RECUPERACIÓN DE INFORMACIÓN Y RECOMENDACIONES EN LA WEB

AÑO 2015

(GRUPO 10)

Andrés Beraldo	CI: 4.716.823-3
Ignacio Decia	CI: 4.492.244-0
Lucia Labat	CI: 4.549.399-1
Manuela Viola	CI: 4.749.107-0

Índice

Introducción

Como estudiantes de Ingeniería en Computación, nos encontramos de cerca con el problema que implica comprender el plan de estudios de nuestra carrera, particularmente el sistema de previaturas. Ésto se debe a que el sitio web de bedelías presenta la información de forma poco clara y distribuida por múltiples lugares. Dada esta realidad, surge como motivación realizar una aplicación que no solo facilite al estudiante la comprensión de dicha información, sino que también la brinde de forma más personalizada y consolidada.

Propuesta

La propuesta es crear un sistema interactivo donde el estudiante de computación indique los cursos que tiene aprobados y/o exonerados y muestre los cursos que está habilitado a cursar. Además mostrará información sobre los cursos y créditos totales acumulados por el estudiante.

Sistema de previas

El sistema de previas de la carrera de Ingeniería en Computación consiste en por un lado los Cursos y por otro lado los Grupos. Un Grupo es un conjunto de Cursos que tiene un puntaje mínimo y máximo. Cada curso aporta un puntaje en el grupo según si está aprobado o exonerado.
Un curso puede tener como previa otro curso o un grupo. Para que el grupo se considere como aprobado se debe llegar al puntaje mínimo requerido por ese grupo.

La aplicación

La aplicación modela el estado de los cursos de la siguiente manera:
Rojo: habilitado a cursar
Amarillo: curso aprobado
Verde: examen aprobado

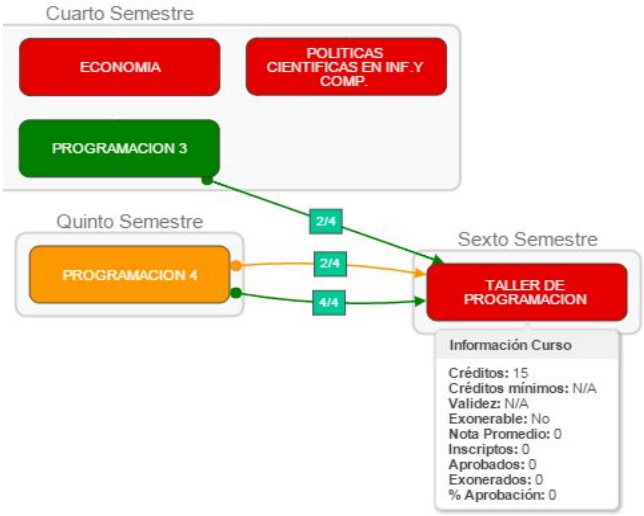
Los cursos son encerrados en un recuadro más grande que indica a qué semestre pertenecen.

Inicialmente la aplicación muestra únicamente aquellos cursos que no tienen previas y que están habilitados (Rojo) para cursarlos. Al hacer un primer clic sobre uno de los cursos, el estado pasa a aprobado (Amarillo). Inmediatamente se muestran en pantalla aquellos cursos que tenían como previa el curso recientemente aprobado y que ahora pasan a estar habilitados (Rojo). Al hacer un segundo clic se pasa del estado curso aprobado a examen aprobado (Verde). A medida que el estudiante vaya indicando los estados de los cursos la aplicación irá mostrando los cursos que esté habilitado a cursar hasta que se construya todo el árbol de la carrera.

Además la aplicación da la posibilidad de ver información sobre los cursos. Con un clic derecho sobre un curso se muestra un tooltip que contiene los siguientes datos:

Créditos: créditos que aporta el curso
Créditos mínimos: créditos mínimos necesarios para poder cursar
Validez: validez del curso en meses
Exonerable: indica si el curso es exonerable
Nota promedio: el promedio de calificaciones obtenidas por los estudiantes
Cantidad de inscriptos: el total de inscriptos
Aprobados: cantidad de estudiantes aprobados

Exonerados: cantidad de estudiantes exonerados
% aprobación: porcentaje de aprobación del curso



Además de la información el sistema muestra las aristas con las previas del curso en cuestión.

Existen dos tipos de aristas:

- Amarillas: indica previa de tipo curso
- Verde: indica previa de tipo examen

Aquellas previas cuyas aristas contengan cuadrados de un mismo color significa que pertenecen a un grupo, siendo el denominador el mínimo de puntos para aprobar el grupo y en el numerador la cantidad de puntos que aporta la previa a ese grupo.

Por ejemplo, la imágen anterior indica que para cursar Taller de programación es necesario cumplir con el grupo correspondiente a las aristas con cuadrados en verde. El mínimo del grupo es de 4 puntos y existen dos formas de alcanzar ese mínimo:

Examen de Programación 3 (arista Verde) -> 2 pts
Curso de Programación 4 (arista Amarilla) -> 2 pts

Examen de Programación 4 (arista Verde) -> 4 pts

Además el sistema da la posibilidad de ocultar todos los cursos de un semestre simplemente con hacer clic derecho en el recuadro del mismo.

Casos interesantes

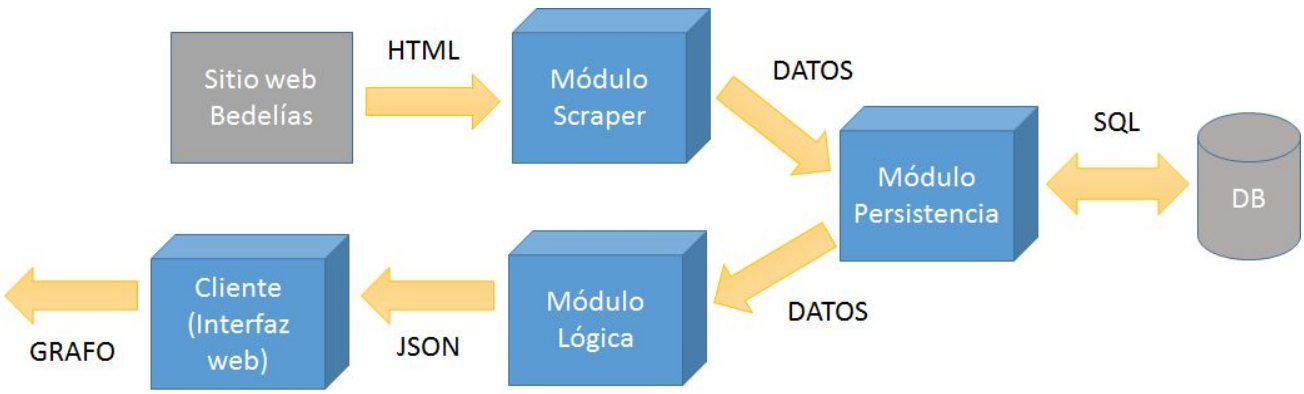
Una situación interesante a tener presente es el caso que para rendir el examen de un curso *X*, se requiere tener aprobado el examen de un curso previo *Y*. Aquí pueden existir dos situaciones dependiendo si el curso *X* es exonerable o no.

- Cuando *X* es exonerable la aplicación permite que el usuario pase al estado Verde (examen aprobado) incluso si no tiene el examen de *Y*.
- Si *X* no es exonerable la aplicación no dejará pasar al estado Verde. La única forma de pasar a este estado es aprobando el examen de *Y*.

Otro caso interesante se da cuando un curso requiere una cantidad mínima de créditos. Aunque se cumpla con todas sus previas, el curso no será habilitado (mostrado) hasta alcanzar la cantidad de créditos mínima.

Arquitectura

La aplicación sigue una arquitectura cliente servidor. El cliente es un navegador web que se comunica con un servidor central para obtener la información de una carrera.
Del lado del servidor se tienen tres módulos los cuales se detallan a continuación:



Módulo Scraper

El Módulo Scraper (implementado en PHP) obtiene documentos html del sitio web de bedelías[1] mediante solicitudes http y luego lo procesa para extraer los datos. Para ello se utiliza la herramienta Goutte[2], una librería PHP para web scraping.
En un primer paso, se obtienen todos los Cursos de la carrera de Ingeniería en Computación. Luego, para cada Curso, se adquieren sus respectivas previas. Si alguna de ellas corresponde a un Grupo, se obtienen los datos del grupo: su puntaje mínimo y máximo, los cursos que lo conforman y los puntos que aporta uno de ellos.
En cuanto a la información de los Cursos, se extraen tanto datos referentes al plan de estudios (créditos que aporta, validez del curso en meses, mínimo de créditos si los requiere), así como datos cuantitativos (cantidad de inscriptos, aprobados y exonerados) y estadísticos (porcentaje de aprobación, nota promedio).

Módulo Lógica

Está compuesto por una función que es llamada por el cliente para obtener la información almacenada en la persistencia (recolectada mediante scraping). Dicha función recorre las distintas tablas de forma tal de construir un grafo (estructura de datos utilizada luego por el cliente) que tiene la siguiente representación:

```
grafo :=  {
        nodos  := { nodo_curso | nodo_grupo | nodo_area }* ,
        aristas := { arista_curso_curso, | arista_curso_grupo, |
        arista_grupo_curso, |      arista_area_curso }*
}

nodo_curso := { idCurso , Nombre, Descripción, Semestre, Créditos, Validez,
Aprobación, Instituto, Exonerable, Semestre, NotaPromedio, Exonerados,
PorcentajeAprobación
 }

nodo_grupo := { idGrupo, Nombre, Mínimo,Máximo }

nodo_area   := { idArea , Nombre}

arista_curso_curso := { idCursoOrigen, idCursoDestino, Actividad,
ActividadPrevia }

arista_curso_grupo := { idCurso, idGrupo, ActividadPrevia, Puntaje }
```

arista_grupo_curso := { idCurso, idGrupo, Actividad}

arista_area_curso := { idArea, idCurso,Creditos, Actividad}

El grafo está compuesto por dos partes principales.

Por un lado, se cuenta con los nodos, de los cuales hay tres tipos:

1. Nodos de tipo Curso: representan los cursos que se tienen en la base de datos y su información asociada.
2. Nodos de tipo Grupo: representan los grupos, de los cuales se cuenta con la información del mínimo y máximo requerido del grupo como previa.
3. Nodos de tipo Área: representan las áreas en las que se agrupan los distintos cursos.

Por otro lado, se cuenta con las aristas, de las cuales hay cuatro tipos:

1. Aristas de tipo Curso - Curso: representan los cursos que son previas de otro curso y se indica la actividad previa (si es necesario el curso o el examen de la previa).

   Ejemplos:

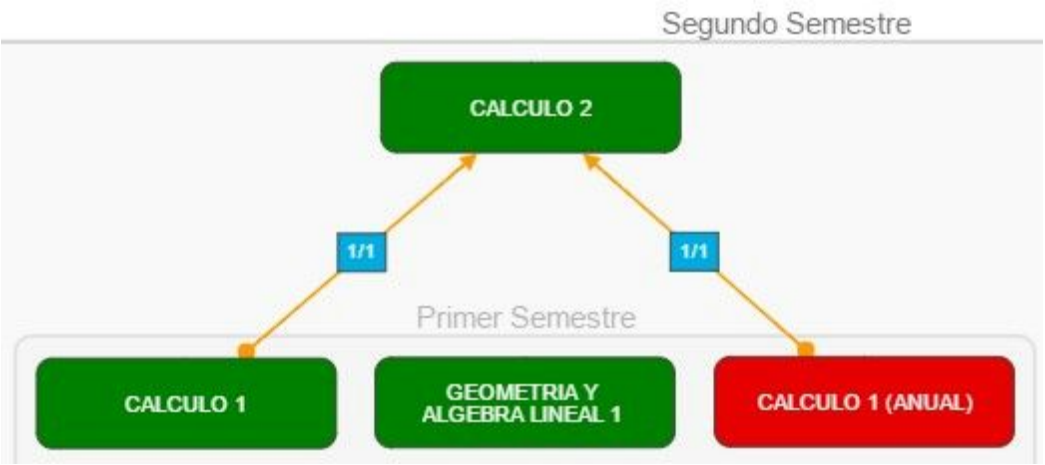
   El curso Matemática Discreta 1 es una previa del curso Lógica, donde la actividad previa es curso (se precisa el curso de Matemática Discreta 1 para poder cursar Lógica).

El curso Matemática Discreta 1 es una previa del curso Programación 3, siendo la actividad previa examen (se precisa el examen de Matemática Discreta 1 para poder cursar Programación 3).

2. Aristas de tipo Curso - Grupo: representan los cursos que pertenecen a un grupo e indican el puntaje que brindan para el grupo y la actividad previa (Curso o Examen).

3. Aristas de tipo Grupo - Curso: representan a los grupos que son previas de un curso.

   Ejemplos:



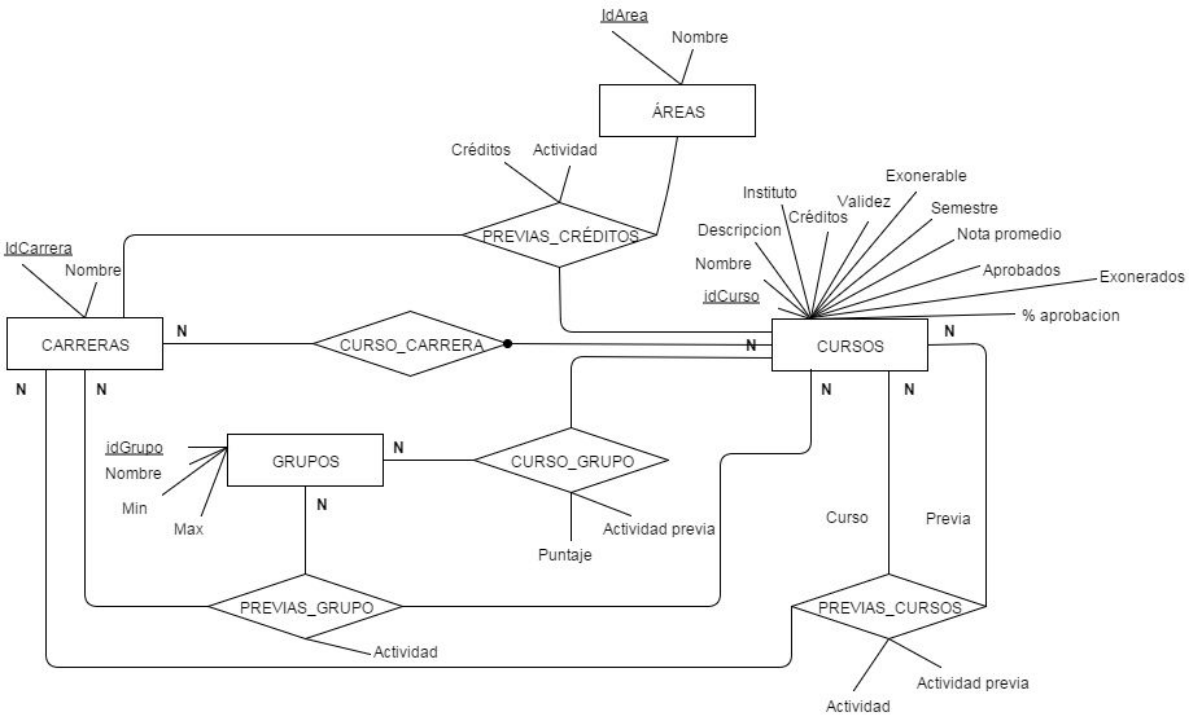
   El grupo compuesto por Cálculo 1 y Cálculo 1 (Anual) es previa del curso Cálculo 2, donde la actividad previa es curso (se precisa el curso de Cálculo 1 o el curso de Cálculo 1 (Anual) para poder cursar Cálculo 2). Además, ambas aristas tienen el cuadrado con valor 1/1, lo que indica que en ambos casos, el puntaje del curso para el grupo es 1 (el numerador) y el puntaje mínimo requerido por el grupo es también 1 (el denominador).

4. Aristas de tipo Área - Curso: representan los cursos que pertenecen al área y contienen los créditos mínimos por área.

Finalmente, al cliente se le devuelve un JSON representando el grafo descrito.

Módulo Persistencia

Almacena la información en la base de datos. A continuación se muestra un diagrama de entidad relación con la realidad modelada.



Puede observarse que para cada tipo de previa existe una relación que la modela, estas son:

previas_cursos: modela cuando un curso es previa de otro.
previas_grupo: modela cuando un grupo es previa de un curso.
previas_créditos: modela el mínimo de créditos que se debe alcanzar en un área para cursar o dar el exámen.

Pasaje de MER a relacional

Del pasaje de MER a relacional surgieron las siguientes tablas:

CARRERAS (idCarrera, nombre)
CURSOS (idCurso, nombre, descripción, instituto, créditos, validez, exonerable, semestre, nota_promedio, aprobados, exonerados, porcentaje_aprobacion)
CURSO_CARRERA (idCarrera, idCurso)
CURSO_GRUPO (idCurso, idGrupo, actividad, puntaje)
GRUPOS (idGrupo, nombre, min, max)
AREAS (idArea, nombre)
PREVIAS_CREDITOS (idCarrera, idCurso, area, creditos)
PREVIAS_CURSOS (idCarrera, idCurso, idPrevia, actividad, actividad_previa)
PREVIAS_GRUPOS (idCarrera, idCurso, idPrevia, actividad)

Cliente

El cliente ejecuta la aplicación web con la que interactúa el usuario. Como se explicó anteriormente el usuario ingresa los cursos aprobados y/o exonerados mientras que la aplicación despliega los cursos habilitados junto a información adicional.
Para esto es necesaria una estructura de datos que contenga la información completa de la carrera, como son sus cursos, previas, estadísticas de aprobación, etc y que será utilizada y consultada por la lógica de la aplicación cliente.
Al comenzar la ejecución del cliente se invoca una operación del módulo lógica en el servidor. Esta es la única operación invocada durante toda la ejecución del cliente y se encarga de construir y retornar la estructura de datos con la información completa de la carrera.
En el primer paso obtiene la información requerida de la persistencia y posteriormente construye la estructura de datos. Finalmente transforma la estructura en formato JSON para poder ser transferida al cliente.
En este punto es importante discutir decisiones de diseño tomadas. En primer lugar hay que destacar que otra opción es realizar múltiples invocaciones al servidor al tiempo que se precisa la información y así lograr un uso eficiente de la memoria.
Por otro lado, la opción tomada minimiza las interacciones entre el cliente y el servidor, logrando una menor dependencia entre ambos y facilitando el desarrollo del cliente. Una razón más fuerte aún es que la aplicación es interactiva, esto quiere decir que el usuario está la mayor parte del tiempo interactuando y observando instantáneamente los resultados de su interacción. Esto conlleva a que los tiempos de transferencia por la red deben ser lo menor posible, de lo contrario el usuario percibirá que la aplicación responde lentamente.

El cliente se encuentra implementado en Javascript y la interfaz gráfica utiliza la librería Cytoscape[3] que permite el dibujo de distintos tipos de grafos y opciones de personalización (media hojas de estilo CSS). La ejecución de la aplicación se resume en los siguientes pasos:

1. *Procesar eventos de E/S*
2. *Actualizar estructura de datos según eventos de E/S*
3. *Dibujar grafo*
4. *Volver a 1.*

En el primer paso se obtienen los eventos de E/S del usuario, como son clics sobre un curso, mouse sobre un curso, etc.
El siguiente paso consiste en actualizar la estructura de datos que mantiene el estado de la aplicación. Por ejemplo, al hacer clic sobre un curso de color rojo, se actualizará la estructura de datos para reflejar el cambio y el color pasará a ser amarillo.
Por último se actualiza el dibujo en pantalla para reflejar los últimos cambios y se vuelve al primer paso.

Herramientas utilizadas

Las herramientas utilizadas a lo largo del proyecto son

Goutte[2]	Librería implementada en PHP utilizada en el módulo scraper para obtener toda la información de la página web de bedelías.
Cytoscape.js[3]	Librería implementada en Javascript para representar en la UI.
Wamp server[4]	Framework compuesto por PHP, Apache y MySQL.

Problemas encontrados

El principal problema encontrado fue que el sistema de previas es tan complejo que es difícil lograr una forma de presentar una buena visualización de la información para el usuario. Se contaba con mucha información y brindarle toda a la vez se volvía engorroso y difícil de entender. Mostrar un grafo de previaturas completo es algo inviable.

En la interfaz brindada, se debe tener en cuenta la diferenciación entre previas de tipo curso y de tipo grupo (se optó por aristas con cuadrados de colores representando los grupos y aristas sin cuadrados para representar las previas de tipo curso cursos) y además poder distinguir entre aprobación del curso y aprobación del examen (se optó por nodos de color amarillo y verde respectivamente).
También se debe poder mostrar toda la información de un curso en particular y debido a que dicha información podría ocupar bastante espacio, se optó por mostrarla en un tooltip.
Además se cuenta con los créditos acumulados, para lo cual se tuvo que obtener la información de créditos de cada curso y mantener un contador de acuerdo a lo que el usuario indique como curso exonerado (sólo se suman créditos a la hora de marcar el curso como curso exonerado - verde).
Dónde mostrar cada nodo tampoco fue algo menor, ya que si no se ubican de forma ordenada, puede ser bastante confuso. Es por esto que se decidió agrupar los cursos por semestre, y no sólo par o impar, sino con números entre 1 y 10 de forma que se pueda ver también el año en el cual se pueden cursar. Esto generó otro problema, ya que dicha información no se encuentra disponible en la web, por lo cual se tuvo que obtener dichos datos manualmente, debido a la falta de información disponible.

Otro problema con el que nos enfrentamos fue la falta de documentación de las librerías. Se tuvo que cambiar tres veces de librería debido a la escasa información sobre ellas en la web.

La Web de Bedelías utiliza HTML obsoleto (casi no utiliza identificadores para etiquetas HTML), lo cual deriva en un Scraping más dificultoso. Además, contiene información ya obsoleta, con lo cual se tuvo que, por ejemplo, realizar un filtrado manual, eliminando los cursos que ya no se dictan más.

Finalmente, un último problema fue que los identificadores de los cursos no siguen un patrón definido, con lo cual se dificulta escribir expresiones regulares que contemplen todos los casos.

Trabajo futuro

Nuestro trabajo se realizó sólo para la carrera de Ingeniería en Computación ya que es la más cercana a nosotros, pero podría perfectamente dar soporte a otras carreras de la Facultad y también a otras Facultades.

Se podría contar con un manejo de usuarios, donde el usuario tiene una cuenta, se loguea y al entrar ya tiene el último estado de su grafo con la cantidad de créditos ya obtenidos, sin necesidad de volver a empezar de cero. También se

podría tener un “vuelta atrás” en caso de que el usuario indique por error un curso salvado.

Debido a que se cuenta con información dinámica de cada curso (por ejemplo, cantidad de aprobados), se debería realizar una actualización de la información (realizar un nuevo Scraping) cada seis meses, es decir, al final de cada semestre, obteniendo los nuevos valores y armando de nuevo la tabla de datos. Además se podría agregar más información extra, como por ejemplo, el Instituto al cual pertenece el curso.

Por último, debido a que el sistema de bedelías es bastante complejo, no se tuvieron en cuenta los casos particulares, es decir, el caso de las reválidas, los cursos expirados y las previas de curso asterisco (cursos que no se pueden cursar si ya se cuenta con una materia que es equivalente, pero probablemente de un plan viejo). Todo esto podría ser tenido en cuenta en un trabajo futuro.

Conclusiones

Nuestro trabajo es un claro ejemplo de un caso donde la complejidad no reside tanto en la recuperación de la información, sino en desplegarla al usuario en forma clara y concisa. Se evitó casos como el que se muestra a continuación en el cual se muestra el grafo completo de la carrera de Ingeniería en computación.

Por otro lado bedelías debería de separar la información obsoleta de la que está vigente debido a que puede confundir a los estudiantes recién ingresados a la facultad.
Respecto a la recuperación de información creemos que es una tarea tediosa si no se cuenta con herramientas que faciliten el trabajo. Otra dificultad es cuando los documentos HTML no tienen una estructura bien definida o utilizan un estándar obsoleto. Además no siempre es posible automatizar la tarea de recuperar información sino que se requiere de algún trabajo manual.

Referencias

[1] - Sitio Web de Bedelías http://bedelias.edu.uy/
[2] - Goutte https://github.com/FriendsOfPHP/Goutte
[3] - Cytoscape js http://js.cytoscape.org/
[4] - Wamp http://www.wampserver.com/