceException {
        returnwebResource.type(javax.ws.rs.core.MediaType.APPLICATION_XML).post(responseType,
requestEntity);

}
```

Código 2. Clase cliente curso

Es importante mencionar que un objeto ubicado en el servidor es el que manejará el recurso que la aplicación cliente necesita para dar de alta un curso. La clase a la que pertenece este objeto tiene definidos 4 métodos los cuales, son ejecutados por los 4 métodos HTTP disponibles para REST: GET, POST, DELETE y PUT[6] .

Cada uno de estos métodos determinará la acción que hará REST sobre la aplicación.

1. GET: Para obtener un valor o un listado de objetos
2. POST Para guardar un nuevo objeto en la aplicación
3. DELETE: Para eliminar un objeto
4. PUT Para actualizar un objeto

La clase que proporciona el recurso para dar de alta cursos se llama `admincurso.` Esta clase está ubicada en el servidor y su código se describe a continuación.

```
/*Se utiliza @path para indicar la ruta desde la que será accedido el recurso via web*/
@Path("/cursos")
public class admincurso {

/*Se declara un objeto de tipo curso, el cual será enviado por la aplicación cliente*/
   curso c;

/*El siguiente método "guardar" indica que se empleará el método HTTP POST para guardar la
información del curso en el servidor*/

@POST
/*La siguiente línea indica que el formato de los mensajes entre cliente y servidor será en XML*/
   @Consumes({"application/xml"})
   public Response guardar(curso c){
      this.c = c;

      cursodao cudao = new cursodao();
      this.c.setid(cudao.guardacurso(this.c));
      return Response.ok(this.c).build();
   }
```

Código 3. Clase `Admin Curso`

Como en este caso vamos a guardar un nuevo objeto de tipo `Curso`, se invocará desde la aplicación cliente al método POST de la clase `admincurso`. El método POST que se localiza en esta clase recibe como parámetro un objeto de tipo curso. Este objeto proviene de la aplicación cliente, creado cuando el usuario terminó de llenar el formulario `alta curso`. Posteriormente se crea un objeto de tipo `cursodao, el` cual tiene como propósito el realizar operaciones sobre la base de datos del SEA.

```
/*método guardacurso de la clase cursodao*/
public Integer guardacurso(curso c) throws HibernateException
   {
      Integer id = 0;
      try
      {
         iniciaOperacion();
         id = (Integer) sesion.save(c);
         tx.commit();
      } catch (HibernateException he)
      {
         manejaExcepcion(he);
         throw he;
      }
      finally
      {
         sesion.close();
      }
      return id;
   }
```

Código 4. Guardar curso

Al utilizar el método `guardarcurso` de la clase `cursodao` los datos del curso quedan registrados en la base de datos. Una vez que el método POST ha terminado de guardar el curso en la base de datos, devuelve como respuesta a la aplicación cliente el ID del curso recién creado. Si el ID es recibido en el cliente sin ningún problema significa que la operación se llevó a cabo correctamente, por lo tanto los datos del curso ya estarán registrados en el sistema para su posterior uso en otras operaciones.

8. Conclusiones y perspectivas del proyecto

El proyecto que presentamos en este documento Gestión de calificaciones mediante servicios Web, tuvo como objetivo el desarrollo del SEA (Sistema de Evaluación de Alumnos), el cual proporciona a los profesores las funciones que permiten llevar el control de las calificaciones de los alumnos inscritos a un curso. El sistema organiza la información necesaria para

gestionar las calificaciones (cursos, grupos de alumnos, evaluaciones, instrumentos de evaluación, etc) y la presenta al usuario de tal manera que le permite crear grupos de alumnos, cursos, dar de alta alumnos, calificar alumnos, obtener promedios, entre otras cosas. Este sistema funciona mediante servicios Web basados en la arquitectura REST; el usuario trabaja desde una aplicación cliente la cual solicita a un servidor remoto los recursos que se necesitan para llevar a cabo la gestión de calificaciones.

Seguimos el Proceso Unificado, el cual marcó las fases de todo el desarrollo: 1) determinación de los requerimientos funcionales y no funcionales, a partir de los cuales se plantearon el documento de visión y el documento de requerimientos; luego se elaboraron los casos de uso y sus escenarios posibles; esto forma parte del documento de diseño. A partir del diseño se especificaron los módulos del sistema; posteriormente, se implementaron los módulos; y finalmente, se programaron los servicios Web para los módulos, a saber: gestión de alumnos, gestión de cursos, gestión de grupos, gestión de instrumentos, gestión de evaluaciones y gestión de calificaciones.

A este proyecto se le pueden agregar otras funcionalidades; por ejemplo, crear un cliente que permita, a un profesor, operar la aplicación SEA en teléfonos móviles. Este cliente invocaría los mismos servicios que el cliente de escritorio, permitiendo que el profesor realice la gestión de calificaciones desde su móvil. También se podrían implementar algunas funcionalidades de uso exclusivo para alumnos, tales como: consulta de kardex, consulta de calificaciones de una UEA, consulta de horarios para el trimestre, entre otras.

La tabla 1 muestra un resumen de los objetivos planteados inicialmente y su grado de alcance con una breve descripción.

	Objetivos particulares	**Alcance**
☑	Diseñar e implementar los servicios Web que permitan administrar los grupos de alumnos.	Se modelaron e implementaron los servicios Web para administrar los grupos de alumnos, a saber: dar de alta, modificar, eliminar y recuperar grupos de alumnos en el sistema desde la aplicación cliente.
☑	Diseñar e implementar los servicios Web que permitan administrar las materias	Se modelaron e implementaron los servicios Web para administrar las materias, a saber: dar de alta, dar de baja, modificar y obtener una materia (o curso) desde la base de datos y llevar esta información hacia

		la aplicación cliente.
	Diseñar e implementar los servicios Web que permitan definir las modalidades de evaluación	Se modelaron e implementaron los servicios Web para administrar las modalidades de evaluación, a saber: dar de alta, dar de baja, modificar y eliminar sobre los instrumentos de evaluación en el sistema desde la aplicación cliente.
	Diseñar e implementar los servicios Web que permitan registrar las calificaciones de los alumnos	Se modelaron e implementaron los servicios Web para administrar las calificaciones de alumnos, a saber: registrar y modificar calificaciones en el sistema desde la aplicación cliente.
	Diseñar e implementar los servicios Web que permitan obtener los grupos de un profesor	Se modelaron e implementaron los servicios Web para recuperar del servidor, la información de los grupos, materias y modalidades de evaluación de los alumnos pertenecientes a un profesor; quien solicita esta información a través de un cliente
	Diseñar e implementar los servicios Web que permitan obtener las materias registradas por un profesor	
	Diseñar e implementar los servicios Web que permitan obtener las modalidades de evaluación de un grupo	
	Diseñar e implementar los servicios Web que permitan obtener las calificaciones de los alumnos	La aplicación cliente fue capaz de recuperar toda la información de las calificaciones de los alumnos de un determinado curso. Además, desde la aplicación cliente se lograron implementar funcionalidades adicionales, como son: mostrar gráficas, guardar archivos, y calcular promedios.

Tabla 2. Resumen de los objetivos y su alcance

 – Objetivo cumplido.

 – Objetivo cumplido, con funcionalidades adicionales.

Bibliografía

[1] I. Y. Olmos Aquino, “Sistema de gestión de calificaciones para los cursos impartidos por un profesor”, proyecto terminal, División de CBI, Universidad Autónoma Metropolitana Azcapotzalco, México, 2010.

[2] M. D. Cruz Rodríguez, “Aplicación Android para la sincronización de las tareas y el material didáctico de sistemas MOODLE”, proyecto terminal, División de CBI, Universidad Autónoma Metropolitana Azcapotzalco, México, 2011.

[3] Comunidad de MOODLE (2011, Noviembre), Acerca de MOODLE [En línea], Disponible: http://docs.moodle.org/19/es/Acerca_de_Moodle

[4] G.. Gay (2011, Noviembre), ATutor Learning Management Tools [En línea], Disponible: http://atutor.ca/

[5] Consortium Claroline (2011, Noviembre), About Claroline [En línea], Disponible: http://www.claroline.net/about-claroline.html

[6] D.E. Silva Límaco (2012, Diciembre), Apuntes de java: Restful[En línea], Disponible: http://www.apuntesdejava.com/search/label/restful

[7] D. Gomez (2012, Diciembre), Principios de Rest [En línea], Disponible: http://www.dosideas.com/noticias/java/332-principios-de-rest.html

[8] G. Jiménez Centeno (2012, Diciembre), Jersey: la implemetación de RESTFull de Sun [En línea], Disponible: http://www.apuntesdejava.com/search/label/restful
[8] NetBeans Community (2012, Diciembre), Portal del IDE Java de Código Abierto

[En línea], Disponible: http://netbeans.org/index_es.html

[9] GlassfishCommunity (2012, Diciembre), GlassFish Community [En línea], Disponible: http://glassfish.java.net/es/

[10] M. Contreras (2012, Diciembre), Mysql en Español [En línea], Disponible: http://mysql-espanol.org/about/

[11] D. M. Palao (2012, Diciembre), Tutorial Básico de Hibernate [En línea], Disponible: http://www.davidmarco.es/tutoriales/hibernate-reference/

Apéndice A

Escenario de uso: Obtener promedio total

Descripción:
Éste escenario de uso consiste en obtener el promedio total de un alumno, en base a todos los instrumentos de evaluación definidos por el profesor al inicio del curso.

Actores y sus intereses:
El actor principal (profesor) desea conocer los promedios totales correspondientes a los alumnos de algún curso.

Disparador:
El escenario de uso da inicio, cuando el actor elige la opción “Obtener promedio total”.

Pre-condición:
Para que ésta operación se lleve a cabo con éxito, el profesor debe haber realizado evaluaciones previamente.

Post-condición:
Al finalizar ésta operación, el actor obtendrá el promedio total correspondiente a cada uno de los alumnos de un grupo.

Flujo principal:

1. El actor elige la opción “Obtener promedio total”.
2. El sistema muestra la lista de cursos existentes.
3. El actor elige el curso, del cual, desea saber el promedio.
4. El sistema muestra la lista de grupos de alumnos correspondientes al curso.
5. El actor elige el grupo de alumnos, del cual, desea saber el promedio.
6. El sistema muestra la lista de alumnos que pertenecen al grupo junto con los promedios totales correspondientes.
7. Si el actor desea obtener más promedios, el proceso se repite desde el paso 2.