

ESCUELA POLITÉCNICA NACIONAL
FACULTAD DE CIENCIAS
DEPARTAMENTO DE MATEMÁTICAS

ALGORITMOS EVOLUTIVOS
APLICADOS A LA GENERACIÓN
DE HORARIOS PARA COLEGIO

TESIS PREVIA A LA OBTENCIÓN DEL
TÍTULO DE INGENIERO
MATEMÁTICO
MENCIÓN INFORMÁTICA

POR
EDISON FERNANDO MERA MENÉNDEZ.

QUITO, 1999

Certifico que la presente tesis ha sido elaborada bajo mi dirección.

Dr. Polo Vaca Arellano.

Director de Tesis

Agradecimientos:

Al Dr. Polo Vaca Arellano.

Al Mat. Alfonso Salazar.

Al Ing. Oscar Parra.

Al Ing. Luis Torres.

Por la ayuda que me brindaron a lo largo del desarrollo del presente trabajo.

Dedicatoria:

A mis padres: Walter y Filadelfia.

A mis hermanos: Walter, Fernando y Julio.

CONTENIDO

1	INTRODUCCIÓN	7
1.1	METODOLOGÍA A USARSE EN LA ELABORACIÓN DE HORARIOS	8
1.2	VARIABLES Y RESTRICCIONES QUE SE PRESENTAN EN LA ELABORACIÓN DE LOS HORARIOS	10
2	ALGORITMOS EVOLUTIVOS	13
2.1	INTRODUCCION	13
2.2	QUÉ ES UN ALGORITMO EVOLUTIVO	13
2.3	VENTAJAS Y DESVENTAJAS DE LOS ALGORITMOS EVOLUTIVOS	15
3	ELABORACIÓN DE HORARIOS POR MEDIO DE ALGORITMOS EVOLUTIVOS	17
3.1	INTRODUCCIÓN	17
3.2	DEFINICIONES BÁSICAS	17
3.3	CONSTRUCCIÓN DE LA FUNCIÓN OBJETIVO Y LAS RESTRICCIONES	19
3.3.1	CRUCE DE PROFESORES	19
3.3.2	CRUCE DE SESIONES EN UN MISMO CURSO	19
3.3.3	AULAS DE UN TIPO DADO OCUPADAS	20
3.3.4	DURACIÓN DE UNA SESIÓN DADA	21
3.3.5	CRUCE DE SESIONES	21
3.3.6	RESPETO DE LAS PROHIBICIONES DE MATERIAS.	21
3.3.7	RESPETO DE LAS PROHIBICIONES DE PROFESORES.	22
3.3.8	RESTRICCIONES ADICIONALES.	22
3.3.9	CONSTRUCCIÓN DEL MODELO MATEMÁTICO	25
3.4	PROGRAMA EVOLUTIVO UTILIZADO	26
3.4.1	OPERADOR DE CRUZAMIENTO	26
3.4.2	OPERADOR DE MUTACIÓN	31
3.4.3	OPERADOR DE REPARACIÓN	32
3.4.4	ALGORITMO EVOLUTIVO UTILIZADO	32
4	DESARROLLO DEL SISTEMA DE GENERACIÓN DE HORARIOS	35
4.1	INTRODUCCIÓN	35
4.2	ANÁLISIS	36
4.2.1	INTRODUCCIÓN	36
4.2.2	DESCRIPCIÓN DEL PROBLEMA	36
4.2.3	IDENTIFICACIÓN Y CONCEPTUALIZACIÓN DE LOS OBJETOS Y PROCESOS	37
4.3	DISEÑO	41
4.3.1	INTRODUCCIÓN	41
4.3.2	MODELO DE DATOS	41
4.3.3	DISEÑO DEL PROCESO	61
4.3.4	DICCIONARIO DE DATOS	77
4.3.5	SERVICIOS QUE OFRECE EL SISTEMA	79
4.4	IMPLEMENTACIÓN	96

4.4.1	INTRODUCCIÓN	96
4.4.2	IMPLEMENTACIÓN DEL MODELO EN COMPUTADORA	97
4.4.3	MODELO FÍSICO DE LA BASE DE DATOS	104
4.4.4	INTERFAZ DEL USUARIO	105
5	<i>PRUEBAS REALIZADAS</i>	144
6	<i>CONCLUSIONES Y RECOMENDACIONES</i>	147
7	<i>BIBLIOGRAFÍA</i>	150

1 INTRODUCCIÓN

El problema de la elaboración de un horario de clases puede considerarse como un problema de asignación de recursos limitados a ciertas tareas. Los recursos que suelen tomarse en cuenta son el tiempo, los profesores con que se dispone, el espacio físico, la disponibilidad de profesores. Las tareas se refieren a las actividades docentes que realizan los profesores.

Entre los problemas que suelen ocurrir en los colegios, debido a la falta de un sistema automático de generación de horarios, tenemos:

- Dificultad en la elaboración de horarios, lo que demanda el uso de una gran cantidad de horas o días en la elaboración manual.
- Profesores que se ven obligados a dictar en horas incómodas, por lo que se dificulta su buen desempeño.
- Materias que no están ubicadas en un horario óptimo que respete criterios pedagógicos.

Una vez que se cuenta con un sistema automático de generación de horarios, desaparece el problema de *cómo hacer el horario*, y es posible enfocar el problema de *cómo debe ser el horario*, con esto se mejora en gran medida la calidad de la educación y el grado de satisfacción del personal docente y administrativo de los colegios.

Existen muchas formas de enfocar el problema[1][9][3][4][2]. La que se presenta a continuación es la que se ajusta más a la realidad de nuestro medio, desde el punto de vista de la modelización matemática, usando una técnica de solución moderna, basada en los algoritmos evolutivos[24].

1.1 METODOLOGÍA A USARSE EN LA ELABORACIÓN DE HORARIOS

El problema de la elaboración de un horario cae dentro del campo de la Investigación de Operaciones. Este ha sido ampliamente estudiado, y se conoce que es NP-duro [24]p.246. En la práctica, resulta difícil resolverlo manualmente, debido al gran número de variables y restricciones a ser tomadas en cuenta.

Los métodos que suelen emplearse comúnmente son:

Técnicas de coloración en grafos.

Técnicas de optimización en redes.

Empleo de metaheurísticas de búsqueda local: Tabú, Recocido Simulado, Descenso rápido, etc.[1][18]

Algoritmos genéticos y evolutivos[24][3][4][2][23][8].

El problema de los horarios incorpora una serie de restricciones no triviales, y esta es la razón por la cual sólo recientemente se han utilizado algoritmos genéticos y evolutivos.

En el presente trabajo se propone un modelo para la generación de horarios para colegio, así como un mecanismo de solución basado en la implementación de un algoritmo evolutivo. Se presenta además un sistema que incorpora estas ideas, el cual tiene como nombre *Sistema de Generación de Horarios para Colegio, SGHC*, así como los primeros resultados obtenidos en pruebas prácticas. Dichas pruebas se efectuaron con datos reales del Colegio Nacional Mixto Nocturno Salamanca.

La tesis ha sido distribuida en cuatro partes fundamentales:

- El fundamento teórico sobre el cual se desarrolla el sistema, presentado en el capítulo 2.
- La elaboración del modelo matemático, así como el algoritmo evolutivo que permitirá generar el horario, capítulo 3.
- Análisis, diseño e implementación del sistema, capítulo 4.

- Resultados de las pruebas realizadas, las mismas que se llevaron a cabo en el mes de Junio de 1999, capítulo 5.

1.2 VARIABLES Y RESTRICCIONES QUE SE PRESENTAN EN LA ELABORACIÓN DE LOS HORARIOS

Las variables a ser tomadas en cuenta, son las siguientes:

- Días laborables de la semana.
- De cada día, las horas *académicas* en que se dictan clases. A la hora académica de un día dado, se la denominará *período*.
- Aulas existentes, clasificadas por tipos.
- Profesores que trabajan en el colegio.
- Materias que se imparten en el colegio. Una materia es una disciplina que se enseña en el Colegio, por ejemplo: Matemática.
- Cursos, es decir, el nivel con su especialización. El nivel es una etapa de estudio.
- Paralelos, que son cada uno de los grupos que conforman un curso.
- Asignaturas, que es la materia que se dicta en un curso, con un contenido en particular, por ejemplo: Matemática del Sexto Secretariado.

En los horarios para colegio no es necesario mantener una lista de estudiantes, debido a que todos los estudiantes de un mismo paralelo tienen el mismo horario, a diferencia de lo que ocurre en los centros de estudios superiores donde se utiliza el sistema de créditos, en el cual un estudiante se inscribe en las materias que él desea.

Una vez que hemos definido las variables que intervienen en el problema, el colegio en cuestión deberá entonces tomar ciertas decisiones referente a cómo estas variables se relacionan entre sí.

Las decisiones que deberán ser tomadas por el colegio son:

- De cada asignatura, el número de horas semanales a dictar y cómo se distribuye en la semana.
- De cada día laborable de la semana, en qué horas se dictan clases.
- Qué asignaturas se dictarán en cada curso y qué tipo de aula se requiere.
- La carga académica, que indica las asignaturas que deben dictar los profesores en cada uno de los paralelos.
- Las prohibiciones de profesores, es decir, un horario en el que se indiquen las horas en las que un profesor no puede dictar clases y las horas en las que no le agradaría ejercer su carga académica.

- Las prohibiciones de materias, es decir, un horario en el que se indique por cada materia, las horas menos convenientes en que ésta puede ser dictada y las que resulta imposible dictar la materia.

Las restricciones que tienen que ser respetadas, son las siguientes:

- En un paralelo no se pueden dictar varias materias a la misma hora, esto implicaría un cruce de materias.
- Un profesor no puede dictar varias materias a la misma hora, esto implicaría un cruce de profesor.
- Una materia no se puede dictar más veces que el número de aulas existentes del tipo requerido para esa materia, en un momento dado.
- Las horas *huecas* deben ser puestas al final del día. Una hora hueca de un paralelo es aquella que siendo laborable, no tiene asignada una materia.
- Una materia no puede dictarse dos veces en el mismo día. Sin embargo, de existir una materia que dure más de un período en un día, los períodos en que se dicte la materia deberán de ser consecutivos.
- Una materia no puede estar fraccionada, es decir, si en un día dado, se debe dictar más de una hora, no debe estar cortada por un período de descanso, por ejemplo, por la hora de recreo.
- La prohibición de un profesor debe ser respetada, es decir, no debe estar asignado a un período en que le es imposible dictar.
- La prohibición de una materia debe ser respetada, es decir, una materia no debe dictarse en un período en que sea imposible.

Una vez tomadas en cuenta estas variables, y restricciones, viene la tarea de elaborar el horario, esto implica distribuir las materias de cada paralelo en la semana.

El objetivo del modelo será el de elaborar el horario, o al menos, encontrar un horario lo suficientemente bueno que sirva para la toma de la decisión final acerca del horario elegido. Se han considerado los siguientes requisitos básicos que deberá tener el horario encontrado:

- No debe tener cruces de profesores.
- No debe tener asignaturas fraccionadas.
- De tener horas huecas, éstas deben estar ubicadas al final del día.
- No debe tener materias asignadas a períodos en los que sea imposible dictar dicha materia.
- No deberá tener profesores asignados a períodos en los que al profesor le sea imposible dictar la materia.

Estos requisitos necesariamente deben cumplirse, porque de lo contrario el horario no se lo podrá usar. El horario que utiliza actualmente el colegio tomado de ejemplo tiene algunos cruces

de materias, sin embargo, puede que en otros establecimientos no se puedan usar horarios de ese tipo.

2 ALGORITMOS EVOLUTIVOS

2.1 INTRODUCCION

Los primeros algoritmos basados en ideas genéticas para resolver problemas de optimización fueron utilizados más de treinta años atrás, siguiendo el desarrollo de las ideas fundamentales de los algoritmos genéticos por John Holland de la Universidad de Michigan. Durante la década de los 70, este trabajo continuó, pero pasó desapercibido. En los últimos años, sin embargo, se ha ido incrementando el interés en los algoritmos genéticos. La razón de este creciente interés se debe en gran medida al aumento de la capacidad de los computadores y los desarrollos que han existido en el campo del procesamiento en paralelo.

2.2 QUÉ ES UN ALGORITMO EVOLUTIVO

Un algoritmo evolutivo es un *algoritmo probabilístico* que mantiene una población de individuos $P(t) = \{x'_1, \dots, x'_n\}$ para una iteración t . Cada individuo representa una solución potencial del problema a resolver, y en un algoritmo evolutivo, es implementado como alguna estructura de datos, posiblemente compleja S . Cada solución x'_i es evaluada para dar alguna medida de su fuerza. Luego, una nueva población se forma seleccionando aleatoriamente los individuos más fuertes, a este paso se lo denomina selección. Algunos de los miembros de la nueva población sufren ciertas transformaciones por medio de operadores *genéticos* para formar nuevas soluciones, este paso se denomina alteración. Existen transformaciones unarias m_i , las cuales crean nuevos individuos a través de un cambio pequeño en un individuo ($m_i : S \rightarrow S$) denominadas *mutaciones*, y transformaciones de mayor orden c_j , las cuales operan sobre un mayor número de individuos ($c_j : S \times \dots \times S \rightarrow S$) denominadas *cruzamientos*. Después de un cierto número de generaciones, el programa converge a una solución casi óptima[24]p.1-3.

Ahora bien, para un problema dado, pueden formularse muchos algoritmos evolutivos. Estos pueden diferir de diversas maneras, por ejemplo, puede que usen diferentes estructuras de datos para representar un individuo, usar diferentes operadores *genéticos*, métodos para crear la población inicial, forma de manejar las restricciones del problema, parámetros del algoritmo como son el tamaño de la población, probabilidad de aplicar uno u otro operador, etc.

Una observación que debe hacerse, respecto a los algoritmos evolutivos frente a los algoritmos genéticos clásicos, es que tienen un carácter más generalizado. En los algoritmos genéticos clásicos, la estructura de datos de un cromosoma la constituye una cadena de bits de longitud fija, y los operadores genéticos son la mutación binaria y el cruce binario. En otras palabras, un algoritmo genético es un algoritmo evolutivo, sólo que más restringido. En los algoritmos evolutivos los cromosomas no tienen por qué ser cadenas de bits, pueden ser estructuras más complejas y abstractas, y el proceso de alteración deberá incluir los operadores apropiados para la estructura y el problema dados[24]p.5-9.

Los algoritmos evolutivos pueden usarse en una amplia gama de problemas, entre estos tenemos, problemas de análisis numérico, investigación de operaciones, simulación de fenómenos biológicos relacionados con la evolución, problemas de inteligencia artificial, redes neuronales, etc.

Para los problemas combinatorios, rara vez suele utilizarse un *algoritmo puramente evolutivo*, por lo general suele incorporarse algo de conocimiento de la naturaleza del problema en forma de heurísticas, dando lugar a lo que se denomina *algoritmos evolutivos híbridos*.

También puede incorporarse conocimiento heurístico al algoritmo evolutivo en forma de operadores de reparación r_k ($r_k : S \rightarrow S$), los cuales dado un individuo, lo alteran para que cumpla al menos con las restricciones principales. Un método de reparación independiente del problema es utilizar otra técnica de solución, como por ejemplo el descenso rápido.

El proceso de simulación de la evolución permite que el principio de *supervivencia del más fuerte* se cumpla para problemas matemáticos, donde el objetivo es encontrar un individuo que sea *el más fuerte*. Por ejemplo, si el problema requiere que se elabore un horario en el cual un profesor no aparezca más de una vez en un período dado, entonces la solución será representada por un horario individual que no tiene cruces de profesores. La población consiste en un conjunto de diferentes horarios, muchos de los cuales podrían no ser factibles. Un individuo podría haber sido generado por horarios padres con cruces, sin embargo, la combinación particular de genes de los padres hacen que dicho individuo no tenga cruces.

2.3 VENTAJAS Y DESVENTAJAS DE LOS ALGORITMOS EVOLUTIVOS

Los algoritmos evolutivos permiten explorar el espacio de búsqueda de una forma más adecuada, ya que éstos no trabajan con una única solución que va cambiando, sino con un conjunto de soluciones variadas, lo que hace menos probable quedarse atrapado en mínimos locales, como es el caso del método de descenso rápido, y bajo ciertas condiciones, del recocido simulado.

Contrariamente a lo que pueda pensarse, los algoritmos evolutivos no ofrecen un rendimiento mayor a otros algoritmos utilizados, la principal razón para que estos se hayan hecho populares es que son paralelizables, es decir, pueden ser modificados para que varias máquinas compartan el trabajo de resolver el problema. Debe aclararse que en la presente tesis no se realizará esta experiencia, puesto que escapa a los alcances de la misma.

En los problemas combinatorios, debido al bajo rendimiento de los algoritmos evolutivos, se deben utilizar técnicas más avanzadas, como son combinación con heurísticas de reparación, métodos de cruce más complejos que procuran que las restricciones más importantes no sean violadas, mejoramiento de cada individuo con heurísticas de búsqueda local, etc. En la práctica, la mayoría de problemas combinatorios son muy difíciles porque simplemente, el algoritmo genético no converge. La razón es justamente la forma de la función objetivo. Los factores más importantes que originan ese problema son los siguientes:

- La función objetivo es en general una suma de penalizaciones, no una función continua.
- La función objetivo depende de la satisfacción de muchas restricciones, de tal forma que un cambio pequeño en un cromosoma puede causar un cambio grande en el valor.
- El espacio de búsqueda es demasiado grande, aún para problemas pequeños.

Estos hechos nos permiten suponer, que existen muchos óptimos locales, bastante aislados. Si el óptimo global está aislado y existen uno o más óptimos locales, que no están tan aislados y, por lo tanto, más fáciles de encontrar, el algoritmo evolutivo puede perderse en su búsqueda y dirigirse al óptimo local. Las funciones con estas estructuras *extraviantes* son llamadas *decepcionantes*[23]. Las funciones decepcionantes han sido bastante estudiadas. Entre las propuestas dadas para optimizar este tipo de funciones, tenemos: reordenación, uso de operadores genéticos especiales y algoritmos genéticos alterados. En la presente tesis, yo he propuesto un algoritmo evolutivo que utiliza una técnica de reparación fuerte, consistente en tomar un individuo y aplicar un descenso rápido que logra evitar que se violen la mayor parte de restricciones, una variante del operador especial de

cruzamiento por *claves aleatorias*[15], cuyo funcionamiento se explicará posteriormente en 3.4.1 y el modelo de selección elitista con cierto ajuste al presente problema.

3 ELABORACIÓN DE HORARIOS POR MEDIO DE ALGORITMOS EVOLUTIVOS

3.1 INTRODUCCIÓN

En este capítulo se realiza la elaboración de un modelo matemático de optimización combinatoria para el problema de la elaboración de un horario para colegio, para lo cual ha sido dividido en tres partes. En la primera parte se realizan las definiciones básicas. La segunda parte construye el modelo matemático, que consiste de una función objetivo y varias restricciones, con esto la elaboración de un horario se reduce a la búsqueda de los valores óptimos de ciertas variables enteras. La tercera parte explica de una forma más amplia el programa evolutivo utilizado para resolver el problema, que es una variación el modelo de selección elitista.

A lo largo del capítulo se muestran ciertas técnicas avanzadas en la realización de algoritmos evolutivos, como son las claves aleatorias, la reparación de individuos, cómo funcionan los operadores de cruce y mutación, y una técnica de relajación que disminuye el número de restricciones fuertes aumentando la complejidad de la función objetivo.

3.2 DEFINICIONES BÁSICAS

A continuación se definen las variables y los datos del problema.

Sean las siguientes matrices:

$$h_{ijk} = \begin{cases} 1 & \text{si en el paralelo } i, \text{ período } j, \text{ se dicta la sesión } k \\ 0 & \text{caso contrario} \end{cases} \quad [1]$$

$$p_{ijk} = \begin{cases} 1 & \text{si el profesor } i, \text{ dicta en el paralelo } j, \text{ la sesión } k \\ 0 & \text{caso contrario} \end{cases} \quad [2]$$

$$m_{ij} = \begin{cases} 1 & \text{si la materia } i \text{ corresponde a la sesión } j \\ 0 & \text{caso contrario} \end{cases} \quad [3]$$

$$g_{ij} = \begin{cases} 1 & \text{si en el paralelo } i \text{ se dicta la sesión } j \\ 0 & \text{caso contrario} \end{cases} \quad [4]$$

$$a_i = \text{número de aulas de tipo } i \quad [5]$$

$$t_{ijk} = \begin{cases} 1 & \text{si la materia } i \text{ del paralelo } j \text{ requiere el aula de tipo } k \\ 0 & \text{caso contrario} \end{cases} \quad [6]$$

$$l_i = \text{duración de la sesión } i \quad [7]$$

$$d_{ij} = \begin{cases} 1 & \text{si el período } i \text{ correspond e al día } j \\ 0 & \text{caso contrario} \end{cases} \quad [8]$$

$$e_{ij} = \begin{cases} 1 & \text{si el período } i \text{ correspond e a la hora } j \\ 0 & \text{caso contrario} \end{cases} \quad [9]$$

$$\alpha_i = \text{valor del nivel de preferenci a } i \quad [10]$$

$$r_{ij} = \begin{cases} \alpha_k & \text{si la materia } i \text{ en el período } j \text{ tiene el nivel de preferenci a } k \\ 0 & \text{caso contrario} \end{cases} \quad [11]$$

$$\beta_i = \text{valor del nivel de preferenci a } i \quad [12]$$

$$s_{ij} = \begin{cases} \beta_k & \text{si el profesor } i \text{ en el período } j \text{ tiene el nivel de disponibil idad } k \\ 0 & \text{caso contrario} \end{cases} \quad [13]$$

El problema se reducirá a encontrar $H = (h_{ijk})$, con el resto de variables como datos, que cumpla ciertos requisitos. En términos matemáticos, el modelo sería el siguiente:

$$\min f(H)$$

sujeto a

$$g(H)$$

Donde f representa la función objetivo y g es una proposición lógica.

Estas variables nos permiten construir el modelo, el cual va a consistir de un conjunto de restricciones *fuertes*, y una función objetivo a minimizar. La función objetivo contiene penalizaciones para aquellas restricciones que no se consideró oportuno incluir entre las restricciones fuertes. El proceso que se sigue para transformar una restricción fuerte en una restricción débil se denomina *relajación*.

3.3 CONSTRUCCIÓN DE LA FUNCIÓN OBJETIVO Y LAS RESTRICCIONES

3.3.1 CRUCE DE PROFESORES

Un profesor no puede dictar en más de un paralelo a la vez, en un período dado.

Utilizando las definiciones [1] y [2], tenemos que:

$$\sum_{j,k} p_{ijk} h_{jlk} = \text{número de veces que dicta el profesor } i \text{ en el período } l \quad [14]$$

Por lo tanto, [14] no puede ser mayor que 1, de acuerdo con la restricción a considerar.

Por lo tanto:

$$\sum_{j,k} p_{ijk} h_{jlk} \leq 1 \quad [15]$$

Ahora bien, para relajar esta restricción, y pasarla a la función objetivo, se debe definir una función f_p , la cual sea igual a cero cuando se cumpla [15], y mayor a cero cuando no. Una función de tales características puede construirse de la siguiente manera:

$$f_p(H) = \sum_{i,l} \max \left(\sum_{j,k} p_{ijk} h_{jlk} - 1, 0 \right) \quad [16]$$

Esta función pasará a ser parte de la función objetivo.

3.3.2 CRUCE DE SESIONES EN UN MISMO CURSO

En los paralelos de un curso no se puede dictar más de una sesión en un período dado.

Esta restricción se traduce a:

$$\sum_k h_{ijk} \leq 1 \quad [17]$$

En el modelo será considerada restricción fuerte.

3.3.3 AULAS DE UN TIPO DADO OCUPADAS

No se pueden dictar más materias que requieren de un tipo de aula dado que el número de aulas disponibles de este tipo.

Considérese lo siguiente:

$$\sum_k m_{kl} h_{ijl} \leq 1 \quad [18]$$

Esta suma representa el número de veces que se dicta la materia l en el paralelo i , período j y no hace falta incluirla como restricción, debido a la restricción [17] y a que una sesión sólo está asociada a una materia, es decir:

$$\sum_i m_{ij} \leq 1 \quad [19]$$

Lo cual es una propiedad de los datos del modelo.

Con ayuda de [17], podemos construir lo siguiente:

$$\sum_{k,l,i} t_{kin} m_{kl} h_{ijl} = \text{número de aulas del tipo } n \text{ ocupadas en el período } j$$

La restricción considerada quedaría escrita de la siguiente forma:

$$\sum_{k,l,i} t_{kin} m_{kl} h_{ijl} \leq a_n \quad [20]$$

Finalmente definimos:

$$f_t(H) = \sum_{n,i} \max \left(\sum_{k,l,i} t_{kin} m_{kl} h_{ijl} - a_n, 0 \right) \quad [21]$$

3.3.4 DURACIÓN DE UNA SESIÓN DADA

Una sesión k dura l_k períodos, como se observa a continuación:

$$\sum_j h_{ijk} = l_k \quad [22]$$

En el modelo será considerada restricción fuerte.

3.3.5 CRUCE DE SESIONES

En un día, no se deben dictar dos sesiones diferentes correspondientes a la misma materia del mismo paralelo.

Esta restricción puede traducirse a:

$$\sum_i d_{in} \sum_k m_{ik} h_{jik} \leq l_o, \quad o \in \{x / m_{ix} > 0\} \quad [23]$$

Sin embargo, sólo se presenta para el conocimiento de la forma de calcular los cruces de sesiones, ya que posteriormente, en las restricciones adicionales, se describe una componente de la función objetivo que contiene esta restricción.

3.3.6 RESPETO DE LAS PROHIBICIONES DE MATERIAS.

Definimos:

$$f_r(H) = \sum_{i,j,k,l} h_{jik} m_{ik} r_{li} \quad [24]$$

Esta es la penalización de todas las prohibiciones de materias en todos los paralelos y todas las horas laborables.

3.3.7 RESPETO DE LAS PROHIBICIONES DE PROFESORES.

Definimos:

$$f_s(H) = \sum_{i, j, k, l} h_{jlk} p_{ijk} s_{il} \quad [25]$$

Esta es la penalización de todas las prohibiciones de profesores en todos los paralelos y todas las horas laborables.

3.3.8 RESTRICCIONES ADICIONALES.

Las restricciones adicionales son aquellas que no aparecen directamente en el problema, sino de la observación de cómo funcionan los colegios. Las restricciones consideradas son las siguientes:

Las horas huecas deben ser puestas al final del día.

Sea n el número de horas disponibles en un día. Para saber si una hora es hueca, tenemos que ver que h_{ijk} sea cero. Para saber si un período j dado está puesto al final del día, tenemos que verificar que e_{nj} sea distinto de cero, luego, $(1 - h_{ijk})e_{nj}$ vale 1 si el período j tiene una hora hueca que no está puesta al final del día, y 0 en caso contrario. Sumando respecto a los índices, se obtiene la siguiente restricción:

$$\sum_{i, j, k} (1 - h_{ijk})e_{nj} = 0 \quad [26]$$

Para posteriormente relajar esta restricción, definimos la función:

$$f_e(H) = \sum_{i, j, k} (1 - h_{ijk})e_{nj} \quad [27]$$

Las materias iguales deben ser consecutivas, es decir, no pueden estar cortadas, ni por el día ni por hora, por ejemplo, por la hora de recreo.

Esta restricción contiene algunos elementos que, si bien se evalúan en la misma componente de la función, tienen significados diferentes. En primer lugar, contiene el hecho de que dos sesiones

de la misma materia no deben dictarse el mismo día, en segundo lugar el que una hora no pueda ser cortada por el recreo o una hora hueca, y por último el que una sesión no se pueda dictar a la última hora el primer período y a la primera hora del siguiente día los demás períodos. En otras palabras, una sesión no puede estar cortada por el día.

La razón de tratar todo esto en conjunto es porque la siguiente componente de la función objetivo contiene todas estas restricciones y las evalúa en un solo paso.

Para realizar la evaluación de esta restricción para el horario de un paralelo, vemos para cada materia asignada al paralelo, la hora mínima y máxima a la que se dicta y se comparara con la duración de la sesión en que consiste dicha materia. Si hay varias sesiones puestas en el mismo día, la duración será menor que la diferencia entre la hora máxima y mínima, lo mismo si está fragmentada ya sea por el día o por una hora libre. Definamos lo siguiente:

$$\begin{aligned}\gamma_{ijk} &= \text{hora mínima en la que se dicta la materia } i \text{ del paralelo } j \text{ en el día } k \\ \Gamma_{ijk} &= \text{hora máxima en la que se dicta la materia } i \text{ del paralelo } j \text{ en el día } k\end{aligned}$$

Por la forma como han sido definidas, ambas son funciones de H y no nuevas variables introducidas al modelo.

Realizando una suma del resto $(\gamma_{ijk} - \Gamma_{ijk})m_{mi} - l_m$, $m_{mj} \neq 0$ respecto a los índices libres, se tiene lo siguiente:

$$f_m(H) = \sum_{m, j, k} (\gamma_{ijk} - \Gamma_{ijk})m_{mi} - l_m \quad [28]$$

Las materias deben estar lo más dispersas posible.

La forma de medir la dispersión de materias es, dada una materia, y un paralelo, contar el número de *saltos*, es decir, el número de veces que se pasa de un día en que no se da la materia a otro día en que sí se da y viceversa. Por ejemplo: si se tiene la materia 1, que se dicta en tres sesiones durante 5 días, tenemos:

1	0	1	0	1
No	Sí	Sí	Sí	Sí

En este caso, la segunda fila indica cuándo hubo *saltos*. Aquí, el número de saltos es 4, que además es el número máximo de saltos que se puede tener. La penalización respectiva para este requerimiento, consistirá en una fórmula que devuelva el número máximo de saltos para una materia dada. Obsérvese que entre el quinto día y el primer día, se debe considerar la existencia de un *salto*, debido a que entre éstos viene el fin de semana, para tomar en cuenta esto se considera al fin de semana como un día virtual en el cual no se dicta la materia. Este requerimiento nos permite observar lo siguiente:

$$\sum_{j,k} h_{ijk} m_{ik} d_{jn} = \text{períodos que se dictan en el paralelo } i, \text{ la materia } l \text{ en el día } n \quad [29]$$

Sea:

$$\delta(x) = \begin{cases} 1 & \text{si } x > 0 \\ 0 & \text{caso contrario} \end{cases} \quad [30]$$

$$\sum_j g_{ij} m_{ij} = \lambda_{il} = \text{número de sesiones de la materia } l \text{ en el paralelo } i \quad [31]$$

$$\sum_j g_{ij} m_{ij} l_j = \text{duración de la materia } l \text{ en el paralelo } i \quad [32]$$

Luego, el número máximo de saltos para una materia dada, será:

$$\sigma_{il} = 2 \min (\lambda_{il}, n_e - \lambda_{il}) \quad [33]$$

Sumando respecto a n , se obtiene el número de saltos de una materia l en un paralelo i :

$$\sum_n \left| \delta \left(\sum_{j,k} h_{ijk} m_{ik} d_{jn} \right) - \delta \left(\sum_{j,k} h_{ijk} m_{ik} d_{j,n+1} \right) \right| \quad [34]$$

Puesto que el objetivo es el de maximizar el número de saltos sin originar cruces, tendremos:

$$\sum_n \left| \delta \left(\sum_{j,k} h_{ijk} m_{lk} d_{jn} \right) - \delta \left(\sum_{j,k} h_{ijk} m_{lk} d_{j,n+1} \right) \right| = \sigma_{il} \quad [35]$$

Finalmente, definimos lo siguiente:

$$f_o(H) = \left| \sum_n \left| \delta \left(\sum_{j,k} h_{ijk} m_{lk} d_{jn} \right) - \delta \left(\sum_{j,k} h_{ijk} m_{lk} d_{j,n+1} \right) \right| - \sigma_{il} \right| \quad [36]$$

3.3.9 CONSTRUCCIÓN DEL MODELO MATEMÁTICO

La función objetivo está constituida de la composición lineal de cada una de las funciones que se definieron anteriormente:

$$f(H) = (w_p f_p + w_t f_t + f_r + f_s + w_e f_e + w_m f_m + w_o f_o)(H) \quad [37]$$

Donde w_p, w_t, w_e, w_m , y w_o son pesos mayores de cero, que deben ser calibrados experimentalmente.

Reuniendo las restricciones fuertes, el modelo puede escribirse de la siguiente manera:

Minimizar $f(H)$

Sujeto a :

$$\begin{aligned} \sum_k h_{ijk} &\leq 1 \\ \sum_j h_{ijk} &= l_k \end{aligned} \quad [38]$$

La forma como están representados los datos en el sistema impide que las restricciones fuertes no se cumplan.

3.4 PROGRAMA EVOLUTIVO UTILIZADO

En los algoritmos evolutivos se deben crear los operadores genéticos, los cuales son: cruzamiento, mutación y reparación, a continuación se explica cómo funciona cada uno.

3.4.1 OPERADOR DE CRUZAMIENTO

Uno de los mayores problemas es el crear un método heurístico que permita a partir de dos horarios obtener otro que cumpla con todas las restricciones fuertes, además que conserve las características de sus padres.

La técnica empleada para este efecto se conoce con el nombre de *claves aleatorias*, la cual consiste en transformar el espacio de búsqueda original en un espacio de números en el cual es más simple hacer el cruzamiento y a los hijos obtenidos se les aplica la transformación inversa para volver al espacio de búsqueda original. Para aclarar estas ideas se mostrará un ejemplo:

Considere la tabla de materias y sesiones siguientes de un paralelo en un colegio hipotético que labora 4 horas diarias durante 5 días:

Materia	Sesión	Clave Aleatoria
Matemática	1	1281002695
Matemática	1	1281002695
Matemática	2	536890299
Matemática	2	536890299
Matemática	3	1488197047
Castellano	4	263320962
Castellano	5	958682929
Castellano	6	1110561000
Castellano	7	203956922
CC.SS	8	2059995090
CC.SS	8	2059995090
CC.SS	9	447116415
CC.SS	9	447116415
CC.NN	10	254681439
CC.NN	11	2009272608
CC.NN	12	292868494
CC.NN	13	1234700359
CC.NN	14	1753380145
Inglés	15	1876341393
Inglés	16	445467496
Inglés	17	965916144

La clave aleatoria para este ejemplo consiste de un número generado aleatoriamente, asociado a la sesión. Ordenando respecto de la clave aleatoria, se genera un listado de sesiones distribuidas aleatoriamente, que es un horario generado aleatoriamente, de esta forma también se procede para encontrar los primeros individuos. Luego, se asocia a cada materia el período en orden, y el horario quedaría configurado de la siguiente manera:

Tabla de Materias y Sesiones ordenada por la Clave Aleatoria:

Materia	Sesión	Clave Aleatoria
Castellano	7	203956922
CC.NN	10	254681439
Castellano	4	263320962
CC.NN	12	292868494
Inglés	16	445467496
CC.SS	9	447116415
CC.SS	9	447116415
Matemática	2	536890299
Matemática	2	536890299
Castellano	5	958682929
Inglés	17	965916144
Castellano	6	1110561000
CC.NN	13	1234700359
Matemática	1	1281002695
Matemática	1	1281002695
Matemática	3	1488197047
CC.NN	14	1753380145
Inglés	15	1876341393
CC.NN	11	2009272608
CC.SS	8	2059995090
CC.SS	8	2059995090

Horario por Materias:

Hora	L	M	M	J	V
1	Castellano	Inglés	Matemática	CC.NN	CC.NN
2	CC.NN	CC.SS	Castellano	Matemática	Inglés
3	Castellano	CC.SS	Inglés	Matemática	CC.NN
4	CC.NN	Matemática	Castellano	Matemática	CC.SS

Para visualizar cómo se cruzan dos horarios, considérese el siguiente horario:

Materia	Sesión	Clave Aleatoria
CC.SS	8	6249955
CC.SS	8	6249955
CC.SS	9	89630970
CC.SS	9	89630970
Ingles	16	105535927
Castellano	7	156079396
CC.NN	14	217472088
Castellano	4	393059963
Matemática	2	724229936
Matemática	2	724229936
Matemática	1	760980852
Matemática	1	760980852
CC.NN	10	890675027
CC.NN	12	910125444
CC.NN	13	968869126
CC.NN	11	1072366980
Castellano	6	1292920585
Matemática	3	1417363633
Ingles	15	1648460537
Castellano	5	1667605070
Ingles	17	1835046322

Horario por Materias:

Hora	L	M	M	J	V
1	CC.SS	Ingles	Matemática	CC.NN	Castellano
2	CC.SS	Castellano	Matemática	CC.NN	Matemática
3	CC.SS	CC.NN	Matemática	CC.NN	Ingles
4	CC.SS	Castellano	Matemática	CC.NN	Castellano

Podemos generar dos hijos, cortando el primer horario y el segundo a la misma altura, y combinando los dos trozos. La otra alternativa es generando un vector aleatorio de unos y ceros, y según valga uno o cero una componente de dicho vector, tomamos del un horario o del otro el valor de la clave aleatoria. En la siguiente tabla se muestra cómo funcionaría esta técnica, que es la utilizada en la implementación del algoritmo:

Horario ordenado por Sesiones:

Materia	Sesión	Clave Horario 1	Clave Horario 2	Vector Aleatorio de Selección	Clave del Nuevo Horario
Matemática	1	1281002695	760980852	0	760980852
Matemática	1	1281002695	760980852	0	760980852
Matemática	2	536890299	724229936	1	536890299
Matemática	2	536890299	724229936	1	536890299
Matemática	3	1488197047	1417363633	0	1417363633
Castellano	4	263320962	393059963	0	393059963
Castellano	5	958682929	1667605070	1	958682929
Castellano	6	1110561000	1292920585	1	1110561000
Castellano	7	203956922	156079396	1	203956922
CC.SS	8	2059995090	6249955	0	6249955
CC.SS	8	2059995090	6249955	0	6249955
CC.SS	9	447116415	89630970	0	89630970
CC.SS	9	447116415	89630970	0	89630970
CC.NN	10	254681439	890675027	1	254681439
CC.NN	11	2009272608	1072366980	0	1072366980
CC.NN	12	292868494	910125444	1	292868494
CC.NN	13	1234700359	968869126	0	968869126
CC.NN	14	1753380145	217472088	0	217472088
Inglés	15	1876341393	1648460537	1	1876341393
Inglés	16	445467496	105535927	1	445467496
Inglés	17	965916144	1835046322	1	965916144

Ordenando por la clave aleatoria del nuevo horario, y construyendo el horario se tiene:

Hora	L	M	M	J	V
1	CC.SS	Castellano	Castellano	Matemática	CC.NN
2	CC.SS	CC.NN	Inglés	Matemática	CC.NN
3	CC.SS	CC.NN	Matemática	Castellano	Castellano
4	CC.SS	CC.NN	Matemática	Inglés	Matemática

Esta forma de representar el horario siempre conserva las restricciones fuertes.

En este ejemplo se ha mostrado la forma de realizar el cruce entre dos horarios de un paralelo cualquiera. Para realizar el cruzamiento del horario del colegio se debe repetir esta operación con todos los paralelos.

3.4.2 OPERADOR DE MUTACIÓN

Primero, vamos a considerar una mutación de orden 1, que consiste en la mínima alteración posible a un horario. Las mutaciones se realizan seleccionando dos períodos aleatoriamente e intercambiando los valores que toman allí las claves aleatorias, lo que provoca el intercambio de las sesiones asociadas a estas claves aleatorias, por ejemplo:

Materia	Sesión	Clave Aleatoria	Clave Aleatoria Mutada
CC.SS	8	6249955	6249955
CC.SS	8	6249955	6249955
CC.SS	9	89630970	*445467496
CC.SS	9	89630970	*445467496
Castellano	7	203956922	203956922
CC.NN	14	217472088	217472088
CC.NN	10	254681439	254681439
CC.NN	12	292868494	292868494
Castellano	4	393059963	393059963
Inglés	16	445467496	*89630970
Matemática	2	536890299	536890299
Matemática	2	536890299	536890299
Matemática	1	760980852	760980852
Matemática	1	760980852	760980852
Castellano	5	958682929	958682929
Inglés	17	965916144	965916144
CC.NN	13	968869126	968869126
CC.NN	11	1072366980	1072366980
Castellano	6	1110561000	1110561000
Matemática	3	1417363633	1417363633
Inglés	15	1876341393	1876341393

Para generar la clave aleatoria mutada, se intercambió el segundo número aleatorio con el octavo. Al ordenar respecto de la clave aleatoria mutada se obtiene un nuevo horario, con una cantidad mínima de horas alteradas.

La mutación de orden n consiste en aplicar n veces la mutación de orden 1 en un horario dado.

El otro operador de mutación que se puede definir consiste en intercambiar las sesiones que se dictan en un día con las de otro día. Esta mutación mueve más sesiones, y fue puesta para mejorar el rendimiento del algoritmo evolutivo.

3.4.3 OPERADOR DE REPARACIÓN

La tarea de este operador es reducir el número de cruces por medio de una técnica heurística, que verifica todo el horario y realiza los movimientos tendientes a minimizar el valor de la función objetivo, cuando éstos son simples. Para aprovechar la evaluación de la función objetivo, se aplica un descenso rápido en el individuo a ser reparado. El algoritmo de descenso rápido tiene el siguiente aspecto:

```
Descenso_Rapido;
Inicio
  Repetir
    NoMejora := Verdadero;
    Para cada (i: paralelo, j: periodo1, j1: periodo2; j <> j1) Hacer
      Valor := MejorHorario.Valor;
      Intercambiar(i, j, j1);
      Valor2 := MejorHorario.Valor;
      Si Valor2 > Valor Entonces
        Deshacer Intercambiar(i, j, j1)
      caso contrario
        Si Valor2 < Valor Entonces
          Valor := Valor2;
          NoMejora := Falso;
        Fin Si;
      Fin Caso contrario;
    Fin Para
  Hasta que NoMejora
Fin.
```

La versión mejorada de este algoritmo, no evalúa toda la función objetivo sino únicamente el incremento o decremento que se produce al alterar el horario. Esto fue lo que se implementó definitivamente y el rendimiento del algoritmo aumentó considerablemente.

3.4.4 ALGORITMO EVOLUTIVO UTILIZADO

En el modelo elitista de selección, la nueva población de horarios generada siempre preserva el mejor horario encontrado hasta el momento, garantizándose una convergencia rápida del algoritmo, sin embargo, existe el problema de que el algoritmo converja a un óptimo local y no al global. Para ajustar el algoritmo evolutivo al problema de los horarios e impedir la convergencia prematura es necesario preservar además del mejor horario encontrado, los horarios con la menor cantidad de cruces de sesiones, cruces de profesor y cruces de tipos de aula.

Otro problema que se presenta es el de elaborar una función que indique la *fuerza* de un individuo. Debido a que el objetivo es minimizar la función, hay que realizar una transformación

sobre ésta para hallar un valor de la *fuerza* del individuo. La forma más simple consiste en restar del máximo valor hallado de la función objetivo, el valor que toma la función objetivo para un individuo dado:

$$v(H) = \max_i (f(H_i)) - f(H)$$

El máximo debe tomarse sobre la población actual, no sobre todos los horarios posibles.

El algoritmo evolutivo quedó configurado de la siguiente manera:

```

Programa_Evolutivo
Inicio
    Inicializar;
    Reparar;
    Evaluar;
    TomarElMejor;
    Numero_Generacion := 0;
    mientras Numero_Generacion < Numero_Maximo_Generacion
    hacer
        Seleccionar;
        Cruzar;
        Mutar;
        Reparar;
        Evaluar;
        Elitista;
        Numero_Generacion := Numero_Generacion + 1;
    Fin mientras;
Fin.

```

La función Cruzar toma dos individuos, y según una probabilidad de cruce, aplica el operador de cruzamiento a los dos individuos, así procede para todos los individuos.

La función Mutar aplica el operador de mutación de acuerdo a una probabilidad de mutación a cada individuo.

La función Reparar aplica el operador de reparación de acuerdo a una probabilidad de reparación a cada individuo.

La función Evaluar encuentra el valor de la *fuerza* de cada individuo.

La función Elitista se encarga de ver si hubo mejora y preservar el mejor individuo encontrado hasta el momento. Si no hubo mejora, el peor individuo es reemplazado por el mejor.

La función Seleccionar selecciona los individuos que pasan a la siguiente generación. Los individuos son seleccionados de acuerdo a un valor probabilístico que se crea con ayuda del valor

de su *fuerza*. El método para seleccionar se conoce con el nombre de *ruleta con pesos*, o *distribución por tabla*.

Sean:

n = tamaño de la población

$$p_i = \frac{v(H_i)}{\sum_j v(H_j)}, \quad i = 1, 2, \dots, n$$

$$l_0 = 0, \quad l_i = l_{i-1} + p_i, \quad i = 1, 2, \dots, n$$

$$I_1 = [l_0, l_1[$$

$$I_i = [l_{i-1}, l_i[, \quad i = 2, 3, \dots, n$$

Como resulta obvio, $l_n = 1$. Luego, se genera un número aleatorio entre 0 y 1 y se selecciona el índice del intervalo al que pertenece el número aleatorio. Este índice indica qué horario debe seleccionarse.

Si, por ejemplo, el número aleatorio es x y el índice es i , se tiene:

$$x \in [0, 1[$$

$$i \in \{j = 1, 2, \dots, n \mid x \in I_j\}$$

Luego, H_i es el horario seleccionado.

El algoritmo se detiene cuando se alcanza el número máximo de generaciones, o cuando éste es finalizado.

Para construir la nueva población, este proceso de selección se repite n veces, donde n es el tamaño de la población. Puede que existan individuos que sean tomados en cuenta dos veces en el proceso, de ocurrir eso se confirma la fuerza del individuo para pasar a la siguiente población, incluso varias veces.

Una vez finalizado el algoritmo evolutivo, a la solución hallada se le aplica una reparación para asegurarse de que no hay mejoras triviales a la vista.

4 DESARROLLO DEL SISTEMA DE GENERACIÓN DE HORARIOS

4.1 INTRODUCCIÓN

En los capítulos anteriores se formuló el método a seguir para elaborar un horario. El presente capítulo muestra los pasos que se siguieron para llevar a cabo la realización del sistema informático SGHC que implementa el algoritmo evolutivo para elaborar el horario, respondiendo a las necesidades básicas que se presentan en los colegios.

El desarrollo de un software se divide en tres etapas bien definidas como son: el análisis, el diseño y la implementación. Cada una de estas será analizada en los subcapítulos que vienen a continuación.

En cada fase de desarrollo aparecen objetos, los cuales se identifican con nombres. Para los nombres de los objetos se utiliza únicamente caracteres del alfabeto inglés y el subrayado, debido a que en general los lenguajes de programación no permiten utilizar otros caracteres.

4.2 ANÁLISIS

4.2.1 INTRODUCCIÓN

El análisis es la primera fase de desarrollo en cualquier sistema. En éste se pretende describir la naturaleza del problema a resolver, fijar los objetivos por alcanzar, así como identificar y conceptualizar los objetos que intervienen en el sistema. Para satisfacer estos requisitos, se ha dividido este subcapítulo en dos partes. En la primera se hará una descripción sistemática de todo el problema. La segunda contendrá la identificación y conceptualización de los objetos del sistema, así como de los procesos.

4.2.2 DESCRIPCIÓN DEL PROBLEMA

Para llevar a cabo la elaboración de los horarios para colegio, se debe contar con cierta información mínima indispensable, que consiste en lo siguiente:

- Los días laborables de la semana, y de cada día las horas en que se dictan clases, lo que establece el horario laborable del colegio.
- Las aulas que existen, clasificadas y contabilizadas por tipos, de acuerdo a sus características físicas similares, como son la capacidad y equipamiento o la finalidad del aula.
- Los cursos existentes, cada uno con sus respectivos niveles y especializaciones, y de cada curso los paralelos con que se dispone.
- Las asignaturas que se dictan en el colegio, y de cada asignatura, el curso en donde se va a dictar, el tipo de aula que ésta requiere y la materia a la que corresponde.
- Las prohibiciones de materias, que es un horario en el cual se indica las horas del horario laborable del colegio en las que no sea adecuado dictar una materia por razones pedagógicas, así como también en las que sea físicamente imposible dictarla.
- Los profesores que trabajan en el colegio, y de cada profesor, se requiere conocer la carga académica, y basándose en el horario laborable del colegio deberá indicar las horas en las que no le agrada ejercer su carga académica y en las que él no puede laborar.
- La carga académica consiste en la distribución de los docentes en las asignaturas que deben dar en los paralelos.

Con estos datos, se procede a la realización del horario del colegio, para lo cual se ejecutan varias pruebas. Cada horario se registra con un número para identificarle, el momento en que

inicia y el momento en que finaliza la prueba, además de un informe con las características del horario.

El horario del colegio constituye de un conjunto de horarios por cada paralelo que exista en la institución. El horario de un paralelo contiene la distribución de las asignaturas en el horario laborable del colegio, respetando las restricciones dadas en 1.2.

En la elaboración del horario del colegio participa el algoritmo evolutivo descrito en los capítulos anteriores, como parte fundamental del sistema. Una vez que se ha elaborado el horario del colegio, éste puede ser modificado, para liberar un cruce o reflejar algún cambio posterior.

El horario se debe presentar de dos formas diferentes: Por paralelos para los estudiantes y por profesores para los mismos.

4.2.3 IDENTIFICACIÓN Y CONCEPTUALIZACIÓN DE LOS OBJETOS Y PROCESOS

Sujetos a la descripción del problema, procedemos a la conceptualización de cada uno de los objetos identificados, y los procesos que se llevan a cabo.

Objetos identificados:

Aula

Nombre descriptivo: Tipo de Aula

Las aulas se encuentran clasificadas en tipos de aula. De cada tipo de aula se debe conocer su nombre y la cantidad de aulas que lo conforman.

Curso

Nombre descriptivo: Curso

Un curso consiste de un nivel y una especialización académica. El nivel es una etapa de estudio. A los cursos que no tengan especialización académica, se les asignará una especialización denominada BÁSICO, y en la mayoría de colegios, los cursos básicos van del primer al tercer nivel.

Paralelo

Nombre descriptivo: Paralelo

Los cursos se dividen en paralelos. Cada paralelo tiene un identificador. Por ejemplo, el curso Cuarto Sociales tiene tres paralelos, que se identifican entre ellos con A, B y C.

HorarioLaborable

Nombre descriptivo: Horario laborable

El horario laborable está conformado por las horas académicas denominadas *períodos*, en las que se puede dictar clases.

Asignatura

Nombre descriptivo: Asignatura

La asignatura consiste de una materia que se dicta en un curso y en un aula de cierto tipo. Debe registrarse el número de horas semanales, y cómo éstas se distribuyen en la semana siguiendo criterios pedagógicos, por ejemplo: Matemáticas del Cuarto Secretariado se dicta en un aula normal, tiene una duración de cinco horas semanales, y conviene que se dicte en dos clases de dos horas y otra de una.

Materia

Nombre descriptivo: Materia

Una materia es una disciplina que se imparte en el colegio, de la cual nos interesa saber su nombre.

MateriaProhibicion

Nombre descriptivo: Prohibición de materia

Una prohibición de materia indica el período en que no debe dictarse por ser físicamente imposible, o que resulte inadecuado.

Profesor

Nombre descriptivo: Profesor

Un profesor es una persona que trabaja en el colegio, al cual se le asigna una carga académica que debe ejercerla. Del profesor se registra la cédula de identidad, el nombre y el apellido.

ProfesorProhibicion

Nombre descriptivo: Prohibición de profesor

Una prohibición de profesor indica el período en que no puede dictar clase por ser físicamente imposible, o que no le guste. Ejemplos: El profesor Pérez no puede dictar clase a la tercera hora del día Viernes porque hay reunión del área de Castellano, a la profesora Machado no le gusta dictar clase a la sexta hora del día Lunes.

CargaAcademica

Nombre descriptivo: Carga académica

La carga académica consiste en la distribución de los docentes en las asignaturas que deben dar en los paralelos.

Horario

Nombre descriptivo: Horario del colegio

Al horario del colegio se lo identifica con un código, además se registran los momentos en que inicia y finaliza su elaboración, y un informe con las características del horario.

HorarioDetalle

Nombre descriptivo: Detalle del horario

El detalle del horario para un horario del colegio, indica de cada paralelo los períodos en los que deben dictarse las asignaturas.

Procesos identificados

ElaborarHorario

Nombre descriptivo:Elaborar un horario

Este proceso permite generar un horario del colegio. Cuando se inicia, se pide un código para el horario, luego se lo elabora. Mientras la elaboración del horario está en progreso, ésta puede ser cancelada, en cuyo caso no se almacena el horario, o de lo contrario, se puede esperar a que termine normalmente la elaboración del horario o finalizarla manualmente cuando ha transcurrido mucho tiempo sin alcanzar el criterio de parada o que la solución actual no mejora, en estos dos casos se almacena el mejor horario elaborado al momento.

IntercambiarPeriodos

Nombre descriptivo: Intercambiar los períodos

Este proceso permite modificar un horario del colegio ya elaborado. Dado un horario del colegio, para un paralelo se seleccionan dos períodos, los que se intercambian, haciendo que las

materias asignadas a éstos se intercambien. Una vez realizada esta acción debe verificarse que las restricciones dadas en 1.2 se cumplan.

PresentarHorarioParalelo

Nombre descriptivo: Presentar los horarios por paralelo

Presenta los horarios de cada uno de los paralelos. Para mostrar el horario de un paralelo se elabora una tabla en la cual aparecen los días como columnas y las horas laborables como filas, y en la cuadrícula de la tabla se colocan las materias correspondientes.

PresentarHorarioProfesor

Nombre descriptivo: Presentar los horarios por profesor

Presenta para cada profesor, un horario con los paralelos, las materias a él asignadas y los períodos en que dicta clases. Para presentar esta información a un profesor se elabora una tabla en la cual aparecen los días como columnas y las horas laborables como filas, y en la cuadrícula se colocan los paralelos y las materias correspondientes.

4.3 DISEÑO

4.3.1 INTRODUCCIÓN

El presente subcapítulo detalla el diseño del sistema, el cual se encuentra dividido en cuatro partes.

La primera parte consiste en el modelo de datos, basado en el modelo entidad–relación. En la segunda se realiza el diseño del proceso, siguiendo como metodología el modelo funcional. La tercera parte resume las dos anteriores para mostrar el diccionario de datos. Finalmente, en la cuarta parte se definen los servicios ofrecidos por el sistema, lo que posteriormente permitirá realizar la implementación de la interfaz del usuario.

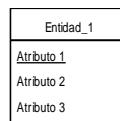
En el modelo de datos, los objetos identificados en el análisis pasarán a ser tablas, cuyos nombres serán los de los objetos asociados y entre las tablas se manejarán restricciones de integridad referencial.

4.3.2 MODELO DE DATOS

Nomenclatura utilizada

Esquemas gráficos

Para explicar los esquemas gráficos se muestran a continuación algunos ejemplos:



Representa una entidad denominada Entidad_1, formada por tres atributos denominados Atributo 1, Atributo 2 y Atributo 3. El Atributo 1 aparece subrayado para indicar que es parte de la clave primaria.



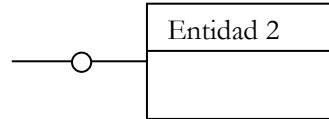
Representa una relación entre dos entidades. El nombre de la relación es Relacion_1. La funcionalidad de una relación, en este caso *Se corresponde con*, es una función que devuelve para un registro de la entidad 1, un conjunto de registros de la entidad 2.

La cardinalidad de la funcionalidad se establece por medio de un intervalo $[a, b]$, en donde a representa la cantidad mínima y b la cantidad máxima de registros de la entidad 2, que están en correspondencia funcional con cualquier registro de la entidad 1. Cuando $b > 1$ se dice que la relación es a muchos, y si $b = 1$, la relación es a uno.

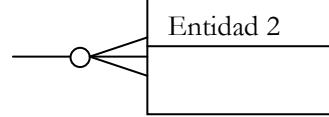
En las cardinalidades a definirse $[0, n]$ y $[1, n]$, los campos de la clave primaria de la entidad 1 pasan a ser atributos de la entidad 2, y se denominan campos foráneos.

La simbología que se utiliza para representar la cardinalidad es la siguiente:

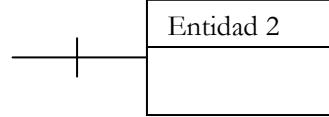
- Cardinalidad $[0, 1]$



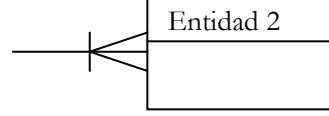
- Cardinalidad $[0, n]$



- Cardinalidad $[1, 1]$

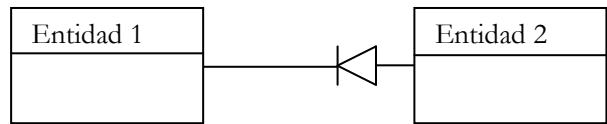


- Cardinalidad $[1, n]$

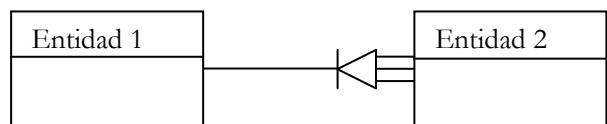


Cuando la entidad 2 es dependiente de la entidad 1, los campos de la clave primaria de la entidad 1 pasan a ser parte de la clave primaria de la entidad 2. A esto se lo denomina *relación de dependencia* y la forma de representarlo es la siguiente:

- Relación de dependencia con cardinalidad [1, 1]



- Relación de dependencia con cardinalidad $[1, n]$



Propiedades de los campos

- P
El campo es parte de la clave primaria.
 - U
El campo es único.
 - R
El campo es requerido.
 - F
El campo es foráneo.

Propiedades de los índices

- P
El índice es la clave primaria.
 - U
Los campos del índice son únicos.

Esquema

esquema = **NombreTabla**(campo1, campo2, ..., campoN)

Indica los campos que componen una tabla, además muestra subrayado los campos que forman la clave primaria.

Objetos de la base de datos

Tabla Nivel

Esquema = **Nivel**(CodNivel, NomNivel, AbrNivel)

Descripción de los campos

Nombre	Descripción	Tipo	Propiedades
CodNivel	Código del nivel	Entero	PUR
NomNivel	Nombre del nivel	Alfanumérico(15)	UR
AbrNivel	Abreviatura del nombre del nivel	Alfanumérico(5)	U

Indices

Nombre	Campos del índice	Propiedades
(clave primaria)	CodNivel	PU
IxNomNivel	NomNivel	U
IxAbrNivel	AbrNivel	U

Tabla Especializacion

Esquema = **Especializacion**(CodEspecializacion, NomEspecializacion, AbrEspecializacion)

Descripción de los campos

Nombre	Descripción	Tipo	Propiedades
CodEspecializacion	Código de la especialización	Entero	PUR
NomEspecializacion	Nombre de la especialización	Alfanumérico(20)	UR
AbrEspecializacion	Abreviatura del nombre de la especialización	Alfanumérico(10)	UR

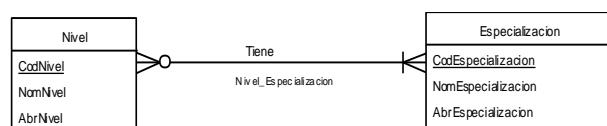
Indices

Nombre	Campos del índice	Propiedades
(clave primaria)	CodEspecializacion	PU
IxAbrEspecializacion	AbrEspecializacion	U
IxNomEspecializacion	NomEspecializacion	U

Tabla Curso

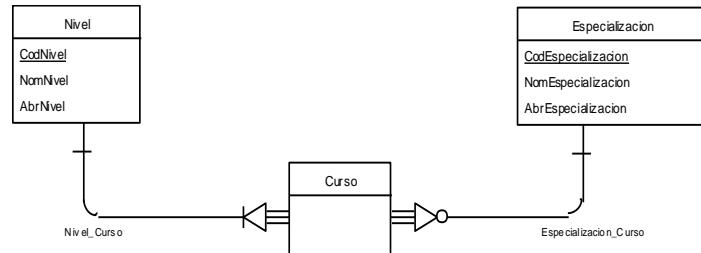
Esquema = **Curso**(CodNivel, CodEspecializacion)

Entre las tablas **Nivel** y **Especializacion** se tiene la relación denominada Nivel_Especializacion:



Un nivel debe al menos tener una especialización, sin embargo, puede ser que una especialización no esté asociada a un nivel.

La funcionalidad de esta relación tiene una cardinalidad muchos a muchos, lo que lleva a su descomposición en la tabla denominada **Curso** y las siguientes relaciones: Nivel_Curso y Especializacion_Curso, como se describe a continuación:



La funcionalidad de la relación Especializacion_Curso tiene una cardinalidad uno a muchos, lo que obliga a que los campos de la clave de **Especialización** bajen como campos foráneos a la tabla **Curso**, por lo tanto:

$$\text{Curso.CodEspecialización} = \text{Especialización.CodEspecialización}$$

La funcionalidad de la relación Nivel_Curso tiene una cardinalidad uno a muchos, lo que obliga a que los campos de la clave de **Nivel** bajen como campos foráneos a la tabla **Curso**, por lo tanto:

$$\text{Curso.CodNivel} = \text{Nivel.CodNivel}$$

Descripción de los campos

Nombre	Descripción	Tipo	Propiedades
CodNivel	Ver Nivel.CodNivel	Entero	PRF
CodEspecialización	Ver Especialización.CodEspecialización	Entero	PRF

Indices

Nombre	Campos del índice	Propiedades
(clave primaria)	CodNivel;CodEspecialización	PU
EspecializaciónCurso	CodEspecialización	
NivelCurso	CodNivel	

Tabla ParaleloId

Esquema = **ParaleloId**(CodParaleloId, NomParaleloId)

La tabla **ParaleloId** contiene cada uno de los identificantes que se usan en el colegio para los paralelos de un determinado curso. Por ejemplo: En un colegio los identificantes de paralelo pueden ser “A”, “B”, “C”, “D” y “S”.

Descripción de los campos

Nombre	Descripción	Tipo	Propiedades
CodParaleloId	<i>Código</i> del identificador de paralelo	Entero	PUR
NomParaleloId	<i>Nombre</i> del identificador de paralelo	Alfanumérico(5)	UR

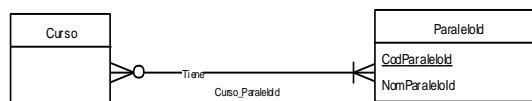
Indices

Nombre	Campos del índice	Propiedades
(clave primaria)	CodParaleloId	PU
IxNomParaleloId	NomParaleloId	U

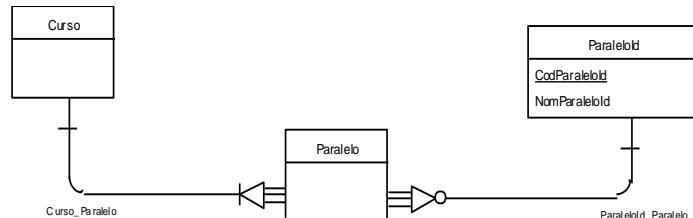
Tabla Paralelo

Esquema = **Paralelo**(CodNivel, CodEspecializacion, CodParaleloId)

Entre las tablas **Curso** y **ParaleloId** se tiene la relación denominada **Curso_ParaleloId**:



La funcionalidad de esta relación tiene una cardinalidad muchos a muchos, lo que lleva a su descomposición en la tabla denominada **Paralelo** y las siguientes relaciones: **Curso_Paralelo** y **ParaleloId_Paralelo**, como se describe a continuación:



La funcionalidad de la relación **ParaleloId_Paralelo** tiene una cardinalidad uno a muchos, lo que obliga a que los campos de la clave de **ParaleloId** bajen como campos foráneos a la tabla **Paralelo**, por lo tanto:

$$\text{Paralelo.CodParaleloId} = \text{ParaleloId.CodParaleloId}$$

La funcionalidad de la relación **Curso_Paralelo** tiene una cardinalidad uno a muchos, lo que obliga a que los campos de la clave de **Curso** bajen como campos foráneos a la tabla **Paralelo**, por lo tanto:

$$\text{Paralelo.CodNivel} = \text{Curso.CodNivel}$$

$$\text{Paralelo.CodEspecializacion} = \text{Curso.CodEspecializacion}$$

Descripción de los campos

Nombre	Descripción	Tipo	Propiedades
CodNivel	Ver Nivel.CodNivel	Entero	PRF
CodEspecializacion	Ver Especializacion.CodEspecializacion	Entero	PRF
CodParaleloId	Ver ParaleloId.CodParaleloId	Entero	PRF

Indices

Nombre	Campos del índice	Propiedades
(clave primaria)	CodNivel;CodEspecializacion;CodParaleloId	PU
CursoParalelo	CodNivel;CodEspecializacion	
TipoParaleloParalelo	CodParaleloId	

Tabla Dia

Esquema = **Dia**(CodDia, NomDia)

Descripción de los campos

Nombre	Descripción	Tipo	Propiedades
CodDia	Código del día	Entero	PUR
NomDia	Nombre del día	Alfanumérico(10)	UR

Indices

Nombre	Campos del índice	Propiedades
(clave primaria)	CodDia	PU
IxNomDia	NomDia	U

Tabla Hora

Esquema = **Hora**(CodHora, NomHora, Intervalo)

Descripción de los campos

Nombre	Descripción	Tipo	Propiedades
CodHora	Código de la hora	Entero	PUR
NomHora	Nombre de la hora	Alfanumérico(10)	UR
Intervalo	Intervalo de la hora	Alfanumérico(21)	UR

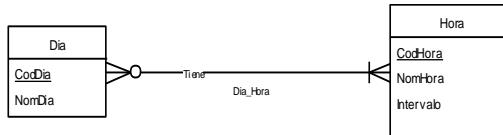
Indices

Nombre	Campos del índice	Propiedades
(clave primaria)	CodHora	PU
IxIntervalo	Intervalo	U
IxNomHora	NomHora	U

Tabla HorarioLaborable

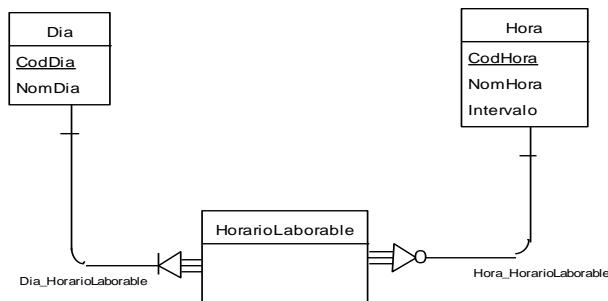
Esquema = **HorarioLaborable**(CodDia, CodHora)

Entre las tablas **Dia** y **Hora** se tiene la relación denominada Dia_Hora:



Una hora puede no ser hora laborable en ningún día, por ejemplo: las horas de descanso, pero en cada día deben registrarse las horas laborables únicamente.

La funcionalidad de esta relación tiene una cardinalidad muchos a muchos, lo que lleva a su descomposición en la tabla denominada **HorarioLaborable** y las siguientes relaciones: Dia_HorarioLaborable y Hora_HorarioLaborable, como se describe a continuación:



La funcionalidad de la relación Hora_HorarioLaborable tiene una cardinalidad uno a muchos, lo que obliga a que los campos de la clave de **Hora** bajen como campos foráneos a la tabla **HorarioLaborable**, por lo tanto:

$$\text{HorarioLaborable.CodHora} = \text{Hora.CodHora}$$

La funcionalidad de la relación Dia_HorarioLaborable tiene una cardinalidad uno a muchos, lo que obliga a que los campos de la clave de **Dia** bajen como campos foráneos a la tabla **HorarioLaborable**, por lo tanto:

$$\text{HorarioLaborable.CodDia} = \text{Dia.CodDia}$$

Descripción de los campos

Nombre	Descripción	Tipo	Propiedades
CodDia	Ver Dia.CodDia	Entero	PRF
CodHora	Ver Hora.CodHora	Entero	PRF

Indices

Nombre	Campos del índice	Propiedades
(clave primaria)	CodDia;CodHora	PU
DiaHorarioLaborable	CodDia	
HoraHorarioLaborable	CodHora	

Tabla Materia

Esquema = **Materia**(CodMateria, NomMateria)

Descripción de los campos

Nombre	Descripción	Tipo	Propiedades
CodMateria	<i>Código</i> de la materia	Entero	PUR
NomMateria	<i>Nombre</i> de la materia	Alfanumérico(20)	UR

Indices

Nombre	Campos del índice	Propiedades
(clave primaria)	CodMateria	PU
IxNomMateria	NomMateria	U

Tabla MateriaProhibicionTipo

Esquema = **MateriaProhibicionTipo**(CodMateProhibicionTipo, NomMateProhibicionTipo, ColMateProhibicionTipo, ValMateProhibicionTipo)

La tabla MateriaProhibicionTipo contiene cada uno de los tipos de prohibición de materia existentes. Los tipos de prohibición permitidos en el presente sistema son dos: *Inadecuado* e *Imposible*. Cada prohibición tiene un color para diferenciarlo y una ponderación.

Descripción de los campos

Nombre	Descripción	Tipo	Propiedades
CodMateProhibicionTipo	<i>Código</i> del tipo de prohibición de materia	Entero	PUR
NomMateProhibicionTipo	<i>Nombre</i> del tipo de prohibición de materia	Alfanumérico(10)	UR
ColMateProhibicionTipo	<i>Color</i> que ayuda a identificar el tipo de prohibición de materia	Entero	R
ValMateProhibicionTipo	<i>Valor</i> (ponderación) que se le da al tipo de prohibición de materia	Flotante	R

Indices

Nombre	Campos del índice	Propiedades
(clave primaria)	CodMateProhibicionTipo	PU
IxNomMateProhibicionTipo	NomMateProhibicionTipo	U

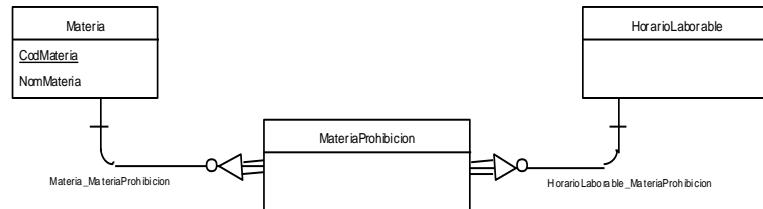
Tabla MateriaProhibicion

Esquema = **MateriaProhibicion**(CodMateria, CodDia, CodHora, CodMateProhibicionTipo)

Entre las tablas **Materia** y **HorarioLaborable** se tiene la relación denominada Materia_HorarioLaborable:



La funcionalidad de esta relación tiene una cardinalidad muchos a muchos, lo que lleva a su descomposición en la tabla denominada **MateriaProhibicion** y las siguientes relaciones: Materia_MateriaProhibicion y HorarioLaborable_MateriaProhibicion:



La funcionalidad de la relación Materia_MateriaProhibicion tiene una cardinalidad uno a muchos, lo que obliga a que los campos de la clave de **Materia** bajen como campos foráneos a la tabla **MateriaProhibicion**, por lo tanto:

$$\text{MateriaProhibicion.CodMateria} = \text{Materia.CodMateria}$$

La funcionalidad de la relación HorarioLaborable_MateriaProhibicion tiene una cardinalidad uno a muchos, lo que obliga a que los campos de la clave de **HorarioLaborable** bajen como campos foráneos a la tabla **MateriaProhibicion**, por lo tanto:

$$\text{MateriaProhibicion.CodDia} = \text{HorarioLaborable.CodDia}$$

$$\text{MateriaProhibicion.CodHora} = \text{HorarioLaborable.CodHora}$$

Entre las tablas **MateriaProhibicion** y **MateriaProhibicionTipo** se tiene la relación denominada MateriaProhibicionTipo_MateriaProhibicion:



Una prohibición de materia debe necesariamente ser de un tipo de prohibición de materia.

La funcionalidad de la relación MateriaProhibicionTipo_MateriaProhibicion tiene una cardinalidad uno a muchos, lo que obliga a que los campos de la clave de **MateriaProhibicionTipo** bajen como campos foráneos a la tabla **MateriaProhibicion**, por lo tanto:

$$\begin{aligned} \text{MateriaProhibicion.CodMaterProhibicionTipo} = \\ \text{MateriaProhibicionTipo.CodMaterProhibicionTipo} \end{aligned}$$

Descripción de los campos

Nombre	Descripción	Tipo	Propiedades
CodMateria	Ver Materia.CodMateria	Entero	PRF
CodDia	Ver Dia.CodDia	Entero	PRF
CodHora	Ver Hora.CodHora	Entero	PRF
CodMaterProhibicionTipo	Ver	Entero	RF

	MateriaProhibicionTipo.CodMat eProhibicionTipo		
--	---	--	--

Indices

Nombre	Campos del índice	Propiedades
(clave primaria)	CodMateria;CodDia;CodHora	PU
HorarioLaborableMateriaPr	CodDia;CodHora	
MateriaMateriaProhibicion	CodMateria	
TipoProhibicionMateriaPro	CodMaterProhibicionTipo	

Tabla AulaTipo

Esquema = **AulaTipo**(CodAulaTipo, NomAulaTipo, AbrAulaTipo, Cantidad)

Descripción de los campos

Nombre	Descripción	Tipo	Propiedades
CodAulaTipo	Código del tipo de aula	Entero	PUR
NomAulaTipo	Nombre del tipo de aula	Alfanumérico(25)	UR
AbrAulaTipo	Abreviatura del tipo de aula	Alfanumérico(10)	UR
Cantidad	Cantidad de aulas del tipo	Entero	R

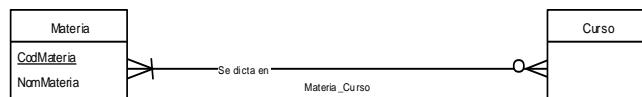
Indices

Nombre	Campos del índice	Propiedades
(clave primaria)	CodAulaTipo	PU
IxAbrAulaTipo	AbrAulaTipo	U
IxNomAulaTipo	NomAulaTipo	U

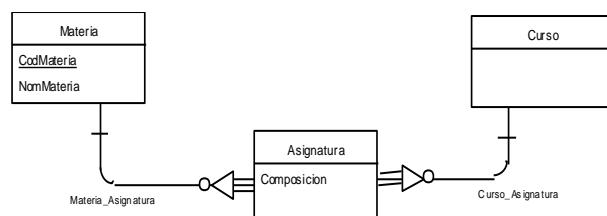
Tabla Asignatura

Esquema = **Asignatura**(CodMateria, CodNivel, CodEspecializacion, CodAulaTipo, Composicion)

Entre las tablas **Curso** y **Materia** se tiene la relación denominada **Materia_Curso**:



La funcionalidad de esta relación tiene una cardinalidad muchos a muchos, lo que lleva a su descomposición en la tabla denominada **Asignatura** y las siguientes relaciones: **Materia_Asignatura** y **Curso_Asignatura**:



La tabla **Asignatura** tiene un atributo propio denominado *Composicion*, el cual contiene en un formato codificado la duración de la asignatura y cómo esta se distribuye en la semana.

La funcionalidad de la relación Materia_Asignatura tiene una cardinalidad uno a muchos, lo que obliga a que los campos de la clave de **Materia** bajen como campos foráneos a la tabla **Asignatura**, por lo tanto:

Asignatura.CodMateria = Materia.CodMateria

La funcionalidad de la relación Curso_Asignatura tiene una cardinalidad uno a muchos, lo que obliga a que los campos de la clave de **Curso** bajen como campos foráneos a la tabla **Asignatura**, por lo tanto:

Asignatura.CodNivel = Curso.CodNivel

Asignatura.CodEspecializacion = Curso.CodEspecializacion

Entre las tablas **Asignatura** y **AulaTipo** se tiene la relación denominada AulaTipo_Asignatura, como se describe a continuación:



La funcionalidad de la relación AulaTipo_Asignatura tiene una cardinalidad uno a muchos, lo que obliga a que los campos de la clave de **AulaTipo** bajen como campos foráneos a la tabla **Asignatura**, por lo tanto:

Asignatura.CodAulaTipo = AulaTipo.CodAulaTipo

Descripción de los campos

Nombre	Descripción	Tipo	Propiedades
CodMateria	Ver Materia.CodMateria	Entero	PRF
CodNivel	Ver Nivel.CodNivel	Entero	PRF
CodEspecializacion	Ver Especializacion.CodEspecializacion	Entero	PRF
CodAulaTipo	Ver AulaTipo.CodAulaTipo	Entero	RF
Composicion	Composición de la asignatura	Alfanumérico(20)	R

Indices

Nombre	Campos del índice	Propiedades
(clave primaria)	CodMateria;CodNivel;CodEspecializacion	PU
AulaTipoAsignatura	CodAulaTipo	
CursoAsignatura	CodNivel;CodEspecializacion	
MateriaAsignatura	CodMateria	

Tabla Profesor

Esquema = **Profesor**(CodProfesor, CedProfesor, ApeProfesor, NomProfesor)

Descripción de los campos

Nombre	Descripción	Tipo	Propiedades
CodProfesor	<i>Código</i> del profesor	Entero	PUR
CedProfesor	<i>Cédula</i> del profesor	Alfanumérico(11)	UR
ApeProfesor	<i>Apellido</i> del profesor	Alfanumérico(15)	R
NomProfesor	<i>Nombre</i> del profesor	Alfanumérico(15)	R

Indices

Nombre	Campos del índice	Propiedades
(clave primaria)	CodProfesor	PU
IxApeNomProfesor	ApeProfesor;NomProfesor	U
IxCedProfesor	CedProfesor	U
IxNomApeProfesor	NomProfesor;ApeProfesor	U

Tabla ProfesorProhibicionTipo

Esquema = **ProfesorProhibicionTipo**(CodProfProhibicionTipo, NomProfProhibicionTipo, ColProfProhibicionTipo, ValProfProhibicionTipo)

La tabla **ProfesorProhibicionTipo** contiene cada uno de los tipos de prohibición de profesor existentes. Los tipos de prohibición permitidos en el presente sistema son dos: *No puede* y *No le gustaría*. Cada prohibición tiene un color para diferenciarlo y una ponderación.

Descripción de los campos

Nombre	Descripción	Tipo	Propiedades
CodProfProhibicionTipo	<i>Código</i> del tipo de prohibición de profesor	Entero	PUR
NomProfProhibicionTipo	<i>Nombre</i> del tipo de prohibición de profesor	Alfanumérico(10)	UR
ColProfProhibicionTipo	<i>Color</i> que ayuda a identificar el tipo de prohibición de profesor	Entero	R
ValProfProhibicionTipo	<i>Valor</i> (ponderación) que se asigna al tipo de prohibición de profesor	Flotante	R

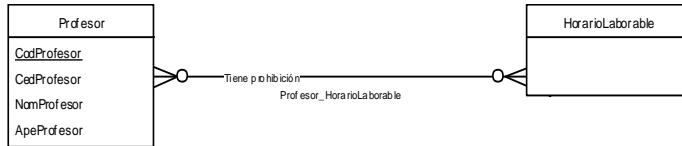
Indices

Nombre	Campos del índice	Propiedades
(clave primaria)	CodProfProhibicionTipo	PU
IxNombre	NomProfProhibicionTipo	U

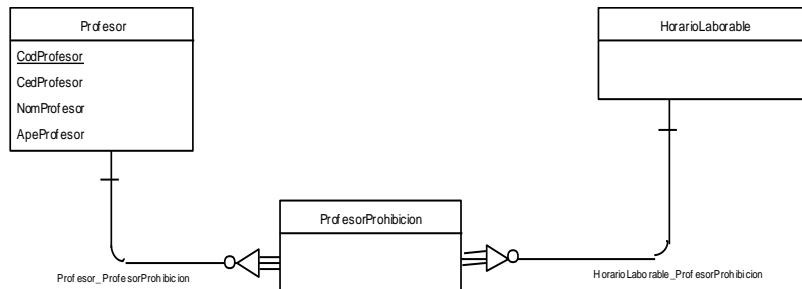
Tabla ProfesorProhibicion

Esquema = **ProfesorProhibicion**(CodProfesor, CodDia, CodHora, CodProfProhibicionTipo)

Entre las tablas **Profesor** y **HorarioLaborable** se tiene la relación denominada Profesor_HorarioLaborable:



La funcionalidad de esta relación tiene una cardinalidad muchos a muchos, lo que lleva a su descomposición en la tabla denominada **ProfesorProhibicion** y las siguientes relaciones: Profesor_ProfesorProhibicion y HorarioLaborable_ProfesorProhibicion:



La funcionalidad de la relación HorarioLaborable_ProfesorProhibicion tiene una cardinalidad uno a muchos, lo que obliga a que los campos de la clave de **HorarioLaborable** bajen como campos foráneos a la tabla **ProfesorProhibicion**, por lo tanto:

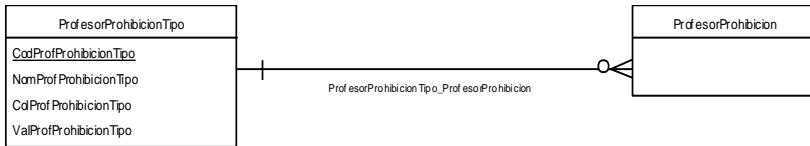
ProfesorProhibicion.CodDia = HorarioLaborable.CodDia

ProfesorProhibicion.CodHora = HorarioLaborable.CodHora

La funcionalidad de la relación Profesor_ProfesorProhibicion tiene una cardinalidad uno a muchos, lo que obliga a que los campos de la clave de **Profesor** bajen como campos foráneos a la tabla **ProfesorProhibicion**, por lo tanto:

ProfesorProhibicion.CodProfesor = Profesor.CodProfesor

Entre las tablas **ProfesorProhibicion** y **ProfesorProhibicionTipo** se tiene la relación denominada ProfesorProhibicionTipo_ProfesorProhibicion:



La funcionalidad de la relación ProfesorProhibicionTipo_ProfesorProhibicion tiene una cardinalidad uno a muchos, lo que obliga a que los campos de la clave de **ProfesorProhibicionTipo** bajen como campos foráneos a la tabla **ProfesorProhibicion**, por lo tanto:

ProfesorProhibicion.CodProfProhibicionTipo =

ProfesorProhibicionTipo.CodProfProhibicionTipo

Descripción de los campos

Nombre	Descripción	Tipo	Propiedades
CodProfesor	Ver Profesor.CodProfesor	Entero	PRF
CodDia	Ver Dia.CodDia	Entero	PRF
CodHora	Ver Hora.CodHora	Entero	PRF
CodProfProhibicionTipo	Ver ProfesorProhibicionTipo.CodProfProhibicionTipo	Entero	RF

Indices

Nombre	Campos del índice	Propiedades
(clave primaria)	CodProfesor;CodDia;CodHora	PU
HorarioLaborableProfesorP	CodDia;CodHora	
ProfesorProfesorProhibici	CodProfesor	
TipoProhibicionProfesorPr	CodProfProhibicionTipo	

Tabla CargaAcademica

Esquema = **CargaAcademica**(CodMateria, CodNivel, CodEspecializacion, CodParaleloId, CodProfesor)

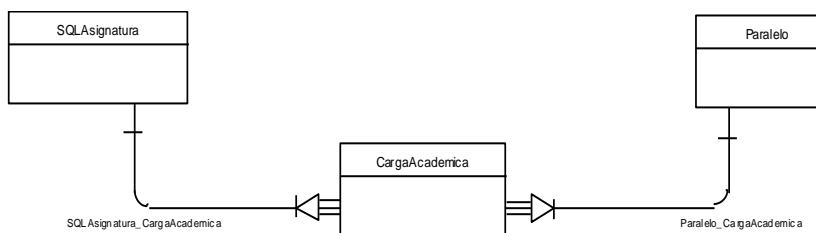
Dado un CodNivel, y un CodEspecialización, la consulta **SQLAsignatura** selecciona de la tabla **Asignatura** los registros que contengan estos dos códigos como parte de su clave primaria, y de estos se extrae el campo CodMateria, los cuales pasan a ser registros de la consulta. La sentencia de la consulta es la siguiente:

```
SELECT Asignatura.CodMateria
FROM Asignatura
WHERE
Asignatura.CodNivel=:CodNivel AND Asignatura.CodEspecializacion=:CodEspecializacion
```

Entre la tabla **Paralelo** y la consulta **SQLAsignatura** se tiene la relación denominada **Paralelo_SQLAsignatura**:



La funcionalidad de esta relación tiene una cardinalidad muchos a muchos, lo que lleva a su descomposición en la tabla denominada **CargaAcademica** y las siguientes relaciones: **Paralelo_CargaAcademica** y **SQLAsignatura_CargaAcademica**:



La funcionalidad de la relación **Paralelo_CargaAcademica** tiene una cardinalidad uno a muchos, lo que obliga a que los campos de la clave de **Paralelo** bajen como campos foráneos a la tabla **CargaAcademica**, por lo tanto:

CargaAcademica.CodNivel = Paralelo.CodNivel

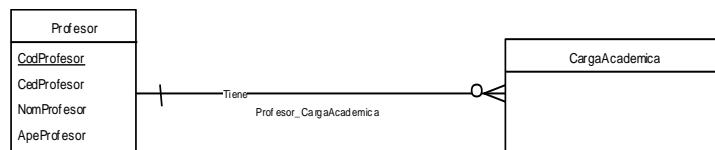
CargaAcademica.CodEspecializacion = Paralelo.CodEspecializacion

CargaAcademica.CodParaleloId = Paralelo.CodParaleloId

La funcionalidad de la relación **SQLAsignatura_CargaAcademica** tiene una cardinalidad uno a muchos, lo que obliga a que los campos de la clave de **SQLAsignatura** bajen como campos foráneos a la tabla **CargaAcademica**, por lo tanto:

CargaAcademica.CodMateria = SQLAsignatura.CodMateria

Entre las tablas **CargaAcademica** y **Profesor** se tiene la relación denominada Profesor_CargaAcademica:



La funcionalidad de la relación Profesor_CargaAcademica tiene una cardinalidad uno a muchos, lo que obliga a que los campos de la clave de **Profesor** bajen como campos foráneos a la tabla **CargaAcademica**, por lo tanto:

CargaAcademica.CodProfesor = Profesor.CodProfesor

Observación: La tabla **CargaAcademica** tiene al campo CodMateria proveniente de la consulta **SQLAsignatura**, por lo tanto, para cualquier cambio en la tabla **Asignatura** la aplicación debe controlar que la tabla **CargaAcademica** mantenga su integridad.

Descripción de los campos

Nombre	Descripción	Tipo	Propiedades
CodMateria	Ver Materia.CodMateria	Entero	PRF
CodNivel	Ver Nivel.CodNivel	Entero	PRF
CodEspecializacion	Ver Especializacion.CodEspecializacion	Entero	PRF
CodParaleloId	Ver ParaleloId.CodParaleloId	Entero	PRF
CodProfesor	Ver Profesor.CodProfesor	Entero	RF

Indices

Nombre	Campos del índice	Propiedades
(clave primaria)	CodMateria;CodNivel;CodEspecializacion;CodParaleloId	PU
CargaAcademicaNivel	CodNivel	
AsignaturaCargaAcademica	CodMateria;CodNivel;CodEspecializacion	
ParaleloCargaAcademica	CodNivel;CodEspecializacion;CodParaleloId	
ProfesorCargaAcademica	CodProfesor	

Tabla Horario

Esquema = **Horario**(CodHorario, MomentoInicial, MomentoFinal, Informe)

Descripción de los campos

Nombre	Descripción	Tipo	Propiedades
CodHorario	Código del horario	Entero	PUR
MomentoInicial	Momento Inicial en que se elaboró el horario	Fecha Hora	R
MomentoFinal	Momento Final en que se elaboró el horario	Fecha Hora	R
Informe	Informe de las características del horario	Memo	

Indices

Nombre	Campos del índice	Propiedades
(clave primaria)	CodHorario	PU
IxMomentoFinal	MomentoFinal	
IxMomentoInicial	MomentoInicial	

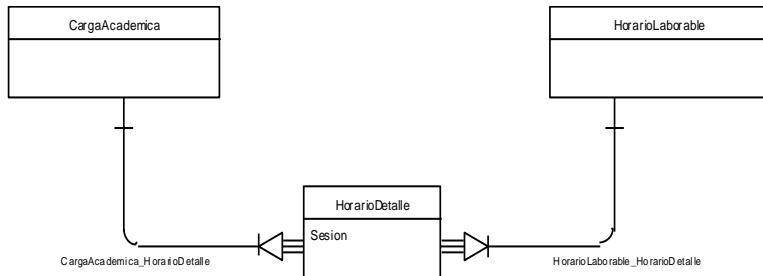
Tabla HorarioDetalle

Esquema = **HorarioDetalle**(CodHorario, CodNivel, CodEspecializacion, CodParaleloId, CodDia, CodHora, CodMateria, Sesion)

Entre las tablas **CargaAcademica** y **HorarioLaborable** se tiene la relación denominada CargaAcademica_HorarioLaborable:



La funcionalidad de esta relación tiene una cardinalidad muchos a muchos, lo que lleva a su descomposición en la tabla denominada **HorarioDetalle** y las siguientes relaciones: CargaAcademica_HorarioDetalle y HorarioLaborable_HorarioDetalle:



La tabla **HorarioDetalle** tiene un atributo propio denominado *Sesion*, el cual es un número de uso interno utilizado por el algoritmo evolutivo, que se almacena en la base de datos para poder recuperar la información de las características del horario.

La funcionalidad de la relación **CargaAcademica_HorarioDetalle** tiene una cardinalidad uno a muchos, lo que obliga a que los campos de la clave de **CargaAcademica** bajen como campos foráneos a la tabla **HorarioDetalle**, por lo tanto:

HorarioDetalle.CodMateria = CargaAcademica.CodMateria
 HorarioDetalle.CodNivel = CargaAcademica.CodNivel
 HorarioDetalle.CodEspecializacion = CargaAcademica.CodEspecializacion
 HorarioDetalle.CodParaleloId = CargaAcademica.CodParaleloId

La funcionalidad de la relación **HorarioLaborable_HorarioDetalle** tiene una cardinalidad uno a muchos, lo que obliga a que los campos de la clave de **HorarioLaborable** bajen como campos foráneos a la tabla **HorarioDetalle**, por lo tanto:

HorarioDetalle.CodDia = HorarioLaborable.CodDia
 HorarioDetalle.CodHora = HorarioLaborable.CodHora

Entre las tablas **HorarioDetalle** y **Horario** se tiene la relación denominada **Horario_HorarioDetalle**:



La funcionalidad de la relación **Horario_HorarioDetalle** tiene una cardinalidad uno a muchos, lo que obliga a que los campos de la clave de **Horario** bajen como campos foráneos a la tabla **HorarioDetalle**, por lo tanto:

HorarioDetalle.CodHorario = Horario.CodHorario

Es necesario aclarar que por ser imposible dictar dos materias en un paralelo durante el mismo período, el campo **CodMateria** no es parte de la clave primaria de **HorarioDetalle**.

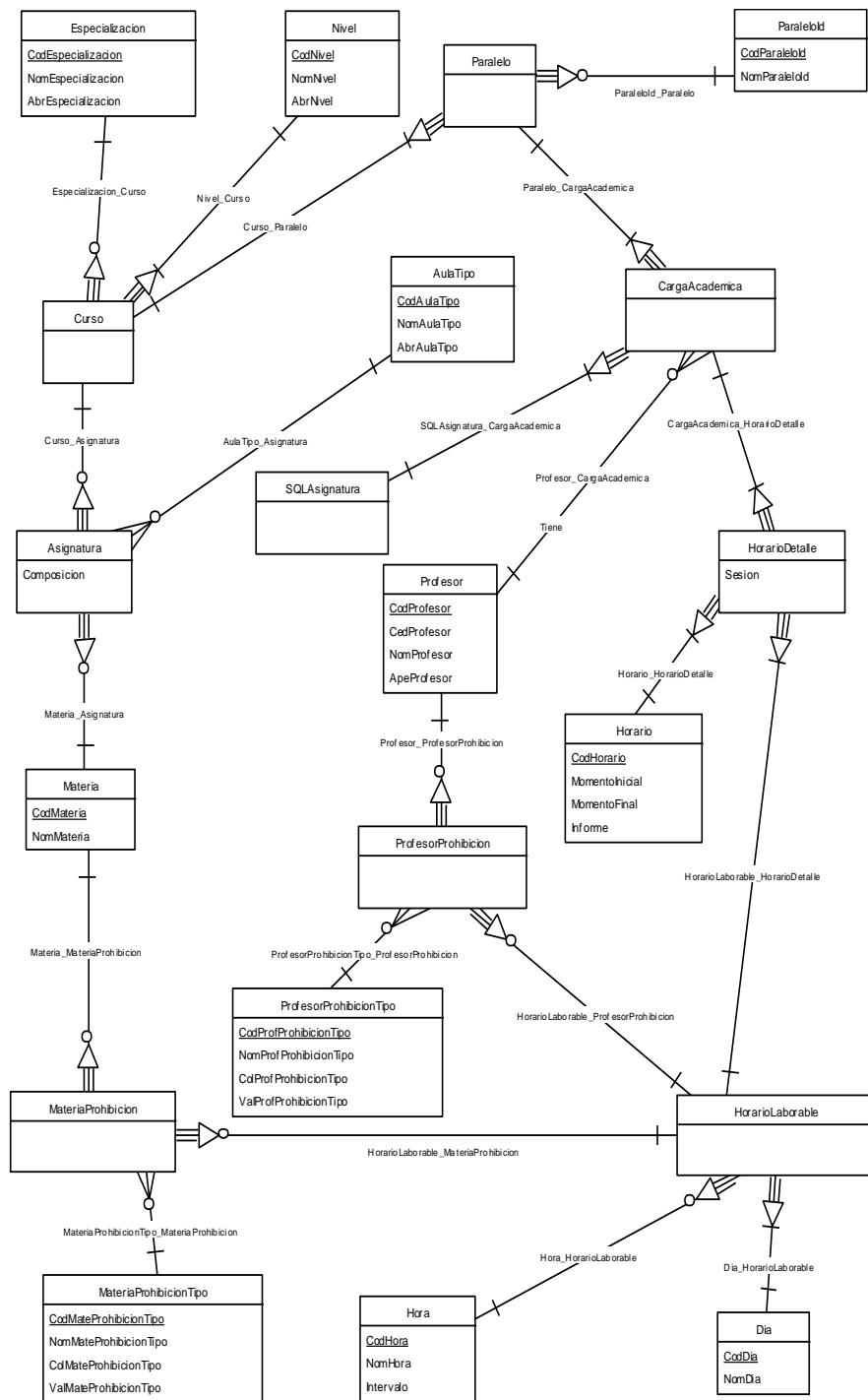
Descripción de los campos

Nombre	Descripción	Tipo	Propiedades
CodHorario	Ver Horario.CodHorario	Entero	PRF
CodNivel	Ver Nivel.CodNivel	Entero	PRF
CodEspecializacion	Ver Especializacion.CodEspecializacion	Entero	PRF
CodParaleloId	Ver ParaleloId.CodParaleloId	Entero	PRF
CodDia	Ver Dia.CodDia	Entero	PRF
CodHora	Ver Hora.CodHora	Entero	PRF
CodMateria	Ver Materia.CodMateria	Entero	RF
Sesion	<i>Sesión</i> del detalle del horario	Entero	R

Indices

Nombre	Campos del índice	Propiedades
(clave primaria)	CodHorario;CodNivel;CodEspecializacion;CodParaleloId;CodDia;CodHora	PU
CargaAcademicaHorarioDeta	CodMateria;CodNivel;CodEspecializacion;CodParaleloId	
HorarioHorarioDetalle	CodHorario	
HorarioLaborableHorarioDe	CodDia;CodHora	

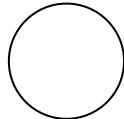
Modelo Relacional del SGHC



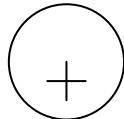
4.3.3 DISEÑO DEL PROCESO

Nomenclatura utilizada

Esquemas gráficos



- *Proceso.* Un proceso es una *caja negra* que recibe datos, los transforma y produce resultados.



- *Descomposición de proceso.* Ruptura del proceso en algunos niveles jerárquicos inferiores de procesamiento.



- *Entidad Externa.* Fuente o destino de los datos usados en el modelo.



- *Almacén de datos.* Lugar donde los datos son almacenados, ya sea temporal o permanentemente en el sistema.



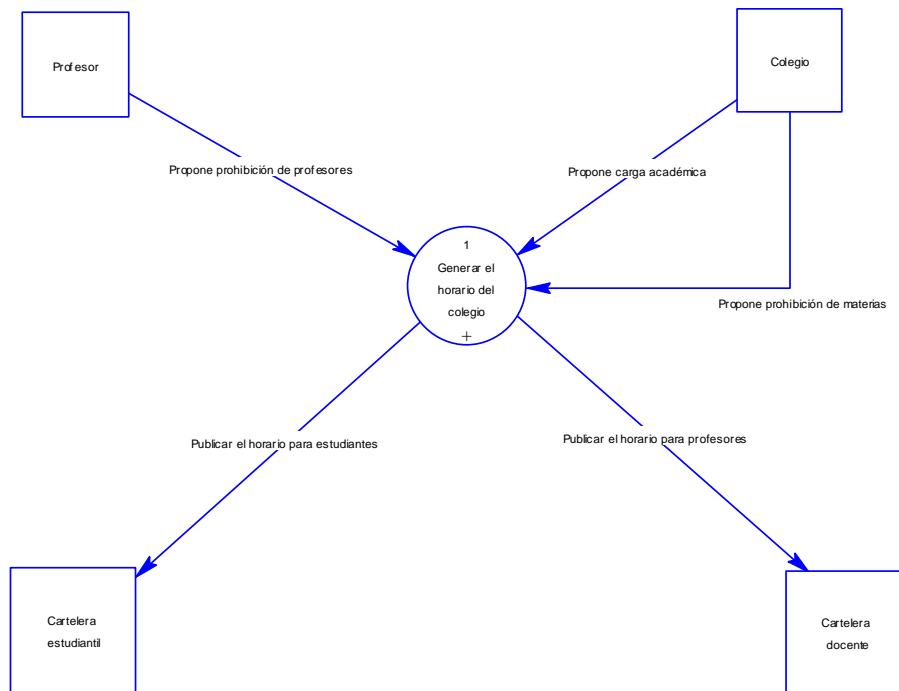
- *Flujo de datos.* Transferencia de datos entre dos componentes del sistema.

Proceso principal del SGHC

Descripción:

El proceso *Generar el horario del colegio* recibe de los profesores el horario de prohibiciones, y del Colegio la carga académica y la prohibición de materias, con esta información se procede a generar el horario. Una vez finalizada la generación, se publica el horario para los estudiantes en la cartelera estudiantil y para los profesores en la cartelera docente.

Gráfico:



Proceso 1: Generar el horario del colegio.

Nombre:

GenerarHorarioColegio

Descripción:

El profesor propone su horario de prohibiciones y en caso de ser aceptado se lo almacena en *Datos del colegio*. El colegio propone la prohibición de materias y la carga académica, que una vez aceptadas pasan a *Datos del colegio*.

Los datos presentes en *Datos del colegio* se envían al proceso *Elaborar un horario*, el cual almacena el horario resultante en *Horarios encontrados*.

Para alterar un horario ya encontrado, el colegio envía los períodos a intercambiar en un horario determinado al proceso *Intercambiar los períodos*, éste a su vez recoge del almacén *Horarios encontrados* el horario especificado. Procede a modificar este horario y lo sustituye por el anterior en el almacén *Horarios encontrados*.

Del almacén *Horarios Encontrados* se selecciona un horario que se guarda en *Horario seleccionado* y se envía para su publicación a dos lugares diferentes, por medio de los procesos *Presentar los horarios por paralelo* que da el horario para estudiantes y *Presentar los horarios por profesor* que da el horario para profesores.

El horario por paralelos se publica en la cartelera estudiantil, y el horario por profesores en la cartelera docente.

Argumentos de entrada:

Documentos que contienen los horarios de prohibiciones que suministran los profesores, las prohibiciones de materias y la carga académica que da el colegio.

Salida del proceso:

Dos reportes que contienen los horarios generados, tanto para docentes como para los estudiantes.

Algoritmo:

Ingreso de datos:

- 1.1 Aceptar prohibición de profesores
- 1.2 Aceptar prohibición de materias
- 1.3 Aceptar carga académica

Una vez ingresado los datos hacer:

- 1.4 Elaborar un horario
- Si el colegio desea modificar el horario Entonces
- 1.5 Intercambiar períodos

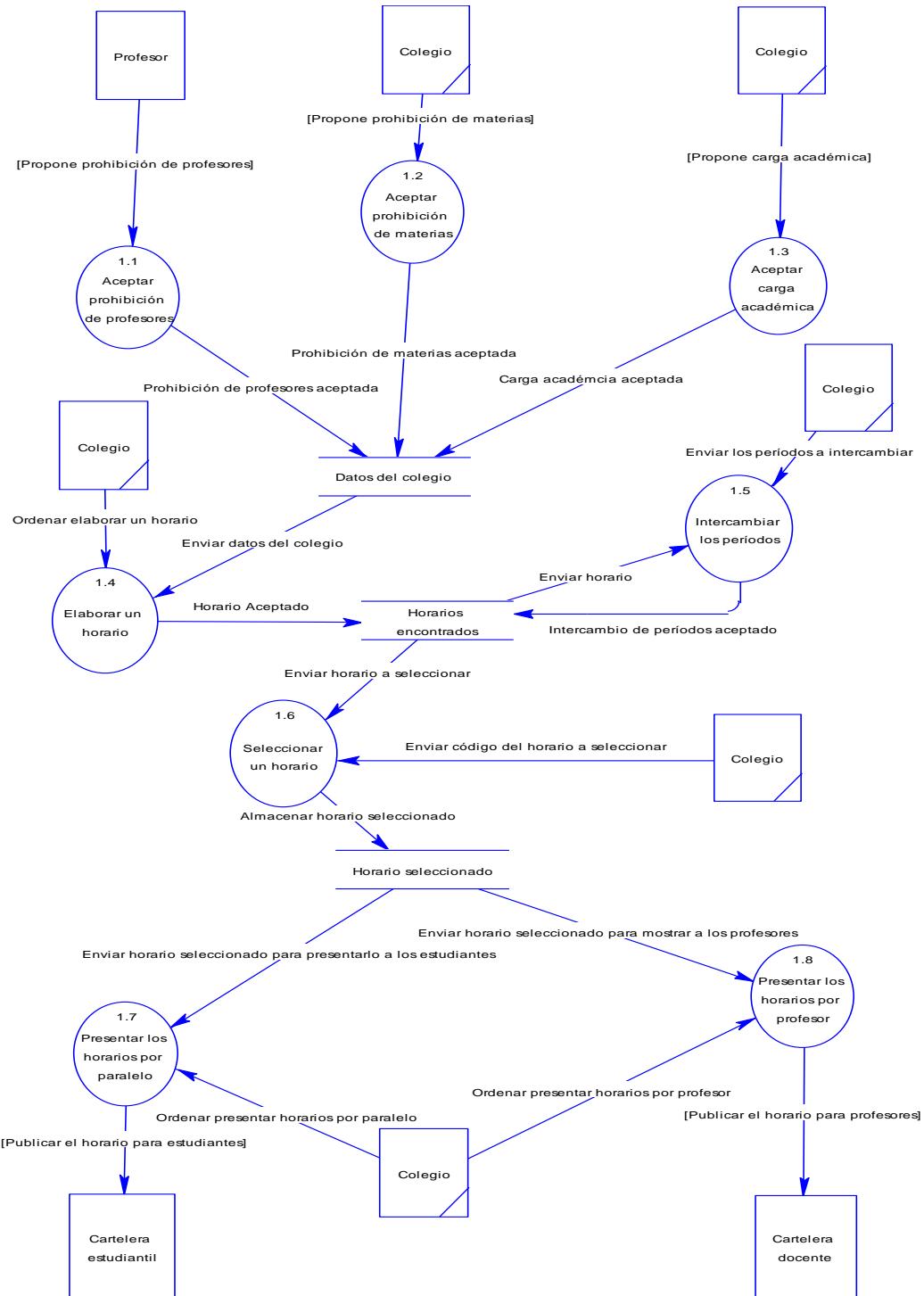
Luego de elaborar el horario, hacer:

- 1.6 Seleccionar un horario

Con el horario seleccionado, hacer:

- 1.7 Presentar los horarios por paralelo
- 1.8 Presentar los horarios por profesor

Gráfico:



Proceso 1.1: Aceptar prohibición de profesores

Nombre:

AceptarProfesorProhibicion

Descripción:

La información en el documento con el horario de prohibiciones de un profesor se almacena en la tabla **ProfesorProhibicion**.

Argumentos de entrada:

El documento con el horario de prohibiciones presentado por un profesor.

Salida del proceso:

Un conjunto de registros que se almacenan en la tabla **ProfesorProhibicion**.

Proceso 1.2: Aceptar prohibición de materias

Nombre:

AceptarMateriaProhibicion

Descripción:

La información en el documento con el horario de prohibiciones de cada materia se almacena en la tabla **MateriaProhibicion**.

Argumentos de entrada:

El documento con la prohibición de materias propuesta por el colegio.

Salida del proceso:

Un conjunto de registros que se almacenan en la tabla **MateriaProhibicion**.

Proceso 1.3: Aceptar carga académica

Nombre:

AceptarCargaAcademica

Descripción:

La información en el documento con la carga académica se almacena en la tabla **CargaAcademica**.

Argumentos de entrada:

Documento con la carga académica propuesta por el colegio.

Salida del proceso:

Un conjunto de registros que se almacenan en la tabla **CargaAcademica**.

Proceso 1.4: Elaborar un horario

Nombre:

ElaborarHorario

Descripción:

Este proceso permite generar un horario del colegio, el mismo que se guarda en *Horarios encontrados* si el usuario lo considera conveniente. El proceso fue identificado en el análisis del sistema bajo el nombre de **ElaborarHorario**.

Argumentos de entrada:

Código del horario a generar.

Salida del proceso:

Un conjunto de registros que se almacenan en las tablas **Horario** y **HorarioDetalle**.

Algoritmo:

```
Ingresar código del horario a generar
Ejecutar algoritmo evolutivo
Mientras se ejecuta el algoritmo evolutivo, permitir Cancelar o Finalizar
  Si la elaboración no fue cancelada Entonces
    Almacenar el horario encontrado al momento
```

Proceso 1.5: Intercambiar períodos

Nombre:

IntercambiarPeriodos

Descripción:

Este proceso permite modificar un horario del colegio ya elaborado intercambiando dos períodos. El proceso está identificado en la parte del análisis del sistema bajo el nombre **IntercambiarPeriodos**.

Argumentos de entrada:

El código del horario y los dos períodos del paralelo a intercambiar.

Salida del proceso:

Se modifican los registros asociados al horario especificado, en la tabla **HorarioDetalle**.

Algoritmo:

```
Ingresar Código del horario del Colegio, paralelo y los dos períodos a intercambiar
Leer de HorarioDetalle los registros del horario correspondientes al paralelo en los
dos períodos dados
Intercambiar las materias contenidas en estos dos períodos
Actualizar los registros correspondientes en HorarioDetalle
```

Proceso 1.6: Seleccionar un horario

Nombre:

SeleccionarHorario

Descripción:

Este proceso permite seleccionar un horario. El proceso recibe del colegio el código del horario a seleccionar y se verifica que exista un horario para ese código en la tabla **Horario**. De existir, se almacena el código en *HorarioSeleccionado*, caso contrario se envía un mensaje informando que no hay un horario para ese código.

Argumentos de entrada:

Código del horario a ser seleccionado.

Salida del proceso:

Registra el código del horario en *Horario seleccionado*.

Algoritmo:

```
Ingresar CodHorario
Si CodHorario existe en Horario Entonces
    Almacenar CodHorario en HorarioSeleccionado
Caso contrario
    Mostrar mensaje "No existe un horario con ese código"
```

Proceso 1.7: Presentar los horarios por paralelo

Nombre:

PresentarHorarioParalelo

Descripción:

Presenta un reporte con los horarios de cada uno de los paralelos. El proceso está identificado en la parte del análisis del sistema bajo el nombre **PresentarHorarioParalelo**.

Algoritmo:

Extrae de la tabla **HorarioDetalle** los registros relacionados con el código del horario previamente seleccionado. Para la presentación del reporte se debe cambiar la información de los campos foráneos presentes en los registros por sus nombres descriptivos ubicados en las tablas de las que provienen estos campos.

Proceso 1.7: Presentar los horarios por profesor

Nombre:

PresentarHorarioProfesor

Descripción:

Presenta para cada profesor, un reporte con los paralelos, las materias a él asignadas y los períodos en que dicta clases. El proceso está identificado en la parte del análisis del sistema bajo el nombre **PresentarHorarioProfesor**.

Algoritmo:

Extrae de la tabla **HorarioDetalle** y **CargaAcademica** los registros relacionados con el código del horario previamente seleccionado, utilizando para ello la consulta descrita a continuación. Para la presentación del reporte se debe cambiar la información de los campos foráneos presentes en los registros por sus nombres descriptivos ubicados en las tablas de las que provienen estos campos:

```
SELECT
    CargaAcademica.CodProfesor,
    HorarioDetalle.CodHora,
    HorarioDetalle.CodDia,
    Nivel.AbrNivel + " " + Especializacion.AbrEspecializacion + " " +
    ParaleloId.NomParaleloId AS NomParalelo,
    Materia.NomMateria
FROM
    (((HorarioDetalle
    INNER JOIN CargaAcademica ON (HorarioDetalle.CodMateria=CargaAcademica.CodMateria)
    AND (HorarioDetalle.CodNivel=CargaAcademica.CodNivel)
    AND (HorarioDetalle.CodEspecializacion=CargaAcademica.CodEspecializacion)
    AND (HorarioDetalle.CodParaleloId=CargaAcademica.CodParaleloId))
    INNER JOIN Nivel ON HorarioDetalle.CodNivel=Nivel.CodNivel)
    INNER JOIN Especializacion ON
    HorarioDetalle.CodEspecializacion=Especializacion.CodEspecializacion)
    INNER JOIN ParaleloId ON CargaAcademica.CodParaleloId=ParaleloId.CodParaleloId)
    INNER JOIN Materia ON CargaAcademica.CodMateria=Materia.CodMateria
WHERE
    HorarioDetalle.CodHorario=:CodHorario
ORDER BY CargaAcademica.CodProfesor, HorarioDetalle.CodHora, HorarioDetalle.CodDia
```

Procesos internos del sistema

Buscar

Nombre:

Find

Descripción:

Posiciona el cursor de la tabla en el primer registro que coincide con un criterio de búsqueda.

Argumentos de entrada:

Tabla, criterio de búsqueda.

Salida del proceso:

Tabla con el cursor posicionado en el registro que cumple con el criterio.

Presentar

Nombre:

Show

Descripción:

Presenta uno o varios registros de un conjunto de datos en un reporte.

Argumentos de entrada:

Conjunto de datos.

Salida del proceso:

Reporte.

Inicializar base de datos

Nombre:

New

Descripción:

Borra la información de la base de datos y la prepara para el ingreso de nueva información.

Algoritmo:

Vaciar base de datos

Llenar la base con datos predeterminados:

Días de la semana: lunes, martes, miércoles, jueves, viernes, sábado, domingo

Horas: primera, segunda, tercera, cuarta, recreo, quinta, sexta, séptima, octava

Tipos de prohibición de materia: Inadecuado con un peso de 50, Imposible con un peso de 100.

Tipos de prohibición de profesor: No gusta con un peso de 50, No puede con un peso de 100.

Guardar

Nombre:

Save

Descripción:

Guarda la información de la base de datos en un archivo binario.

Argumentos de entrada:

Nombre del archivo.

Salida del proceso:

Archivo binario.

Abrir

Nombre:

Open

Descripción:

Abre un archivo binario, que contiene la información de una base de datos guardada previamente.

Argumentos de entrada:

Nombre del archivo binario.

Salida del proceso:

Base de datos con la información recuperada.

Ingresar clave de acceso a la base de datos

Nombre:

Passwd

Descripción:

Permite ingresar la clave de acceso a la base de datos.

Argumentos de entrada:

Clave

Cambiar clave

Nombre:

ChangePasswd

Descripción:

Permite modificar la clave de acceso a la base de datos para garantizar que personas no autorizadas alteren la información.

Argumentos de entrada:

Clave anterior y clave nueva.

Salida del proceso:

Base de datos con la clave de acceso cambiada.

Presentar prohibición de profesores

Nombre:

PresentarProfesorProhibicion

Descripción:

Muestra un reporte con el horario de prohibiciones de cada profesor.

Seleccionar tipo de prohibición de profesor

Nombre:

ProfesorProhibicionTipo

Descripción:

Permite escoger un tipo de prohibición para el horario de prohibiciones del profesor. Los dos tipos a escoger son: *No Gusta* y *No Puede*.

Argumentos de entrada:

Profesor, tipo de prohibición, día y hora.

Salida del proceso:

Registro que se almacena en la tabla **ProfesorProhibicion**.

Seleccionar asignación de período

Nombre:

SeleccionarPeriodo

Descripción:

Permite marcar como laborable una hora de un día dado.

Argumentos de entrada:

Día y hora.

Salida del proceso:

Registro que se almacena en la tabla **HorarioLaborable**.

Asignar Paralelos

Nombre

AsignarParalelo

Descripción:

Asigna al curso un paralelo que lo conforma.

Argumentos de entrada:

Identificador del paralelo y Curso.

Salida del proceso:

Registro que se almacena en la tabla **Paralelo**.

Presentar la carga académica por materias

Nombre

PresentarCargaAcademicaMateria

Descripción:

Muestra un reporte con la carga académica en función de las materias.

Presentar prohibición de materias

Nombre:

PresentarMateriaProhibicion

Descripción:

Muestra un reporte con el horario de prohibiciones de cada materia.

Seleccionar tipo de prohibición de materia

Nombre:

MateriaProhibicion

Descripción:

Permite escoger un tipo de prohibición en el horario de prohibiciones de la materia. Los dos tipos a escoger son: *Inadecuado* e *Imposible*.

Argumentos de entrada:

Materia, tipo de prohibición, día y hora.

Salida del proceso:

Registro que se almacena en la tabla **MateriaProhibicion**.

Chequear factibilidad de los datos para generar el horario.

Nombre:

ChequearFactibilidad

Descripción:

Verifica la integridad de los datos en la base. Si bien la base de datos controla que no se violen una serie de restricciones (restricciones de integridad referencial, unicidad de valores claves, formatos de los campos, etc.), existen otras restricciones que podrían haberse pasado por alto. En general, son todas aquellas reglas que por limitaciones de la base de datos, debieron implementarse en la aplicación. Este servicio está subdividido en los siguientes subprocesos:

- **CheckProfesorCarga:** Chequea que el total de horas de clase asignadas a un profesor no exceda el límite de horas que puede trabajar semanalmente.
- **CheckCursoCarga:** El total de horas de las asignaturas a dictarse en un curso no debe sobrepasar el total de horas laborables semanales.
- **CheckCargaAcademica:** Las materias que se dictan en un paralelo deben tener un profesor.
- **CheckCargaAcademicaParalelo:** No puede haberse asignado a un paralelo una asignatura de un curso que no le corresponde.

Salida del proceso:

Reporte con los problemas que se estén dando en la base de datos, o un mensaje indicando que se está en condiciones de generar el horario.

Algoritmo:

```
CheckProfesorCarga;
CheckCursoCarga;
CheckCargaAcademica;
CheckCargaAcademicaParalelo;
Si todos los chequeos se pasan Entonces
    Mostrar Mensaje "Está en condiciones de generar el horario"
Caso contrario
    Mostrar reporte de problemas encontrados
```

Configurar

Nombre:

Configurar

Descripción:

Permite establecer los diversos parámetros de configuración, tanto del colegio como del método de búsqueda del horario. Está subdividido en cuatro categorías de parámetros que se pueden cambiar: Datos del colegio, opciones del sistema, los pesos de la función objetivo y los parámetros del algoritmo evolutivo, como se describe a continuación:

Datos del colegio: Nombre del colegio, nombre y cargo de la autoridad, nombre y cargo del responsable, carga máxima por profesor, año lectivo y comentarios.

Opciones del sistema: Generador de números aleatorios, inicializar con la hora o ingreso de las cuatro semillas del generador de números aleatorios y cada cuantas iteraciones se debe actualizar la pantalla.

Pesos de la función objetivo: Cruce de profesores, cruce de aulas, horas huecas mal ubicadas, materias cortadas, materias no dispersas, ingreso de los valores de prohibición de materias y profesores.

Parámetros del algoritmo evolutivo: Tamaño de la población, número máximo de generaciones, probabilidad de cruceamiento, probabilidad de mutación 1, orden de la mutación 1, probabilidad de mutación 2, probabilidad de reparación.

Argumentos de entrada:

Parámetros de configuración.

Salida del proceso:

Archivo de configuraciones modificado.

Horario de profesores

Nombre:

PresentarProfesorHorario

Descripción:

Muestra el horario del profesor, de uno de los horarios que están almacenados en **HorarioDetalle** con la finalidad de ver qué calidad tiene.

Argumentos de entrada:

Profesor, horario del colegio a mostrar.

Horario de paralelos

Nombre:

PresentarParaleloHorario

Descripción:

Muestra el horario del paralelo, de uno de los horarios que están almacenados en **HorarioDetalle** con la finalidad de ver qué calidad tiene.

Argumentos de entrada:

Paralelo, horario del colegio a mostrar.

Cruce de profesores

Nombre:

CruceProfesor

Descripción:

Muestra un reporte con los profesores que tienen cruce de períodos en el horario.

Argumentos de entrada:

Código del horario del colegio.

Cruce de materias

Nombre:

CruceMateria

Descripción:

Para un horario dado, muestra un reporte de las materias que, habiendo sido planificadas para que se dicten en días diferentes, constan en el mismo día.

Argumentos de entrada:

Código del horario del colegio.

Cruce de Aulas

Nombre:

CruceAula

Descripción:

Para un horario dado, muestra un reporte de los cruces de aula del mismo tipo. Un cruce de aula se da cuando dos o más paralelos requieren la misma aula a cierta hora, por ejemplo, para un aula de laboratorio.

Argumentos de entrada:

Código del horario del colegio.

Prohibición de materias no respetadas

Nombre:

ProfesorProhibicionNoRespetada

Descripción:

Muestra un reporte con las prohibiciones de las materias que no se cumplen en un horario dado.

Argumentos de entrada:

Código del horario del colegio.

Prohibición de profesores no respetadas

Nombre:

MateriaProhibicionNoRespetada

Descripción:

Muestra un reporte con las prohibiciones de los profesores que no se cumplen en un horario dado.

Argumentos de entrada:

Código del horario del colegio.

Guardar información

Nombre:

Save

Descripción:

Guarda los cambios efectuados en uno o varios registros de un conjunto de datos

Argumentos de entrada:

Conjunto de datos.

4.3.4 DICCIONARIO DE DATOS

Lista de atributos:

Nombre	Descripción	Tabla en que aparece
CodNivel	<i>Código</i> del nivel	Nivel
NomNivel	<i>Nombre</i> del nivel	Nivel
AbrNivel	<i>Abreviatura</i> del nombre del nivel	Nivel
CodEspecializacion	<i>Código</i> de la especialización	Especializacion
NomEspecializacion	Nombre de la especialización	Especializacion
AbrEspecializacion	<i>Abreviatura</i> del nombre de la especialización	Especializacion
CodParaleloId	<i>Código</i> del identificador de paralelo	ParaleloId
NomParaleloId	<i>Nombre</i> del identificador de paralelo	ParaleloId
CodDia	<i>Código</i> del día	Dia
NomDia	<i>Nombre</i> del día	Dia
CodHora	<i>Código</i> de la hora	Hora
NomHora	<i>Nombre</i> de la hora	Hora
Intervalo	<i>Intervalo</i> de la hora	Hora
CodMateria	<i>Código</i> de la materia	Materia
NomMateria	<i>Nombre</i> de la materia	Materia
CodMateProhibicionTipo	<i>Código</i> del tipo de prohibición de materia	MateriaProhibicionTipo
NomMateProhibicionTipo	<i>Nombre</i> del tipo de prohibición de materia	MateriaProhibicionTipo
ColMateProhibicionTipo	<i>Color</i> que ayuda a identificar el tipo de prohibición de materia	MateriaProhibicionTipo
ValMateProhibicionTipo	<i>Valor</i> (ponderación) que se le da al tipo de prohibición de materia	MateriaProhibicionTipo
CodAulaTipo	<i>Código</i> del tipo de aula	AulaTipo
NomAulaTipo	<i>Nombre</i> del tipo de aula	AulaTipo
AbrAulaTipo	<i>Abreviatura</i> del tipo de aula	AulaTipo
Cantidad	<i>Cantidad</i> de aulas del tipo	AulaTipo
Composicion	<i>Composición</i> de la asignatura	Asignatura
CodProfesor	<i>Código</i> del profesor	Profesor
CedProfesor	<i>Cédula</i> del profesor	Profesor
ApeProfesor	<i>Apellido</i> del profesor	Profesor
NomProfesor	<i>Nombre</i> del profesor	Profesor
CodProfProhibicionTipo	<i>Código</i> del tipo de prohibición de	ProfesorProhibicionTipo

	profesor	
NomProfProhibicionTipo	<i>Nombre</i> del tipo de prohibición de profesor	ProfesorProhibicionTipo
ColProfProhibicionTipo	<i>Color</i> que ayuda a identificar el tipo de prohibición de profesor	ProfesorProhibicionTipo
ValProfProhibicionTipo	<i>Valor</i> (ponderación) que se asigna al tipo de prohibición de profesor	ProfesorProhibicionTipo
CodHorario	<i>Código</i> del horario	Horario
MomentoInicial	<i>Momento Inicial</i> en que se elaboró el horario	Horario
MomentoFinal	<i>Momento Final</i> en que se elaboró el horario	Horario
Informe	<i>Informe</i> de las características del horario	Horario
Sesion	<i>Sesión</i> del detalle del horario	HorarioDetalle

Lista de procesos:

Nombre	Descripción	Es subproceso de
GenerarHorarioColegio	Generar el horario del colegio	Proceso principal
AceptarProfesorProhibicion	Aceptar prohibición del profesor	GenerarHorarioColegio
AceptarMateriaProhibicion	Aceptar prohibición de materias	GenerarHorarioColegio
AceptarCargaAcademica	Aceptar carga académica	GenerarHorarioColegio
ElaborarHorario	Elaborar un horario	GenerarHorarioColegio
IntercambiarPeriodos	Intercambiar los períodos	GenerarHorarioColegio
SeleccionarHorario	Seleccionar un horario	GenerarHorarioColegio
PresentarHorarioParalelo	Presentar los horarios por paralelo	GenerarHorarioColegio
PresentarHorarioProfesor	Presentar los horarios por profesor	GenerarHorarioColegio

4.3.5 SERVICIOS QUE OFRECE EL SISTEMA

Los servicios que ofrece el sistema son los que solicita el usuario al sistema por medio de una interfaz del usuario. El pedido de un servicio se lo hace en general por medio de botones, al pulsar un botón se ejecuta un proceso que al finalizar su ejecución retorna el servicio solicitado, el mismo que se visualiza en objetos de la interfaz del usuario, como son: mensajes, reportes, cuadrículas, etc.

Estos servicios pueden requerir el ingreso de información al sistema por parte del usuario, los cuales se ingresan a través de cajas de edición.

Otro servicio del sistema es el permitir el ingreso de información, lo cual se lo hace principalmente por medio de cajas de edición y cuadrículas, organizadas en formularios según la información a manipular.

Servicios de base de datos:

Los servicios que permiten manipular los registros de una base de datos, se encuentran presentes en el objeto denominado navegador de base de datos, el cual tiene el siguiente aspecto:



◀ Inicio(First):

Posiciona el cursor de la tabla en el primer registro.

◀ Anterior(Last):

Mueve el cursor de la tabla al registro que le antecede.

▶ Siguiente(Next):

Mueve el cursor de la tabla al registro que sigue.

▶ Final>Last):

Posiciona el cursor de la tabla en el último registro.

✚ Agregar(Append):

Agrega un registro a una tabla dada.

■ **Borrar(Delete):**

Borra o destruye un registro.

▲ **Modificar(Edit):**

Modifica los valores que tienen los atributos de un registro.

✓ **Guardar(Post):**

Guarda los cambios efectuados a un registro.

✗ **Cancelar(Cancel):**

Cancela los cambios efectuados a un registro que ha sido modificado, volviendo al estado inicial.

⟳ **Actualizar(Refresh):**

Actualiza los registros del conjunto de datos a los valores más recientes.

Orden natural



Ordenar(Ordenar):

Ordena los registros de acuerdo a un criterio de orden.

🔍 **Buscar:**

Este servicio utiliza el proceso Buscar, cuyo nombre es btn97FindClick

Servicios de la aplicación:

Para estudiar adecuadamente los servicios, se han agrupado como se muestra a continuación:

Archivo:

- Servicio solicitado: Inicializar la base de datos
 - Aplicado a: Botón.
 - Servicio en: Diálogo confirmando el borrado de la información existente.
 - Proceso llamado: Inicializar base de datos.
- Servicio solicitado: Abrir archivo binario.
 - Aplicado a: Botón.
 - Servicio en: Diálogo para abrir un archivo.
 - Proceso llamado: Abrir.
- Servicio solicitado: Guardar la base de datos en un archivo binario.
 - Aplicado a: Botón.
 - Servicio en: Diálogo para guardar un archivo.
 - Proceso llamado: Guardar.
- Servicio solicitado: Ingresar clave de acceso a la base de datos.
 - Aplicado a: Botón.
 - Servicio en: Diálogo solicitando clave.
 - Proceso llamado: Ingresar clave.
- Servicio solicitado: Cambiar la clave de acceso a la base de datos.
 - Aplicado a: Botón.
 - Servicio en: Diálogo solicitando clave anterior y nueva clave.
 - Proceso llamado: Cambiar clave.
- Servicio solicitado: Salir de la aplicación.
 - Aplicado a: Botón.
 - Servicio en: Diálogo confirmando el salir.

Datos:

Horario laborable

- Servicio solicitado: ingresar los días laborables.
 - Aplicado a: Botón.
 - Servicio en: Formulario para ingresar los días laborables.

- Proceso: “Mostrar formulario”
- Servicio solicitado: ingresar las horas académicas.
 - Aplicado a: Botón.
 - Servicio en: Formulario para ingresar las horas académicas.
 - Proceso: “Mostrar formulario”
- Servicio solicitado: ingresar los períodos laborables.
 - Aplicado a: Botón.
 - Servicio en: Formulario para ingresar los períodos laborables.
 - Proceso: “Mostrar formulario”
- Servicio solicitado: Formulario para ingresar los días laborables
 - Aplicado a: Formulario.
 - Servicio interno: Ingresar información
 - Aplicado a: Cuadrícula.
 - Servicio interno: Manejo de la información en la cuadrícula.
 - Aplicado a: Navegador de base de datos.
 - Servicio interno: Presentar un reporte.
 - Aplicado a: Botón.
 - Servicio en: Reporte.
 - Proceso llamado: Presentar.
- Servicio solicitado: Formulario para ingresar las horas académicas.
 - Aplicado a: Formulario.
 - Servicio interno: Ingresar información
 - Aplicado a: Cuadrícula.
 - Servicio interno: Manejo de la información en la cuadrícula.
 - Aplicado a: Navegador de base de datos.
 - Servicio interno: Presentar un reporte.
 - Aplicado a: Botón.
 - Servicio en: Reporte.
 - Proceso llamado: Presentar.
- Servicio solicitado: Formulario para ingresar los períodos laborables
 - Aplicado a: Formulario.
 - Servicio interno: Ingresar información.

- Aplicado a: Cuadrícula.
- Servicio interno: Seleccionar período.
 - Aplicado a: Lista.
 - Servicio en: Celda de cuadrícula (período).
 - Proceso llamado: Seleccionar período.
- Servicio interno: Aceptar.
 - Aplicado a: Botón.
 - Proceso llamado: Guardar información.
- Servicio interno: Cancelar.
 - Aplicado a: Botón.
 - Servicio en: Cuadrícula.
 - Proceso “cancelar información ingresada”.

Paralelos

- Servicio solicitado: Ingresar los niveles.
 - Aplicado a: Botón.
 - Servicio en: Formulario para ingresar los niveles.
 - Proceso: “Mostrar formulario”
- Servicio solicitado: Ingresar las especializaciones.
 - Aplicado a: Botón.
 - Servicio en: Formulario para ingresar las especializaciones.
 - Proceso: “Mostrar formulario”
- Servicio solicitado: Ingresar los identificadores de paralelo.
 - Aplicado a: Botón.
 - Servicio en: Formulario para ingresar los identificadores de paralelo.
 - Proceso: “Mostrar formulario”
- Servicio solicitado: Ingresar los paralelos.
 - Aplicado a: Botón.
 - Servicio en: Formulario para ingresar los paralelos.
 - Proceso: “Mostrar formulario”
- Servicio solicitado: Formulario para ingresar los niveles
 - Aplicado a: Formulario.
 - Servicio interno: Ingresar información
 - Aplicado a: Cuadrícula.

- Servicio interno: Manejo de la información en la cuadrícula.
 - Aplicado a: Navegador de base de datos.
- Servicio interno: Presentar un reporte.
 - Aplicado a: Botón.
 - Servicio en: Reporte.
 - Proceso llamado: Presentar.
- Servicio solicitado: Formulario para ingresar las especializaciones
 - Aplicado a: Formulario.
 - Servicio interno: Ingresar información
 - Aplicado a: Cuadrícula.
 - Servicio interno: Manejo de la información en la cuadrícula.
 - Aplicado a: Navegador de base de datos.
 - Servicio interno: Presentar un reporte.
 - Aplicado a: Botón.
 - Servicio en: Reporte.
 - Proceso llamado: Presentar.
- Servicio solicitado: Formulario para ingresar los identificadores de paralelo
 - Aplicado a: Formulario.
 - Servicio interno: Ingresar información
 - Aplicado a: Cuadrícula.
 - Servicio interno: Manejo de la información en la cuadrícula.
 - Aplicado a: Navegador de base de datos.
 - Servicio interno: Presentar un reporte.
 - Aplicado a: Botón.
 - Servicio en: Reporte.
 - Proceso llamado: Presentar.
- Servicio solicitado: Formulario para ingresar los paralelos
 - Aplicado a: Formulario.
 - Servicio interno: Ingresar información
 - Aplicado a: Cuadrícula.
 - Servicio interno: Manejo de la información en la cuadrícula.
 - Aplicado a: Navegador de base de datos.

- Servicio interno: Presentar un reporte.
 - Aplicado a: Botón.
 - Servicio en: Reporte.
 - Proceso llamado: Presentar.
- Servicio interno: Seleccionar paralelos.
 - Aplicado a: lista de chequeadores.
 - Servicio en: Fila de la cuadrícula (curso).
 - Proceso llamado: Seleccionar paralelos.

Asignaturas

- Servicio solicitado: Ingresar las materias.
 - Aplicado a: Botón.
 - Servicio en: Formulario para ingresar las materias.
 - Proceso: “Mostrar formulario”
- Servicio solicitado: ingresar los tipos de aula.
 - Aplicado a: Botón.
 - Servicio en: Formulario para ingresar los tipos de aula.
 - Proceso: “Mostrar formulario”
- Servicio solicitado: ingresar las asignaturas.
 - Aplicado a: Botón.
 - Servicio en: Formulario para ingresar las asignaturas.
 - Proceso: “Mostrar formulario”
- Servicio solicitado: Formulario para ingresar las materias.
 - Aplicado a: Formulario.
 - Servicio interno: Ingresar información
 - Aplicado a: Cuadrícula.
 - Servicio interno: Manejo de la información en la cuadrícula.
 - Aplicado a: Navegador de base de datos.
 - Servicio interno: Presentar un reporte.
 - Aplicado a: Botón.
 - Servicio en: Reporte.
 - Proceso llamado: Presentar.
 - Servicio interno: Ingresar el horario de prohibiciones de la materia.
 - Aplicado a: Botón.

- Servicio en: Formulario para ingresar el horario de prohibiciones de la materia.
- Proceso: “Mostrar formulario”
- Servicio solicitado: Formulario para ingresar el horario de prohibiciones de la materia.
 - Aplicado a: Formulario.
 - Servicio interno: Ingresar información.
 - Aplicado a: Cuadrícula.
 - Servicio interno: Seleccionar tipo de prohibición.
 - Aplicado a: Lista.
 - Servicio en: celda de la cuadrícula (período).
 - Proceso llamado: Seleccionar tipo de prohibición de materia.
 - Servicio interno: Aceptar.
 - Aplicado a: Botón.
 - Proceso llamado: Guardar información.
 - Servicio interno: Cancelar.
 - Aplicado a: Botón.
 - Servicio en: Cuadrícula.
 - Proceso “Cancelar la información ingresada”.
- Servicio solicitado: Formulario para ingresar los tipos de aula.
 - Aplicado a: Formulario.
 - Servicio interno: Ingresar información
 - Aplicado a: Cuadrícula.
 - Servicio interno: Manejo de la información en la cuadrícula.
 - Aplicado a: Navegador de base de datos.
 - Servicio interno: Presentar un reporte.
 - Aplicado a: Botón.
 - Servicio en: Reporte.
 - Proceso llamado: Presentar.
- Servicio solicitado: Formulario para ingresar las asignaturas.
 - Aplicado a: Formulario.
 - Servicio interno: Ingresar información

- Aplicado a: Cuadrícula.
- Servicio interno: Manejo de la información en la cuadrícula.
 - Aplicado a: Navegador de base de datos.
- Servicio interno: Presentar un reporte.
 - Aplicado a: Botón.
 - Servicio en: Reporte.
 - Proceso llamado: Presentar.

Profesores

- Servicio solicitado: ingresar los profesores.
 - Aplicado a: Botón.
 - Servicio en: Formulario para ingresar los profesores.
 - Proceso: “Mostrar formulario”
- Servicio solicitado: Formulario para ingresar los profesores.
 - Aplicado a: Formulario.
 - Servicio interno: Ingresar información
 - Aplicado a: Cuadrícula.
 - Servicio interno: Manejo de la información en la cuadrícula.
 - Aplicado a: Navegador de base de datos.
 - Servicio interno: Presentar un reporte.
 - Aplicado a: Botón.
 - Servicio en: Reporte.
 - Proceso llamado: Presentar.
- Servicio interno: Ingresar el horario de prohibiciones del profesor
 - Aplicado a: Botón.
 - Servicio en: Formulario para ingresar el horario de prohibiciones del profesor.
 - Proceso: “Mostrar formulario”.
- Servicio interno: ingresar la carga académica del profesor.
 - Aplicado a: Botón.
 - Servicio en: Formulario para ingresar la carga académica del profesor.
 - Proceso: “Mostrar formulario”.
- Servicio solicitado: Formulario para ingresar el horario de prohibiciones del profesor.
 - Aplicado a: Formulario.

- Servicio interno: Ingresar información.
 - Aplicado a: Cuadrícula.
- Servicio interno: Seleccionar tipo de prohibición.
 - Aplicado a: Lista.
- Servicio interno: Aceptar.
 - Aplicado a: Botón.
 - Proceso llamado: Aceptar prohibición de profesores.
- Servicio interno: Cancelar.
 - Aplicado a: Botón.
 - Proceso: “Cancelar información ingresada”.
- Servicio solicitado: Formulario para ingresar la carga académica del profesor.
 - Aplicado a: Formulario.
 - Servicio interno: Ingresar información.
 - Aplicado a: Cuadrícula.
 - Servicio interno: Manejo de la información en la cuadrícula.
 - Aplicado a: Navegador de base de datos.
 - Proceso llamado: Aceptar carga académica.
 - Servicio interno: Presentar un reporte.
 - Aplicado a: Botón.
 - Servicio en: Reporte.
 - Proceso llamado: Presentar.

Herramientas

- Servicio solicitado: Establecer los parámetros de configuración.
 - Aplicado a: Botón.
 - Servicio en: Formulario para establecer los parámetros de configuración.
 - Proceso: “Mostrar formulario”
- Servicio solicitado: Chequear la factibilidad de los datos para generar el horario.
 - Aplicado a: Botón.
 - Servicio en dos casos excluyentes:
 1. Diálogo indicando: “no se encontraron errores, está listo para generar horario”.

2. Formulario mostrando los problemas encontrados en la base de datos.
- Proceso llamado: Chequear factibilidad de los datos para generar el horario.
 - Servicio solicitado: Elaborar un horario.
 - Aplicado a: Botón.
 - Servicio en: Diálogo pidiendo el código del horario a elaborar.
 - Proceso llamado al aceptar el diálogo: Formulario mostrando el progreso de la elaboración del horario.
 - Proceso llamado: Elaborar un horario.
 - Servicio solicitado: Formulario para establecer los parámetros de configuración.
 - Aplicado a: Formulario.
 - Servicio en: cajas de edición que permiten establecer los diferentes parámetros de configuración.
 - Servicio interno: Aceptar
 - Aplicado a: Botón.
 - Proceso llamado: Configuración.
 - Servicio Solicitado: Formulario de problemas encontrados en la base de datos.
 - Aplicado a: Formulario.
 - Servicio en: Memos que muestran los problemas encontrados en la base de datos.
 - Servicio interno: Cerrar.
 - Aplicado a: Botón.
 - Proceso “Cerrar formulario”.
 - Servicio solicitado: Formulario mostrando el progreso de la elaboración del horario.
 - Aplicado a: Formulario.
 - Servicio interno: Mostrar información
 - Aplicado a: Paneles y rótulos.
 - Servicio interno: Cancelar.
 - Aplicado a: Botón.
 - Proceso “Cancelar Elaboración del horario”.
 - Servicio interno: Finalizar
 - Aplicado a: Botón.
 - Proceso “Almacenar el horario encontrado al momento”.

Ver

- Servicio solicitado: Ver los horarios elaborados.
 - Aplicado a: Botón.
 - Servicio en: Formulario para ver los horarios elaborados.
 - Proceso: “Mostrar formulario”.
- Servicio solicitado: Mostrar un reporte con los horarios por profesor.
 - Aplicado a: Botón.
 - Servicio en: Reporte.
 - Proceso llamado: Presentar los horarios por profesor.
- Servicio solicitado: Mostrar un reporte con los horarios por paralelo.
 - Aplicado a: Botón.
 - Servicio en: Reporte.
 - Proceso llamado: Presentar los horarios por paralelo.
- Servicio solicitado: Mostrar un reporte con las prohibiciones de profesores.
 - Aplicado a: Botón.
 - Servicio en: Reporte.
 - Proceso llamado: Presentar prohibición de profesores.
- Servicio solicitado: Mostrar un reporte con las prohibiciones de materias.
 - Aplicado a: Botón.
 - Servicio en: Reporte.
 - Proceso llamado: Presentar prohibición de materias.
- Servicio solicitado: Mostrar un reporte con la carga académica por materias
 - Aplicado a: Botón.
 - Servicio en: Reporte.
 - Proceso llamado: Presentar la carga académica por materias.
- Servicio solicitado: Mostrar un reporte con la carga académica por profesores
 - Aplicado a: Botón.
 - Servicio en: Reporte.
- Servicio solicitado: Formulario para ver los horarios elaborados.
 - Aplicado a: Formulario.
 - Servicio interno: Ver información
 - Aplicado a: Cuadrícula.

- Servicio interno: Manejo de la información en la cuadrícula.
 - Aplicado a: Navegador de base de datos.
- Servicio interno: Presentar un reporte.
 - Aplicado a: Botón.
 - Servicio en: Reporte.
 - Proceso llamado: Presentar.
- Servicio interno: Seleccionar un horario.
 - Aplicado a: Botón.
 - Servicio en: Fila de la cuadrícula (horario).
 - Proceso llamado: Seleccionar un horario.
- Servicio interno: Horario de profesores.
 - Aplicado a: Botón
 - Servicio en: Formulario que muestra el horario de profesores.
 - Proceso: “Mostrar formulario”
- Servicio interno: Horario de paralelos.
 - Aplicado a: Botón
 - Servicio en: Formulario que muestra el horario de paralelos.
 - Proceso: “Mostrar formulario”
- Servicio interno: Cruce de profesores.
 - Aplicado a: Botón.
 - Servicio en: Formulario que muestra el cruce de profesores.
 - Proceso: “Mostrar formulario”
- Servicio interno: Cruce de materias
 - Aplicado a: Botón
 - Servicio en: Formulario que muestra el cruce de materias.
 - Proceso: “Mostrar formulario”
- Servicio interno: Cruce de aulas
 - Aplicado a: Botón
 - Servicio en: Formulario que muestra el cruce de aulas.
 - Proceso: “Mostrar formulario”
- Servicio interno: Prohibición de materias no respetadas
 - Aplicado a: Botón
 - Servicio en: Formulario que muestra la prohibición de materias no respetadas.

- Proceso: “Mostrar formulario”
- Servicio interno: Prohibición de profesores no respetadas
 - Aplicado a: Botón
 - Servicio en: Formulario que muestra la prohibición de profesores no respetadas.
 - Proceso: “Mostrar formulario”
- Servicio solicitado: Formulario que muestra el horario de profesores.
 - Aplicado a: Formulario.
 - Servicio interno: Ver información.
 - Aplicado a: Cuadrícula.
 - Servicio interno: Seleccionar profesor.
 - Aplicado a: Lista desplegable que mira en otro conjunto de datos.
 - Servicio interno: Seleccionar qué mostrar en el horario, Materias, Nivel-Paralelo, Nivel-Especialización-Paralelo o Nivel-Paralelo-Especialización.
 - Aplicado a: Lista desplegable.
 - Servicio interno: Mostrar.
 - Aplicado a: Botón.
 - Servicio en: Cuadrícula.
 - Proceso llamado: Horario de profesores, cuyos argumentos son los datos marcados en las listas desplegables.
- Servicio solicitado: Formulario que muestra el horario de paralelos.
 - Aplicado a: Formulario.
 - Servicio interno: Ver información.
 - Aplicado a: Cuadrícula.
 - Servicio interno: Seleccionar nivel.
 - Aplicado a: Lista desplegable que mira a otro conjunto de datos.
 - Servicio interno: Seleccionar especialización.
 - Aplicado a: Lista desplegable que mira a otro conjunto de datos.
 - Servicio interno: Seleccionar paralelo.
 - Aplicado a: Lista desplegable que mira a otro conjunto de datos.
 - Servicio interno: Seleccionar qué mostrar en el horario: Materia o Profesor.
 - Aplicado a: Lista desplegable.
 - Servicio interno: Mostrar.

- Aplicado a: Botón.
- Servicio en: Cuadrícula.
- Proceso llamado: Horario de paralelos, cuyos argumentos son los datos marcados en las listas desplegables.
- Servicio interno: Intercambiar períodos.
 - Aplicado a: Botón
 - Servicio en: Diálogo solicitando el período que será intercambiado con el que ya está seleccionado en la cuadrícula.
 - Proceso llamado: Intercambiar períodos.
- Servicio solicitado: Formulario que muestra el cruce de profesores.
 - Aplicado a: Formulario.
 - Proceso llamado: Cruce de profesores (al abrir el formulario).
 - Servicio en: Dos cuadrículas sincronizadas, la una para ver los profesores y la otra para ver los cruces del profesor seleccionado.
 - Servicio interno: Manejo de la información en la cuadrícula del profesor.
 - Aplicado a: Navegador de base de datos.
 - Servicio interno: Presentar un reporte.
 - Aplicado a: Botón.
 - Servicio en: Reporte.
 - Proceso llamado: Presentar.
- Servicio solicitado: Formulario que muestra el cruce de materias.
 - Aplicado a: Formulario.
 - Proceso llamado: Cruce de materias (al abrir el formulario).
 - Servicio en: Dos cuadrículas sincronizadas, la una para ver las materias, y la otra para ver los cruces de la materia seleccionada.
 - Servicio interno: Manejo de la información en la cuadrícula de la materia
 - Aplicado a: Navegador de base de datos.
 - Servicio interno: Presentar un reporte.
 - Aplicado a: Botón.
 - Servicio en: Reporte.
 - Proceso llamado: Presentar.
- Servicio solicitado: Formulario que muestra el cruce de aulas.
 - Aplicado a: Formulario.
 - Proceso llamado: Cruce de aulas (al abrir el formulario).

- Servicio en: Dos cuadrículas sincronizadas, la una para ver los tipos de aula y la otra para ver los cruces del tipo de aula seleccionado.
- Servicio interno: Manejo de la información en la cuadrícula del tipo de aula.
 - Aplicado a: Navegador de base de datos.
- Servicio interno: Presentar un reporte.
 - Aplicado a: Botón.
 - Servicio en: Reporte.
 - Proceso llamado: Presentar.
- Servicio solicitado: Formulario que muestra la prohibición de materias no respetadas.
 - Aplicado a: Formulario.
 - Proceso llamado: Prohibición de materias no respetadas (al abrir formulario).
 - Servicio en: Cuadrícula.
 - Servicio interno: Manejo de la información en la cuadrícula.
 - Aplicado a: Navegador de base de datos.
 - Servicio interno: Presentar un reporte.
 - Aplicado a: Botón.
 - Servicio en: Reporte.
 - Proceso llamado: Presentar.
- Servicio solicitado: Formulario que muestra la prohibición de profesores no respetadas.
 - Aplicado a: Formulario.
 - Proceso llamado: Prohibición de profesores no respetadas (al abrir formulario).
 - Servicio en: Cuadrícula.
 - Servicio interno: Manejo de la información en la cuadrícula.
 - Aplicado a: Navegador de base de datos.
 - Servicio interno: Presentar un reporte.
 - Aplicado a: Botón.
 - Servicio en: Reporte.
 - Proceso llamado: Presentar.

Ayuda

- Servicio solicitado: Mostrar el contenido de la ayuda.
 - Aplicado a: Botón.
 - Servicio en: Diálogo que muestra el contenido de la ayuda.
- Servicio solicitado: Mostrar el índice de la ayuda.
 - Aplicado a: Botón.
 - Servicio en: Diálogo que muestra el Índice de la ayuda.
- Servicio solicitado: Mostrar el acerca de... de la aplicación.
 - Aplicado a: Botón.
 - Servicio en: Diálogo con información de la aplicación.

4.4 IMPLEMENTACIÓN

4.4.1 INTRODUCCIÓN

Para la implementación del sistema, se utiliza el Borland Delphi versión 4.03, puesto que es una de las herramientas de programación más poderosas existentes en el mercado y además la mejor conocida en la Facultad de Ciencias, lo que significa que el costo de aprendizaje es menor y que será de más fácil acceso a aquellos miembros de la facultad que tengan interés en la presente tesis. Los requisitos de software para el desarrollo del sistema son los siguientes:

Herramientas para el desarrollo del sistema:			
Nombre	Categoría	Tipo	Dónde conseguir o informarse
Windows 98	Sistema operativo	Comercial	www.microsoft.com/
Borland Delphi 4.03	Compilador	Comercial	www.borland.com/
ArCtrls	Librería	FreeWare	http://www.geocities.com/SiliconValley/Grid/3690/
NavBtn	Librería	FreeWare	http://www.geocities.com/SiliconValley/Heights/7874/delphi.htm
RxLib 2.60	Librería	FreeWare	http://rx.demo.ru/
Toolbar 97 1.70	Librería	FreeWare	http://www.jordanr.dhs.org/
Herramientas para la distribución del sistema:			
Nombre	Categoría	Tipo	Dónde conseguir o informarse
ASPack 1.08.03	Compresor EXE	ShareWare	www.entechtaian.com/aspack.htm
Inno Setup Compiler 1.11	Instalador	FreeWare	http://www.jordanr.dhs.org/
ScriptMaker 1.11.x*	Shell del ISC 1.11	FreeWare	http://tafweb.hypermart.net/
Internet Explorer 5.0	Navegador/Shell	Actualización	www.microsoft.com/ie/
WinZip 7.0 SP-1	Compresor ZIP	ShareWare	http://www.winzip.com/

El equipo mínimo indispensable para ejecutar la aplicación es un Pentium de 100MHz con 32MB de RAM y espacio libre en disco duro de 10MB, con sistema operativo Windows 95 o superior. Para el código fuente y la documentación se requieren de 30MB de espacio adicional en el disco duro.

La implementación de la base de datos se la realizó en PARADOX 7.0, debido a que el Delphi utiliza este formato de manera estándar.

4.4.2 IMPLEMENTACIÓN DEL MODELO EN COMPUTADORA

Como se vio en 3.3, el modelo matemático fue planteado en forma de un conjunto de variables de unos y ceros, sin embargo, en el computador los datos fueron almacenados en arreglos dinámicos de enteros de 16 bits, ya que resulta más natural trabajar así. A continuación se listan la interfaz de las dos unidades involucradas en la implementación del algoritmo evolutivo:

Unidad Kermodel

Unidad que contiene la implementación de la evaluación de la función objetivo en un horario, los operadores genéticos y el guardado de un horario en la base de datos.

```
unit KerModel;

interface

uses
  Classes, DB, DBTables, Dialogs, Math, SysConst;

type
  TDYNAMICWordArray = array of Word;
  TDYNAMICWordArrayArray = array of TDYNAMICWordArray;
  TDYNAMICSmallintArray = array of Smallint;
  TDYNAMICSmallintArrayArray = array of TDYNAMICSmallintArray;
  TDYNAMICSmallintArrayArrayArray = array of TDYNAMICSmallintArrayArray;
  TDYNAMICShortintArray = array of Shortint;
  TDYNAMICShortintArrayArray = array of TDYNAMICShortintArray;
  TDYNAMICIntegerArray = array of Integer;
  TDYNAMICIntegerArrayArray = array of TDYNAMICIntegerArray;
  TDYNAMICLongintArray = array of Longint;
  TDYNAMICLongintArrayArray = array of TDYNAMICLongintArray;
  TDYNAMICDoubleArray = array of Double;
  TDYNAMICDoubleArrayArray = array of TDYNAMICDoubleArray;
  TDYNAMICStringArray = array of string;
  PLongintArray = ^TLongintArray;
  TLongintArray = array[0..16383] of Longint;
  PSmallintArray = ^TSmallintArray;
  TSmallintArray = array[0..16383] of Smallint;
  PSmallintArrayArray = ^TSmallintArrayArray;
  TSmallintArrayArray = array[0..0] of PSmallintArray;
  PShortintArray = ^TShortintArray;
  TShortintArray = array[0..32767] of Shortint;
  PDoubleArray = ^TDoubleArray;
  TDoubleArray = array[0..0] of Double;
  {
  Clase TModeloHorario
  Descripción:
    Implementa la carga de la información desde la base de datos a la memoria
    RAM.
  Miembros privados:
```

Todos los arreglos dinámicos que contienen la información, los campos que contienen los pesos de la función objetivo y las funciones de uso interno.

Miembros protegidos:

Arreglo dinámico que contiene el molde de un horario, que facilita la construcción de soluciones.

Miembros públicos:

El constructor, la función que permite configurar los pesos y los pesos de cada restricción.

```

} TModeloHorario = class
private
  FCruceProfesorValor,
  FCruceAulaTipoValor,
  FHoraHuecaDesubicadaValor,
  FSesionCortadaValor,
  FMateriaNoDispersaValor: Double;
  FHorarioLaborableAdia,
  FHorarioLaborableAHora,
  FDiaAMaxHorarioLaborable,
  FSesionAAsignatura,
  FSesionAMateria,
  FAulaTipoACantidad,
  FMateriaProhibicionAMateria,
  FMateriaProhibicionAHorarioLaborable,
  FMateriaProhibicionAMateriaProhibicionTipo,
  FProfesorProhibicionAProfesor,
  FProfesorProhibicionAHorarioLaborable,
  FProfesorProhibicionAProfesorProhibicionTipo,
  FAsignaturaAAulaTipo,
  FParaleloACurso,
  FParaleloANivel,
  FParaleloAParaleloTipo,
  FParaleloAEspecializacion,
  FCursoADuracion,
  FParaleloASesionCant: TDynamicSmallintArray;
  FSesionADuracion: array[-1..16382] of Smallint;
  FDiaHoraAHorarioLaborable,
  FNivelEspecializacionACurso,
  FCursoParaleloTipoAParalelo,
  FParaleloMateriaAProfesor,
  FMateriaCursoAAsignatura,
  FMoldeHorarioDetalle,
  FAsignaturaASesiones: TDynamicSmallintArrayArray;
  FProfesorHorarioLaborableAProfesorProhibicionTipo,
  FMateriaHorarioLaborableAMateriaProhibicionTipo:
    TDynamicShortintArrayArray,
  FMateriaProhibicionTipoAValor,
  FProfesorProhibicionTipoAValor: array[-1..4094] of Double;
  FMateriaProhibicionAValor,
  FProfesorProhibicionAValor: TDynamicDoubleArray;
  FParaleloCant,
  FMateriaCant,
  FDiaCant,
  FHoraCant,
  FHorarioLaborableCant,
  FProfesorCant,
  FNivelCant,
  FEspecializacionCant,
  FCursoCant,
  FAulaTipoCant,
  FMaxProfesorProhibicionTipo,
  FMaxMateriaProhibicionTipo: Smallint;
  FParaleloTipoACodParaleloTipo,
  FMateriaACodMateria,

```

```

FDiaACodDia,
FHoraACodHora,
FNivelACodNivel,
FEspecializacionACodEspecializacion: TDynamicLongintArray;
function GetDiaAMaxHorarioLaborable(d: Smallint): Smallint;
protected
  property MoldeHorarioDetalle: TDynamicSmallintArrayArray read
    FMoldeHorarioDetalle;
public
  procedure Configurar(
    ACruceProfesorValor,
    ACruceAulaTipoValor,
    AHoraHuecaDesubicadaValor,
    ASesionCortadaValor,
    AMateriaNoDispersaValor: Double);
  constructor CrearDesdeDatabase(
    ADatabase: TDatabase;
    ACruceProfesorValor,
    ACruceAulaTipoValor,
    AHoraHuecaDesubicadaValor,
    ASesionCortadaValor,
    AMateriaNoDispersaValor: Double);
  property CruceProfesorValor: Double read FCruceProfesorValor;
  property CruceAulaTipoValor: Double read FCruceAulaTipoValor;
  property HoraHuecaDesubicadaValor: Double read FHoraHuecaDesubicadaValor;
  property SesionCortadaValor: Double read FSesionCortadaValor;
  property MateriaNoDispersaValor: Double read FMateriaNoDispersaValor;
end;
{
Clase TObjetoModeloHorario
Descripción:
  Implementa una solución del problema.
Miembros privados:
  Todos los arreglos dinámicos que contienen la información consistentes en
  el horario, las claves aleatorias y datos temporales que aceleran la
  evaluación de la función objetivo, los campos que contienen los valores
  de cada parte de la función objetivo y las funciones de uso interno.
Miembros protegidos:
  Propiedad que fuerza a que se reevalúe toda la función objetivo.
Miembros públicos:
  propiedades que devuelven el valor de cada componente de la función
  objetivo, el valor de la función objetivo, el constructor, funciones que
  permiten guardar el horario en una base de datos, métodos que consisten
  en los operadores genéticos, como son cruce, mutación, etc., ModeloHorario
  al que pertenece esta solución.
}
TObjetoModeloHorario = class
private
  FModeloHorario: TModeloHorario;
  FClaveAleatoria: TDynamicLongintArrayArray;
  FHorarioDetalle,
  FCruceMateriaHorarioLaborable,
  FCruceProfesorHorarioLaborable,
  FCruceAulaTipoHorarioLaborable,
  FAntMateriaDiaMinHora,
  FAntMateriaDiaMaxHora: TDynamicSmallintArrayArray;
  FParaleloMateriaDiaMinHora,
  FParaleloMateriaDiaMaxHora
  : TDynamicSmallintArrayArray;
  FParaleloMateriaNoDispersa: TDynamicSmallintArray;
  FCruceProfesor,
  FCruceAulaTipo,
  FHoraHuecaDesubicada,
  FSesionCortada,
  FMateriaNoDispersa,
  FMateriaProhibicion,
  FProfesorProhibicion,

```

```

    FAntMateriaNoDispersa: Integer;
    FMateriaProhibicionValor,
    FProfesorProhibicionValor,
    FValor: Double;
    FRecalcularValor: Boolean;
    function GetValor: Double;
    procedure DoGetValor;
    procedure Normalizar(AParalelo: Smallint; var APeriodo: Smallint);
    procedure SetClaveAleatoriaInterno(AParalelo, APeriodo: Smallint;
        AClaveAleatoria: Integer); overload;
    procedure SetClaveAleatoriaInterno(AParalelo, APeriodo, ADuracion: Smallint;
        AClaveAleatoria: Integer); overload;
    procedure AleatorizarClave(AParalelo: Smallint);
    function GetMateriaNoDispersa: Integer;
    procedure DoGetMateriaNoDispersa;
    procedure DoGetHoraHuecaDesubicada;
    procedure DoGetSesionCortada;
    function GetHoraHuecaDesubicada: Integer;
    function GetSesionCortada: Integer;
    procedure DoGetProfesorProhibicionValor;
    procedure DoGetMateriaProhibicionValor;
    function GetProfesorProhibicionValor: Double;
    function GetMateriaProhibicionValor: Double;
    function GetMateriaNoDispersaValor: Double;
    function GetHoraHuecaDesubicadaValor: Double;
    function GetCruceProfesorValor: Double;
    function GetSesionCortadaValor: Double;
    function GetCruceAulaTipoValor: Double;
    function GetProfesorProhibicion: Integer;
    function GetMateriaProhibicion: Integer;
    procedure MutarInterno;
    function DescensoRapidoInterno: Boolean;
    procedure IntercambiarInterno(AParalelo, APeriodo, APeriodo1: Smallint;
        FueEvaluado: Boolean = False);
    procedure Intercambiar(AParalelo, APeriodo, APeriodo1: Smallint);
    function EvaluarIntercambioInterno(AParalelo, APeriodo,
        APeriodo1: Smallint): Double;
    property CruceProfesorHorarioLaborable: TDynamicSmallintArrayArray read
        FCruceProfesorHorarioLaborable;
    property CruceMateriaHorarioLaborable: TDynamicSmallintArrayArray read
        FCruceMateriaHorarioLaborable;
    property CruceAulaTipoHorarioLaborable: TDynamicSmallintArrayArray read
        FCruceAulaTipoHorarioLaborable;
    procedure DoGetCruceProfesor;
    procedure DoGetCruceAulaTipo;
    procedure Actualizar;
    procedure ActualizarCruceAulaTipoHorarioLaborable;
    procedure ActualizarCruceProfesorHorarioLaborable;
    procedure ActualizarCruceMateriaHorarioLaborable;
    procedure ActualizarParaleloMateriaDiaMinMaxHora; overload;
    procedure ActualizarParaleloMateriaDiaMinMaxHora(AParalelo: Smallint);
        overload;
    function GetParaleloMateriaNoDispersa(AParalelo: Smallint;
        var AMateriaDiaMaxHora: TDynamicSmallintArrayArray): Smallint;
protected
    property RecalcularValor: boolean read FRecalcularValor write
        FRecalcularValor;
public
    procedure DescensoRapido;
    procedure DescensoRapidoForzado;
    procedure InvalidarValor;
    procedure SaveToFile(const AFileName: string);
    procedure SaveToDatabase(CodHorario: Integer; MomentoInicial,
        MomentoFinal: TDateTime; Informe: TStrings; ADatabase: TDatabase);

```

```

procedure SaveToStream(Stream: TStream);
procedure LoadFromStream(Stream: TStream);
constructor CrearDesdeModelo(AModeloHorario: TModeloHorario);
procedure HacerAleatorio;
procedure Mutar; overload;
procedure Mutar(Orden: Integer); overload;
procedure MutarDia;
procedure Assign(AObjetoModeloHorario: TObjetoModeloHorario);
destructor Destroy; override;
property Valor: Double read GetValor;
property CruceProfesor: Integer read FCruceProfesor;
property CruceAulaTipo: Integer read FCruceAulaTipo;
property HoraHuecaDesubicada: Integer read GetHoraHuecaDesubicada;
property SesionCortada: Integer read GetSesionCortada;
property CruceProfesorValor: Double read GetCruceProfesorValor;
property CruceAulaTipoValor: Double read GetCruceAulaTipoValor;
property HoraHuecaDesubicadaValor: Double read GetHoraHuecaDesubicadaValor;
property SesionCortadaValor: Double read GetSesionCortadaValor;
property MateriaProhibicion: Integer read GetMateriaProhibicion;
property ProfesorProhibicion: Integer read GetProfesorProhibicion;
property MateriaProhibicionValor: Double read GetMateriaProhibicionValor;
property ProfesorProhibicionValor: Double read GetProfesorProhibicionValor;
property MateriaNoDispersa: Integer read GetMateriaNoDispersa;
property MateriaNoDispersaValor: Double read GetMateriaNoDispersaValor;
property HorarioDetalle: TDynamicSmallintArrayArray read FHorarioDetalle
  write FHorarioDetalle;
property ModeloHorario: TModeloHorario read FModeloHorario;
end;

// Procedimiento que crea una solución aleatoria de un TModeloHorario
procedure CrearAleatorioDesdeModelo(var AObjetoModeloHorario:
  TObjetoModeloHorario; AModeloHorario: TModeloHorario);

// Procedimiento que aplica el operador de cruzamiento sobre dos TObjetoModeloHorario
procedure CruzarIndividuos(var Uno, Dos: TObjetoModeloHorario);

implementation
end.

```

Unidad *KerEvolE*

Unidad que contiene la implementación del algoritmo evolutivo, utilizando el modelo elitista.

```

unit KerEvolE;

interface

uses
  Classes, SysUtils, DB, DBTables, KerModel;

type
  TObjetoModeloHorarioArray = array of TObjetoModeloHorario;
{
  Clase TEvolElitista
  Descripción:
    Implementa el algoritmo evolutivo, utilizando el modelo elitista.
  Miembros privados:
    Los parámetros de configuración del algoritmo evolutivo, y las funciones
    de uso interno.
  Miembros protegidos:
    No los hay.
  Miembros públicos:
    propiedades que devuelven el valor de cada componente de la evaluación
    de la función objetivo en la mejor solución encontrada,
}

```

el valor de la función objetivo, el constructor, funciones que permiten guardar el horario encontrado en una base de datos, métodos que consisten en los operadores genéticos aplicados sobre la población, como son cruce, mutación, etc., *ModeloHorario* al que se aplica el algoritmo evolutivo, método que permite ejecutar el algoritmo evolutivo, Eventos que se disparan cada cierto número de generaciones y cuando mejora la solución.

```

}

TEvolElitista = class
private
  FModeloHorario: TModeloHorario;
  FSemilla1,
  FSemilla2,
  FSemilla3,
  FSemilla4,
  FTamPoblacion,
  FNumGeneracion,
  FNumMaxGeneracion,
  FOrdenMutacion1: Longint;
  FProbCruce,
  FProbMutacion1,
  FProbMutacion2,
  FProbReparacion: Double;
  FPoblacion,
  FNuevaPoblacion: TObjetoModeloHorarioArray;
  FValorArray,
  FAptitudArray,
  FNuevoValorArray,
  FNuevoAptitudArray,
  FRAptitudArray,
  FCAptitudArray: TDynamicDoubleArray;
  FSesionCortadaArray, FNuevoSesionCortadaArray,
  FCruceProfesorArray, FNuevoCruceProfesorArray,
  FCruceAulaTipoArray, FNuevoCruceAulaTipoArray: TDynamicLongintArray;
  FTerminado: Boolean;
  FOnIterar: TNotifyEvent;
  FOnRegistrarMejor: TNotifyEvent;
  procedure Inicializar;
  procedure Evaluar;
  procedure TomarElMejor;
  procedure Elitista;
  procedure Seleccionar;
  procedure Cruzar;
  procedure Mutar;
  procedure CruzarIndividuosInterno(Uno, Dos: Integer);
  function GetMejorValor: Double;
  function GetPromedioValor: Double;
  function GetMejorHoraHuecaDesubicada: Integer;
  function GetMejorCruceProfesor: Integer;
  function GetMejorCruceAulaTipo: Integer;
  function GetMejorSesionCortada: Integer;
  function GetMejorProfesorProhibicionValor: Double;
  function GetMejorMateriaProhibicionValor: Double;
  function GetMejorHoraHuecaDesubicadaValor: Double;
  function GetMejorCruceProfesorValor: Double;
  function GetMejorSesionCortadaValor: Double;
  function GetMejorCruceAulaTipoValor: Double;
  function GetMejorProfesorProhibicion: Integer;
  function GetMejorMateriaProhibicion: Integer;
  function GetMejorMateriaNoDispersa: Integer;
  function GetMejorMateriaNoDispersaValor: Double;
  procedure CopiarIndividuo(Destino, Fuente: Integer);
protected
public
  constructor CrearDesdeModelo(

```

```

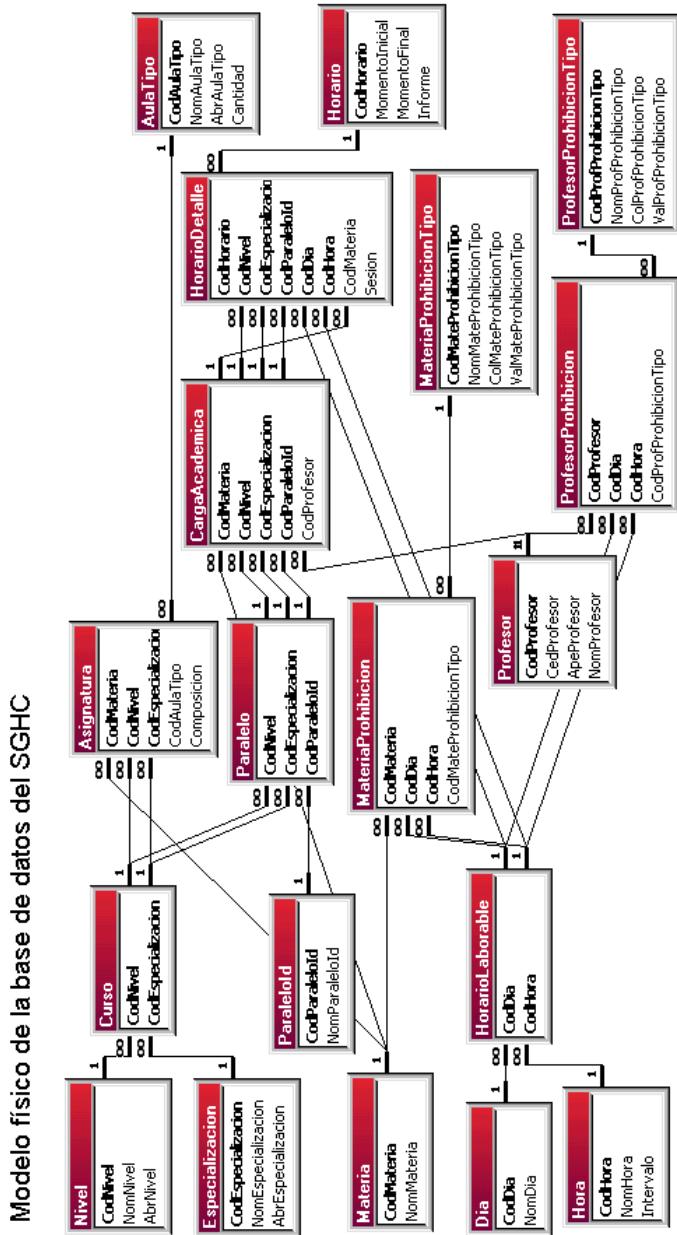
AModeloHorario: TModeloHorario;
ATamPoblacion,
ANumMaxGeneracion: Longint;
AProbCruce,
AProbMutacion1: Double;
AOrdenMutacion1: Longint;
AProbMutacion2,
AProbReparacion: Double);
procedure InvalidarValores;
procedure Configurar(
  ATamPoblacion,
  ANumMaxGeneracion: Longint;
  AProbCruce,
  AProbMutacion1: Double;
  AOrdenMutacion1: Longint;
  AProbMutacion2,
  AProbReparacion: Double);
destructor Destroy; override;
procedure SaveMejorToDatabase(
  CodHorario: Integer;
  MomentoInicial,
  MomentoFinal: TDateTime;
  ADatabase: TDatabase);
procedure LlenarInforme(AInforme: TStrings);
procedure Ejecutar;
procedure DescensoRapidoForzado;
procedure Terminar;
procedure Reparar;
property OnIterar: TNotifyEvent read FOnIterar write FOnIterar;
property OnRegistrarMejor: TNotifyEvent read FOnRegistrarMejor write
  FOnRegistrarMejor;
property NumGeneracion: Longint read FNumGeneracion;
property NumMaxGeneracion: Longint read FNumMaxGeneracion;
property MejorValor: Double read GetMejorValor;
property MejorCruceProfesor: Integer read GetMejorCruceProfesor;
property MejorCruceAulaTipo: Integer read GetMejorCruceAulaTipo;
property MejorHoraHuecaDesubicada: Integer read GetMejorHoraHuecaDesubicada;
property MejorSesionCortada: Integer read GetMejorSesionCortada;
property MejorMateriaProhibicion: Integer read GetMejorMateriaProhibicion;
property MejorProfesorProhibicion: Integer read GetMejorProfesorProhibicion;
property MejorMateriaProhibicionValor: Double read
  GetMejorMateriaProhibicionValor;
property MejorDisponibilidadValor: Double read
  GetMejorProfesorProhibicionValor;
property MejorCruceProfesorValor: Double read GetMejorCruceProfesorValor;
property MejorCruceAulaTipoValor: Double read GetMejorCruceAulaTipoValor;
property MejorMateriaNoDispersaValor: Double read
  GetMejorMateriaNoDispersaValor;
property MejorMateriaNoDispersa: Integer read GetMejorMateriaNoDispersa;
property MejorHoraHuecaDesubicadaValor: Double read
  GetMejorHoraHuecaDesubicadaValor;
property MejorSesionCortadaValor: Double read GetMejorSesionCortadaValor;
property PromedioValor: Double read GetPromedioValor;
property ModeloHorario: TModeloHorario read
  FModeloHorario;
end;  

implementation
end.

```

4.4.3 MODELO FÍSICO DE LA BASE DE DATOS

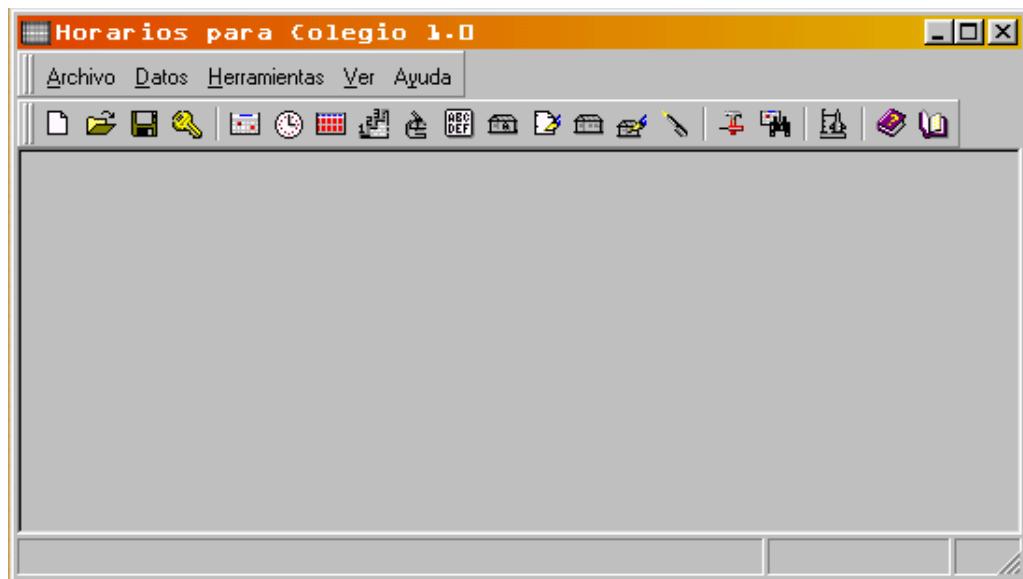
En el modelo físmico de la base de datos, se muestra la configuración final de las tablas y relaciones, como se muestra a continuación:



4.4.4 INTERFAZ DEL USUARIO

La interfaz del usuario es el medio de comunicación entre el usuario y el sistema, por lo tanto la interfaz contiene todos los servicios que se describieron en “servicios que ofrece el sistema”.

Menú Principal



Clase a la que pertenece el formulario: TMainForm.

Submenú Archivo

Contiene todo lo referente al manejo de la base de datos y la opción de salir del programa.

Submenú Datos

Ingreso de los datos del colegio.

Submenú Herramientas

Opciones para configurar el programa y elaborar horarios.

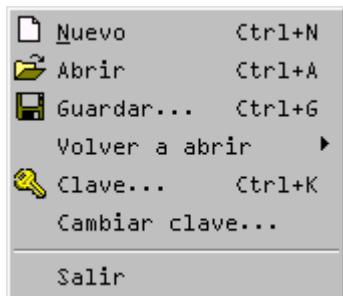
Submenú Ver

Presentación de los resultados obtenidos.

Submenú Ayuda

Ayuda en línea de la aplicación.

1. Submenú Archivo



Submenú Archivo | Nuevo

Inicializa la base de datos.

Submenú Archivo | Abrir

Abre un archivo binario.

Submenú Archivo | Guardar

Guarda la base de datos en un archivo binario.

Submenú Archivo | Volver a abrir

Lista los archivos binarios más recientemente usados.

Submenú Archivo | Clave...

Permite ingresar la clave de acceso a la base de datos.

Submenú Archivo | Cambiar clave...

Permite cambiar la clave de acceso a la base de datos.

Submenú Archivo | Salir

Sale de la aplicación.

1.1 Acción: Nuevo

Servicio del sistema: Nuevo.

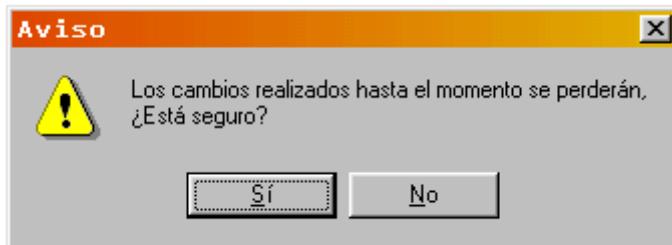
Servicio solicitado: Iniciar la base de datos.

Método: TMainForm.actNewClick.

Acción: Inicia la base de datos.

Servicio en: Diálogo confirmando el borrado de la información existente.

Diálogo confirmando el borrado de la información existente



1.2 Acción: Abrir

Servicio del sistema: Abrir.

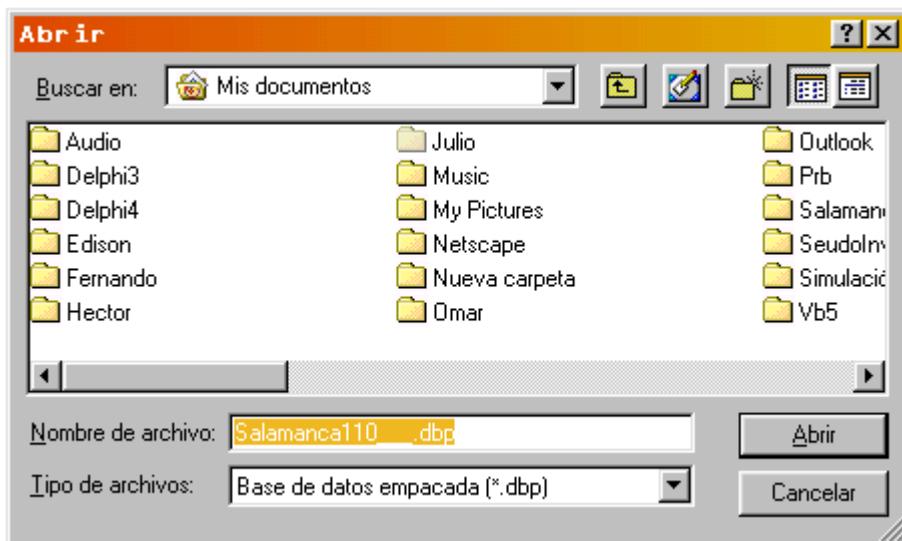
Servicio solicitado: Abrir archivo binario.

Método: TMainForm.actOpenClick.

Acción: Abrir archivo binario.

Servicio en: Diálogo para abrir un archivo.

Diálogo para abrir un archivo



1.3 Acción: Guardar

Servicio del sistema: Guardar.

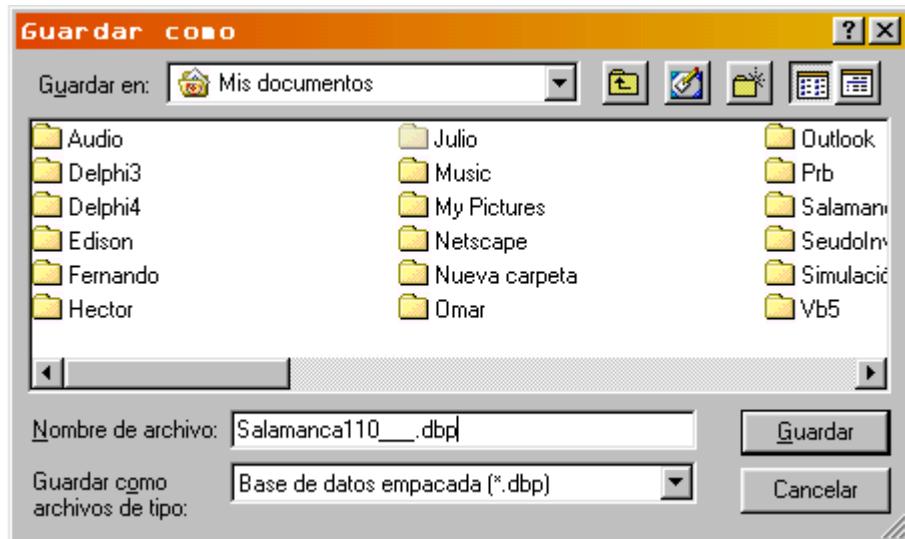
Servicio solicitado: Guardar la base de datos en un archivo binario.

Método: TMainForm.actSaveClick.

Acción: Guarda la base de datos en un archivo binario.

Servicio en: Diálogo para guardar un archivo.

Diálogo para guardar un archivo



1.4 Acción: Clave

Servicio del sistema: Clave.

Servicio solicitado: Ingresar la clave de acceso a la base de datos.

Método: TMainForm.actPasswdClick.

Acción: Ingresar la clave de acceso a la base de datos.

Servicio en: Diálogo solicitando clave.

Diálogo solicitando clave



1.5 Acción: Cambiar clave

Servicio del sistema: Cambiar clave.

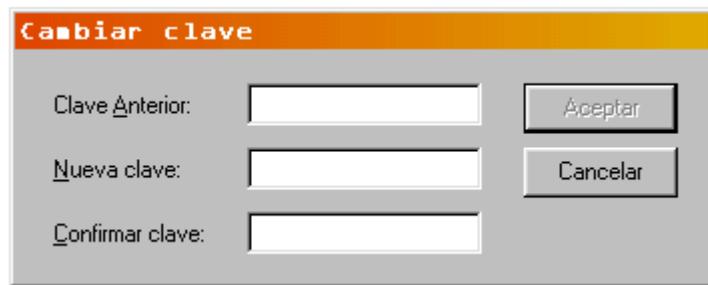
Servicio solicitado: Cambiar la clave de acceso a la base de datos.

Método: TMainForm.actChangePasswdClick.

Acción: Cambiar la clave de acceso a la base de datos.

Servicio en: Diálogo solicitando clave anterior y nueva clave.

Diálogo solicitando clave anterior y nueva clave



1.6 Acción: Salir

Servicio del sistema: Salir.

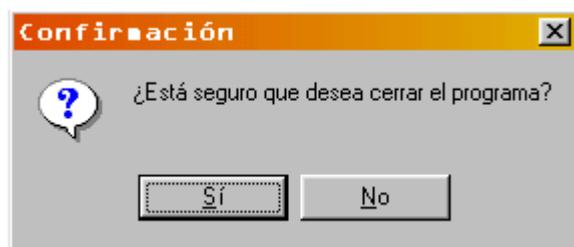
Servicio solicitado: Salir de la aplicación.

Método: TMainForm.actExitClick.

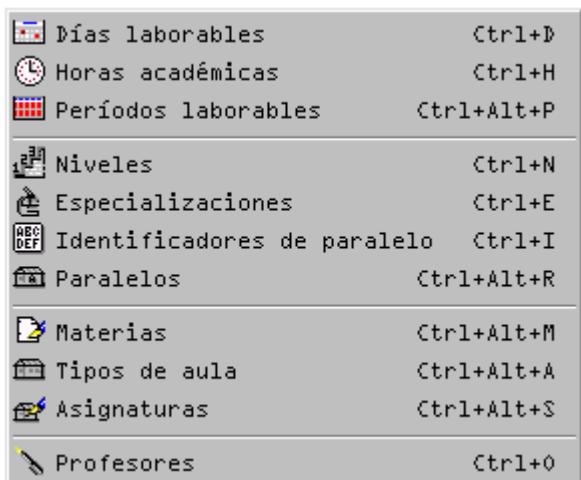
Acción: Salir de la aplicación.

Servicio en: Diálogo confirmando el salir.

Diálogo confirmando el salir



2. Submenú Datos



Submenú Datos | Días laborables

Permite el ingreso de los días laborables.

Submenú Datos | Horas académicas

Permite el ingreso de las horas académicas.

Submenú Datos | Períodos laborables

Permite el ingreso de los períodos laborables.

Submenú Datos | Niveles

Permite el ingreso de los niveles.

Submenú Datos | Especializaciones

Permite el ingreso de las especializaciones.

Submenú Datos | Identificadores de paralelo

Permite el ingreso de los identificadores de paralelo.

Submenú Datos | Paralelos

Permite el ingreso de los paralelos.

Submenú Datos | Materias

Permite el ingreso de las materias.

Submenú Datos | Tipos de aula

Permite el ingreso de los tipos de aula.

Submenú Datos | Asignaturas

Permite el ingreso de las asignaturas.

Submenú Datos | Profesores

Permite el ingreso de los profesores.

2.1 Acción: Días laborables

Servicio del sistema: Días laborables.

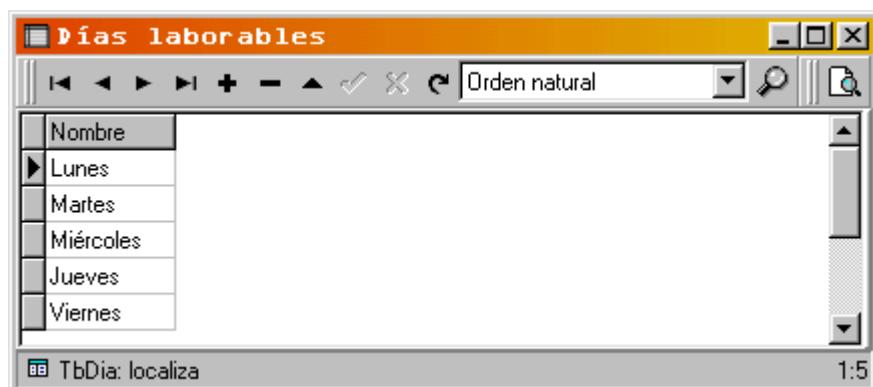
Servicio solicitado: Ingresar los días laborables.

Método: TMainForm.actDiaClick.

Acción: Mostrar formulario.

Servicio en: Formulario para ingresar los días laborables.

Formulario para ingresar los días laborables



Clase a la que pertenece el formulario: TSingleEditorForm.

Argumentos que usan los procesos del formulario:

DataSource = Tabla **Día**.

Servicio del sistema: Presentar.

Servicio interno: Presentar un reporte.

Método: TSsingleEditorForm.btn97ShowClick.

Acción: Presentar un reporte con los días.

2.2 Acción: Horas académicas

Servicio del sistema: Horas académicas.

Servicio solicitado: Ingresar las horas académicas.

Método: TMainForm.actHoraClick.

Acción: Mostrar formulario.

Servicio en: Formulario para ingresar las horas académicas.

Formulario para ingresar las horas académicas

Hora	Intervalo
Primera	07:00 p.m.-07:35 p.m.
Segunda	07:35 p.m.-08:10 p.m.
Tercera	08:10 p.m.-08:45 p.m.
Recreo	08:45 p.m.-09:00 p.m.
Cuarta	09:00 p.m.-09:35 p.m.
Quinta	09:35 p.m.-10:10 p.m.
Sexta	10:10 a.m.-10:45 p.m.

Clase a la que pertenece el formulario: `TSingleEditorForm`.

Argumentos que usan los procesos del formulario:

DataSource = Tabla **Hora**.

Servicio del sistema: Presentar.

Servicio interno: Presentar un reporte.

Método: `TSingleEditorForm.btn97ShowClick`.

Acción: Presentar un reporte con las Horas.

2.3 Acción: Períodos laborables

Servicio del sistema: Períodos laborables.

Servicio solicitado: Ingresar los períodos laborables.

Método: `TMainForm.actPeriodoClick`.

Acción: Mostrar formulario.

Servicio en: Formulario para ingresar los períodos laborables.

Formulario para ingresar los períodos laborables

Clase a la que pertenece el formulario: TCrossManyToManyEditor0Form.

Argumentos que usan los procesos del formulario:

DataSource = Tabla **HorarioLaborable**.

Servicio del sistema: Lista con las opciones *No asignar* y *Asignar*.

Servicio interno: Seleccionar período.

Método: TCrossManyToManyEditor0Form.ListBoxClick.

Acción: Seleccionar período.

Servicio del sistema: Aceptar.

Servicio interno: Aceptar.

Método: TCrossManyToManyEditor0Form.btn97OkClick.

Acción: Guardar información.

Servicio del sistema: Cancelar.

Servicio interno: Cancelar.

Método: TCrossManyToManyEditor0Form.btn97CancelClick.

Acción: Cancelar información ingresada.

2.4 Acción: Niveles

Servicio del sistema: Niveles

Servicio solicitado: Ingresar los niveles.

Método: TMainForm.actNivelClick

Acción: Mostrar formulario

Servicio en: Formulario para ingresar los niveles.

Formulario para ingresar los niveles

Nombre	Abreviatura
Primero	1
Segundo	2
Tercero	3
Cuarto	4
Quinto	5
Sexto	6

Clase a la que pertenece el formulario: `TSingleEditorForm`.

Argumentos que usan los procesos del formulario:

DataSource = Tabla **Nivel**.

Servicio del sistema: Presentar.

Servicio interno: Presentar un reporte.

Método: `TSingleEditorForm.btn97ShowClick`.

Acción: Presentar un reporte con los niveles.

2.5 Acción: Especializaciones

Servicio del sistema: Especializaciones.

Servicio solicitado: Ingresar las especializaciones.

Método: `TMainForm.actEspecializacionClick`.

Acción: Mostrar formulario.

Servicio en: Formulario para ingresar las especializaciones.

Formulario para ingresar las especializaciones

Nombre	Abreviatura
Básico	BASICO
Contabilidad	CONTA
Secretariado	SECRE
Informática	INFO

Clase a la que pertenece el formulario: TSsingleEditorForm.

Argumentos que usan los procesos del formulario:

DataSource = Tabla **Especialización**.

Servicio del sistema:  Presentar.

Servicio interno: Presentar un reporte.

Método: TSsingleEditorForm.btn97ShowClick.

Acción: Presentar un reporte con las especializaciones.

2.6 Acción: Identificadores de paralelo

Servicio del sistema:  Identificadores de paralelo.

Servicio solicitado: Ingresar los identificadores de paralelo.

Método: TMainForm.actParaleloIdClick.

Acción: Mostrar formulario.

Servicio en: Formulario para ingresar los identificadores de paralelo.

Formulario para ingresar los identificadores de paralelo



Clase a la que pertenece el formulario: TSsingleEditorForm.

Argumentos que usan los procesos del formulario:

DataSource = Tabla **ParaleloId**.

Servicio del sistema:  Presentar.

Servicio interno: Presentar un reporte.

Método: TSsingleEditorForm.btn97ShowClick.

Acción: Presentar un reporte con los identificadores de paralelo.

2.7 Acción: Paralelos

Servicio del sistema:  Paralelos.

Servicio solicitado: Ingresar los paralelos.

Método: TMainForm.actParaleloClick.

Acción: Mostrar formulario.

Servicio en: Formulario para ingresar los paralelos.

Formulario para ingresar los paralelos

	Nivel	Espec.
1		BASICO
2		BASICO
3		BASICO
4		CONTA
4		SECRE

Clase a la que pertenece el formulario: TParaleloForm.

Argumentos que usan los procesos del formulario:

DataSource = Tabla **Curso**.

DataSourceDetail = Tabla **Paralelo**.

DataSourceList = Tabla **ParaleloId**.

Servicio del sistema: Presentar.

Servicio interno: Presentar un reporte.

Método: TSsingleEditorForm.btn97ShowClick.

Acción: Presentar un reporte con los cursos y paralelos.

Servicio del sistema: Seleccionar paralelos.

Servicio interno: Seleccionar paralelos.

Método: Interno del componente.

Acción: Seleccionar paralelos.

2.8 Acción: Materias

Servicio del sistema: Materias.

Servicio solicitado: Ingresar las materias.

Método: TMainForm.actMateriaClick.

Acción: Mostrar formulario.

Servicio en: Formulario para ingresar las materias.

Formulario para ingresar las materias:

Clase a la que pertenece el formulario: TMateriaForm.

Argumentos que usan los procesos del formulario:

DataSource = Tabla **Materia**.

Servicio del sistema: Presentar.

Servicio interno: Presentar un reporte.

Método: TSsingleEditorForm.btn97ShowClick.

Acción: Presentar un reporte con las Materias.

Servicio del sistema: Prohibiciones de materia.

Servicio interno: Ingresar el horario de prohibiciones de materia.

Método: TMateriaForm.btn97MateriaProhibicionClick.

Acción: Mostrar formulario.

Servicio en: Formulario para ingresar el horario de prohibiciones de la materia.

Formulario para ingresar el horario de prohibiciones de la materia

Clase a la que pertenece el formulario: TCrossManyToManyEditorRForm.

Argumentos que usan los procesos del formulario:

DataSource = Tabla **MateriaProhibicion**.

Servicio del sistema: Lista con las opciones *Inadecuado* e *Imposible*.

Servicio interno: Seleccionar tipo de prohibición de materia.

Método: TCrossManyToManyEditorRForm.ListBoxClick.

Acción: Seleccionar tipo de prohibición de materia.

Servicio del sistema: Aceptar.

Servicio interno: Aceptar.

Método: TCrossManyToManyEditorRForm.btn97OkClick.

Acción: Guardar información.

Servicio del sistema: Cancelar.

Servicio solicitado: Cancelar.

Método: TCrossManyToManyEditorRForm.btn97CancelClick.

Acción: Cancelar información ingresada.

2.9 Acción: Tipos de aula

Servicio del sistema: Tipos de aula.

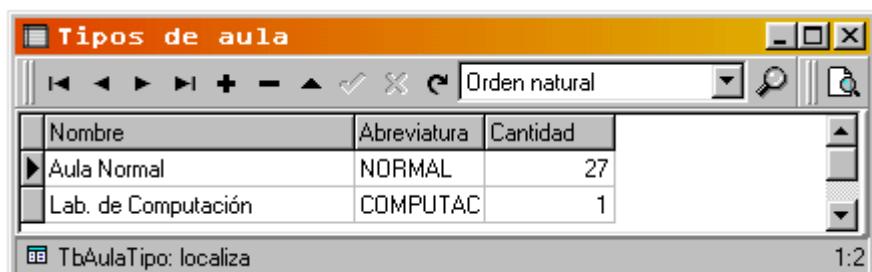
Servicio solicitado: Tipos de aula.

Método: TMateriaForm.actAulaTipoClick.

Acción: Mostrar formulario.

Servicio en: Formulario para ingresar los tipos de aula.

Formulario para ingresar los tipos de aula



Clase a la que pertenece el formulario: TSingleEditorForm.

Argumentos que usan los procesos del formulario:

DataSource = Tabla **ParaleloId**.

Servicio del sistema: Presentar.

Servicio interno: Presentar un reporte.

Método: TSsingleEditorForm.btn97ShowClick.

Acción: Presentar un reporte con los identificadores de paralelo.

2.10 Acción: Asignaturas

Servicio del sistema: Ingresar las asignaturas.

Servicio solicitado: Ingresar las asignaturas.

Método: actAsignaturaClick.

Acción: Mostrar formulario.

Servicio en: Formulario para ingresar las asignaturas.

Formulario para ingresar las asignaturas

Materia	Nivel	Espec.	Tipo aula	Composición
Aso. de Clase	1	BASICO	NORMAL	1
Aso. de Clase	2	BASICO	NORMAL	1
Aso. de Clase	3	BASICO	NORMAL	1
Contabilidad	4	CONTA	NORMAL	3.3.2.2
Contabilidad	4	SECRE	NORMAL	2

Clase a la que pertenece el formulario: TSsingleEditorForm.

Argumentos que usan los procesos del formulario:

DataSource = Tabla **Asignatura**.

Servicio del sistema: Presentar.

Servicio interno: Presentar un reporte.

Método: TSsingleEditorForm.btn97ShowClick.

Acción: Presentar un reporte con las Asignaturas.

2.11 Acción: Profesores

Servicio del sistema: Profesores.

Servicio solicitado: Ingresar los profesores.

Método: TProfesorForm.actProfesorClick.

Acción: Mostrar formulario.

Servicio en: Formulario para ingresar los profesores.

Formulario para ingresar los profesores

Cédula	Apellido	Nombre
16	Aguas	Patricio
26	Albuja	Ramiro
46	Baca	Patricio
50	Baquero	Rosario
34	Baquero	Zuly

Clase a la que pertenece el formulario: TProfesorForm.

Argumentos que usan los procesos del formulario:

DataSource = Tabla **Profesor**.

Servicio del sistema: Presentar.

Servicio interno: Presentar un reporte.

Método: TSsingleEditorForm.btn97ShowClick.

Acción: Presentar un reporte con los profesores.

Servicio del sistema: Ingresar prohibición de profesores.

Servicio interno: Ingresar prohibición de profesores.

Método: TProfesorForm.btn97ProfesorProhibicionClick.

Acción: Mostrar formulario.

Servicio en: Formulario para ingresar la prohibición de profesores.

Servicio del sistema: Ingresar la carga académica del profesor.

Servicio interno: Ingresar la carga académica del profesor.

Método: TProfesorForm.btn97CargaAcademicaClick.

Acción: Mostrar formulario.

Servicio en: Formulario para ingresar la carga académica del profesor.

Formulario para ingresar la prohibición de profesores

	Lunes	Martes	Miércoles	Jueves	Viernes
Primera					
Segunda					
Tercera					
Recreo					
Cuarta					
Quinta					
Sexta	No Gusta	No Gusta	No Gusta	No Gusta	No Gusta

Clase a la que pertenece el formulario: TCrossManyToManyEditorRForm

Argumentos que usan los procesos del formulario:

DataSource = Tabla **Profesor**.

DataSourceDetail = Tabla **ProfesorProhibicion**.

DataSourceList = Tabla **ProfesorProhibicionTipo**.

Servicio del sistema: Lista con las opciones *No gusta* y *No puede*.

Servicio interno: Seleccionar tipo de prohibición de profesor.

Método: TCrossManyToManyEditorRForm.ListBoxClick.

Acción: Seleccionar tipo de prohibición de materia.

Servicio del sistema: Aceptar.

Servicio interno: Aceptar.

Método: TCrossManyToManyEditorRForm.btnExit97OkClick.

Acción: Guardar información.

Servicio del sistema: Cancelar.

Servicio interno: Cancelar.

Método: TCrossManyToManyEditorRForm.btnExit97CancelClick.

Acción: Cancelar información ingresada.

Formulario para ingresar la carga académica del profesor

	Materia	Nivel	Espec.	Par.	Composición	Duración
►	CC.SS.	2	BASICO	A	1.1.1.1.1	05
	CC.SS.	2	BASICO	B	1.1.1.1.1	05
	E.S.	4	CONTA	A	1.1	02
	E.S.	4	CONTA	B	1.1	02

Clase a la que pertenece el formulario: TSsingleEditorForm.

Argumentos que usan los procesos del formulario:

MasterDataSource = Tabla **Profesor**.

DataSource = Tabla **CargaAcadémica**.

Servicio del sistema:  Presentar.

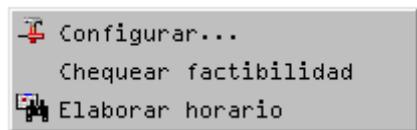
Servicio interno: Presentar un reporte.

Método: TSingleEditorForm.btn97ShowClick.

Acción: Presentar un reporte con los profesores.

Observación: La tabla **CargaAcademica** está sincronizada con la tabla **Profesor**.

3. Submenú Herramientas



Submenú Herramientas | *Configurar...*

Permite establecer los parámetros de configuración.

Submenú Herramientas | *Chequear factibilidad*

Chequea la factibilidad de los datos para generar el horario.

Submenú Herramientas | *Elaborar horario*

Permite elaborar un horario.

3.1 Acción: *Configurar*

Servicio del sistema:  Configurar.

Servicio solicitado: Establecer los parámetros de configuración.

Método: actConfigurarClick.

Servicio en: Formulario para establecer los parámetros de configuración.

Formulario para establecer los parámetros de configuración

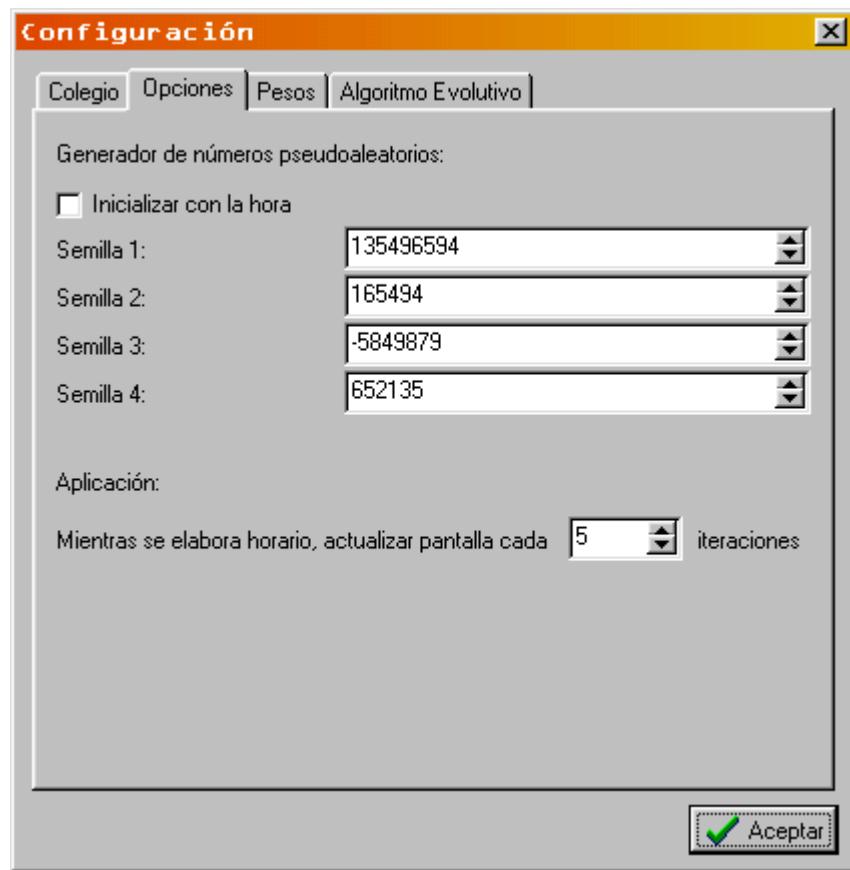
Categoría Datos del colegio

Configuración

Colegio | Opciones | Pesos | Algoritmo Evolutivo | X

Colegio:	SALAMANCA		
Año Lectivo:	1998-1999		
Autoridad:	Edgar Velasco	Cargo:	Rector
Responsable:	Edison Mera	Cargo:	Prof. Suplente
Carga Máxima por Profesor:	23	<input style="float: right; margin-right: 10px;" type="button" value="▲"/> <input style="float: right; margin-right: 10px;" type="button" value="▼"/>	
Horario seleccionado: 23			
Comentarios: Pruebas realizadas en el Colegio Salamanca			
<input checked="" type="checkbox"/> Aceptar			

Categoría **Opciones** del sistema:



Categoría **Pesos** de la función objetivo:

Configuración

Colegio | **Opciones** | **Pesos** | **Algoritmo Evolutivo**

Cruce de profesores:	1405
Cruce de aulas:	670
Horas huecas desubicadas:	997
Materias cortadas:	756
Materias no dispersas:	19

Prohibiciones de materia:

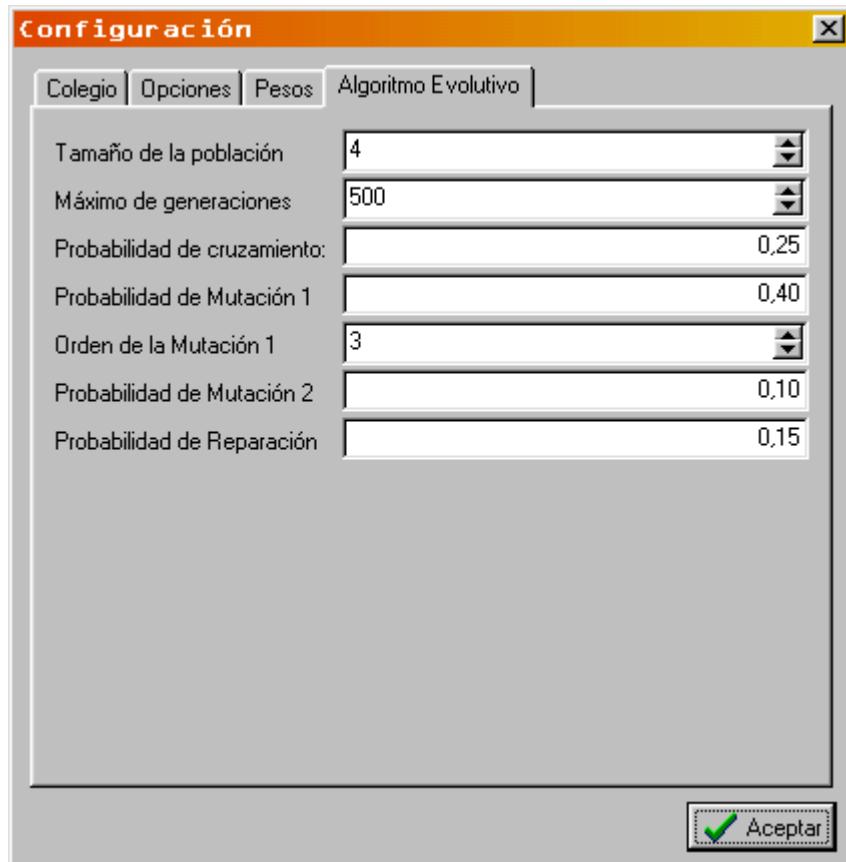
Nombre	Color	Valor
Inadecuado	65280	99
Imposible	255	3000

Prohibiciones de profesor:

Nombre	Color	Valor
No Gusta	65280	123
No Puede	255	3000

Aceptar

Categoría Parámetros del **algoritmo evolutivo**:



Clase a la que pertenece el diálogo: TConfiguracionForm.

Servicio de sistema: Aceptar.

Servicio interno: Aceptar.

Acción: Cerrar formulario.

Observación: La información se guarda en el archivo de configuración con ayuda del componente FormStorage, por lo que no es necesario programar ningún método.

3.2 Acción: *Chequear factibilidad*

Servicio solicitado: Chequear la factibilidad de los datos para generar el horario

Método: actChequearFactibilidadClick.

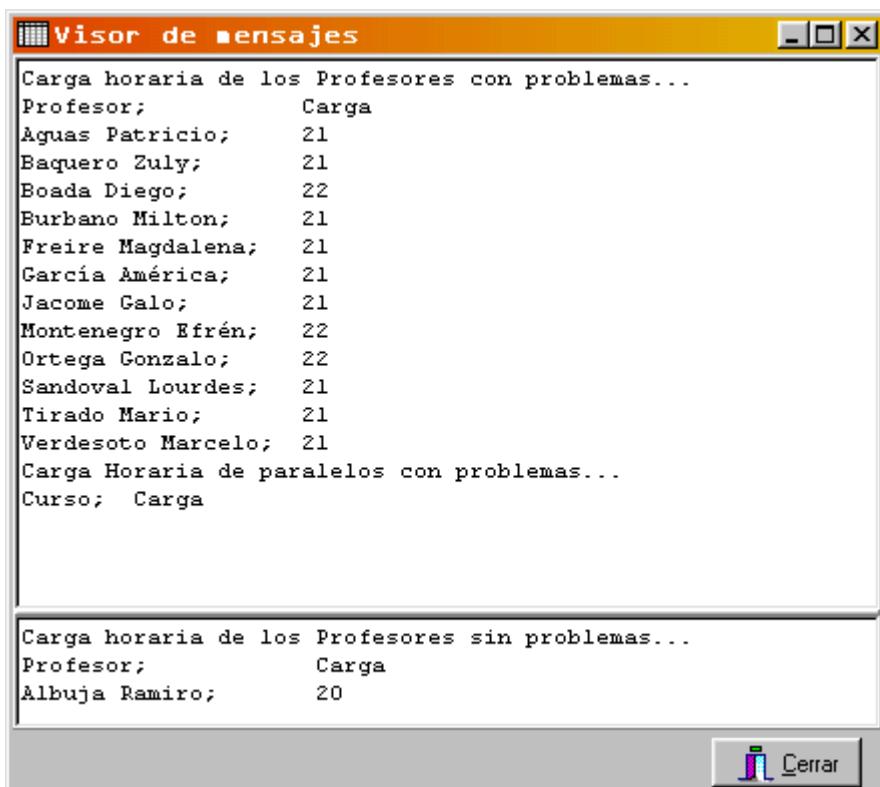
Servicio en: Dos casos excluyentes:

- 1) Diálogo indicando: “no se encontraron errores, está listo para generar horario”.
- 2) Formulario mostrando los problemas encontrados en la base de datos.

Diálogo que indica que se está listo para generar el horario:



Formulario de problemas encontrados en la base de datos:



Clase a la que pertenece el diálogo: TLogisticForm.

Servicio de sistema: Cerrar.

Servicio interno: Cerrar.

Acción: Cerrar formulario.

3.3 Acción: *Elaborar horario*

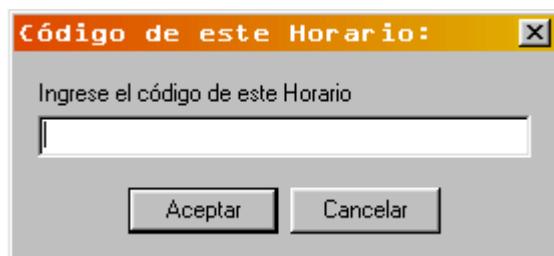
Servicio del sistema:  Elaborar un horario.

Servicio solicitado: Elaborar un horario.

Método: actElaborarHorarioClick.

Servicio en: Diálogo pidiendo el código del horario a elaborar, y al aceptar el diálogo, se llama al formulario que muestra el progreso de la elaboración del horario.

Diálogo pidiendo el código del horario a elaborar



Formulario mostrando el progreso de la elaboración del horario

Clase a la que pertenece el formulario: TProgressForm.

Servicio del sistema: Cancelar.

Servicio interno: Cancelar.

Aplicado a: Botón.

Acción: Cancelar Elaboración del horario.

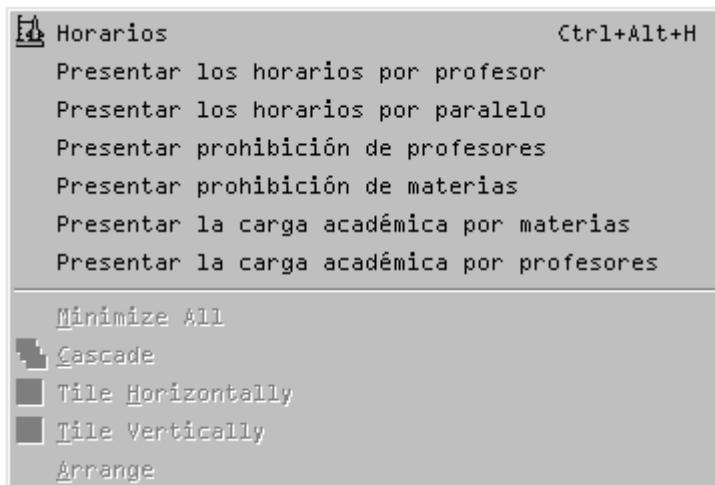
Servicio del sistema: Finalizar.

Servicio interno: Finalizar.

Aplicado a: Botón.

Acción: Almacenar el horario encontrado al momento.

4. Submenú Ver



Submenú Ver|Horarios

Muestra los horarios que han sido elaborados.

Submenú Ver|Presentar los horarios por profesor

Presenta los horarios por profesor.

Submenú Ver|Presentar los horarios por paralelo

Presenta los horarios por paralelo.

Submenú Ver|Presentar prohibición de profesores

Presenta la prohibición de profesores.

Submenú Ver|Presentar prohibición de materias

Presenta la prohibición de materias.

Submenú Ver|Presentar la carga académica por materias

Presenta la carga académica por materias.

Submenú Ver|Presentar la carga académica por profesores

Presenta la carga académica por profesores.

4.1 Acción: Horarios

Servicio del sistema:  Horarios.

Servicio solicitado: Ver horarios.

Método: actHorarioClick.

Servicio en: Formulario para ver los horarios.

Formulario para ver los horarios

Clase a la que pertenece el formulario: THorarioForm.

Argumentos que usan los procesos del formulario:

DataSource = Tabla **Profesor**.

Servicio del sistema: Presentar.

Servicio interno: Presentar un reporte.

Método: TSingleEditorForm.btn97ShowClick.

Acción: Presentar un reporte con los profesores.

Servicio del sistema: Seleccionar un horario.

Servicio interno: Seleccionar un horario.

Método: TSingleEditorForm.btn97SeleccionarHorarioClick.

Acción: Seleccionar un horario.

Servicio del sistema: Horario de profesores.

Servicio interno: Horario de profesores.

Método: THorarioForm.btn97HorarioProfesorClick.

Acción: Mostrar formulario.

Servicio en: Formulario que muestra el horario de profesores.

Servicio del sistema: Horario de paralelos.

Servicio interno: Horario de paralelos.

Método: THorarioForm.btn97HorarioParaleloClick.

Acción: Mostrar formulario.

Servicio en: Formulario que muestra el horario de paralelos.

Servicio del sistema: Cruce de profesores.

Servicio interno: Cruce de profesores.

Método: THorarioForm.btn97CruceProfesorClick.

Acción: Mostrar formulario.

Servicio en: Formulario que muestra el cruce de profesores.

Servicio del sistema:  Cruce de materias.

Servicio interno: Cruce de materias.

Método: THorarioForm.btn97CruceMateriaClick.

Acción: Mostrar formulario.

Servicio en: Formulario que muestra el cruce de materias.

Servicio del sistema:  Cruce de aulas.

Servicio interno: Cruce de aulas.

Método: THorarioForm.btn97CruceAulaClick.

Acción: Mostrar formulario.

Servicio en: Formulario que muestra el cruce de aulas.

Servicio del sistema:  Prohibición de materias no respetadas.

Servicio interno: Prohibición de materias no respetadas.

Método: THorarioForm.btn97MateriaProhibicionNoRespetadaClick.

Acción: Mostrar formulario.

Servicio en: Formulario que muestra la prohibición de materias no respetadas.

Servicio del sistema:  Prohibición de profesores no respetadas.

Servicio interno: Prohibición de profesores no respetadas.

Método: THorarioForm.btn97ProfesorProhibicionNoRespetadaClick.

Acción: Mostrar formulario.

Servicio en: Formulario que muestra la prohibición de profesores no respetadas.

Formulario que muestra el horario de profesores



	Lunes	Martes	Miércoles	Jueves	Viernes
Primera	5 A CONTA	4 S SECRE	5 D INFO	4 B CONTA	
Segunda	2 B BASICO	2 B BASICO	2 B BASICO		
Tercera		4 B CONTA	5 C CONTA	4 S SECRE	2 B BASICO
Recreo					
Cuarta	5 B CONTA	2 A BASICO		4 A CONTA	2 A BASICO
Quinta		4 A CONTA	2 A BASICO	2 A BASICO	2 A BASICO
Sexta				2 B BASICO	

Clase a la que pertenece el formulario: THorarioProfesorForm.

Argumentos que usan los procesos del formulario:

DataSource = Tabla **HorarioDetalle**.

Servicio del sistema: Lista desplegable con los profesores.

Servicio interno: Seleccionar profesor.

Método: interno del componente.

Acción: Seleccionar profesor.

Servicio del sistema: Lista desplegable indicando qué mostrar en el horario.

Servicio interno: Seleccionar qué mostrar en el horario, Materias, Nivel-Paralelo, Nivel-Especialización-Paralelo o Nivel-Paralelo-Especialización.

Método: interno del componente.

Acción: Seleccionar qué mostrar en el horario.

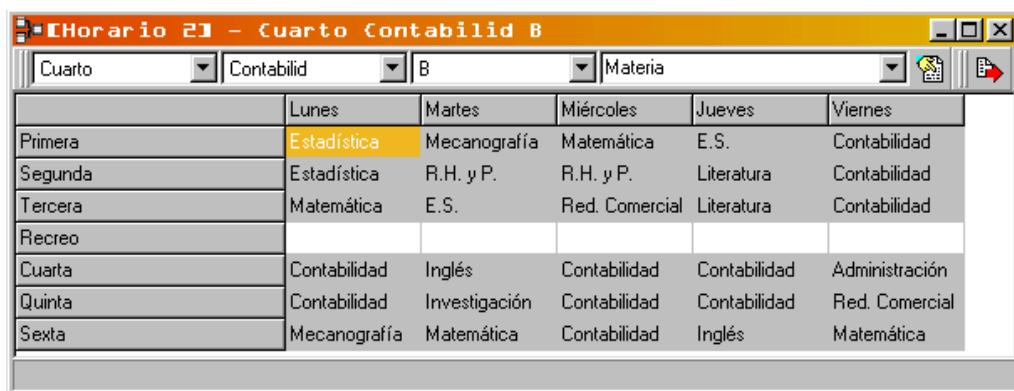
Servicio del sistema:  Mostrar.

Servicio interno: Mostrar.

Método: THorarioProfesorForm.btn97MostrarClick.

Acción: Horario de profesores, tomando los datos marcados en las listas desplegables como argumentos.

Formulario que muestra el horario de paralelos



Cuarto	Lunes	Martes	Miércoles	Jueves	Viernes
Primera	Estadística	Mecanografía	Matemática	E.S.	Contabilidad
Segunda	Estadística	R.H. y P.	R.H. y P.	Literatura	Contabilidad
Tercera	Matemática	E.S.	Red. Comercial	Literatura	Contabilidad
Recreo					
Cuarta	Contabilidad	Inglés	Contabilidad	Contabilidad	Administración
Quinta	Contabilidad	Investigación	Contabilidad	Contabilidad	Red. Comercial
Sexta	Mecanografía	Matemática	Contabilidad	Inglés	Matemática

Clase a la que pertenece el formulario: TParaleloHorarioForm.

Argumentos que usan los procesos del formulario:

DataSource = Tabla **HorarioDetalle**.

Servicio del sistema: Lista desplegable con los niveles.

Servicio interno: Seleccionar nivel.

Método: interno del componente.

Acción: Seleccionar nivel.

Servicio del sistema: Lista desplegable con las especializaciones.

Servicio interno: Seleccionar especialización.

Método: interno del componente.

Acción: Seleccionar especialización.

Servicio del sistema: Lista desplegable con los identificadores de paralelo.

Servicio interno: Seleccionar identificador de paralelo.

Método: interno del componente.

Acción: Seleccionar identificador de paralelo.

Servicio del sistema: Lista desplegable indicando qué mostrar en el horario.

Servicio interno: Seleccionar qué mostrar en el horario, Materia o Profesor.

Método: interno del componente.

Acción: Seleccionar qué mostrar en el horario.

Servicio del sistema:  Mostrar.

Servicio interno: Mostrar.

Método: THorarioParaleloForm.btn97MostrarClick.

Acción: Horario de paralelos, tomando los datos marcados en las listas desplegables como argumentos.

Servicio del sistema:  Intercambiar.

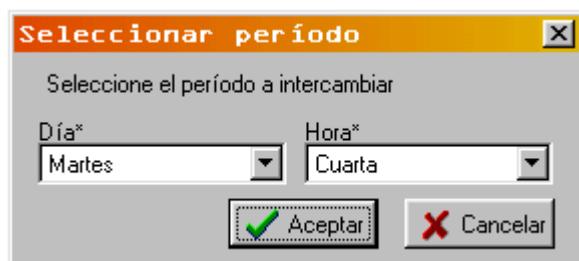
Servicio interno: Intercambiar períodos.

Método: THorarioParaleloForm.btn97MostrarClick.

Acción: Intercambiar períodos.

Servicio en: Diálogo solicitando el período que será intercambiado con el que ya está seleccionado en la cuadrícula.

Diálogo solicitando el período que será intercambiado con el que ya está seleccionado en la cuadrícula



Formulario que muestra el cruce de profesores



Clase a la que pertenece el formulario: TMasterDetailEditorForm.

Argumentos que usan los procesos del formulario:

DataSource = Consulta **QuCruceProfesor**, que permite obtener los profesores que tienen cruces y en qué período del horario laborable.

DataSourceDetail = Consulta **QuCruceProfesorDetail**, que permite obtener de cada uno de los profesores, los paralelos y las materias con cruces en caso de haberlos.

Servicio del sistema:  Presentar.

Servicio interno: Presentar un reporte.

Método: TMasterDetailEditorForm.btn97ShowClick.

Acción: Presentar un reporte con los cruces.

Formulario que muestra el cruce de materias



Clase a la que pertenece el formulario: TMasterDetailEditorForm.

Argumentos que usan los procesos del formulario:

DataSource = Consulta **QuCruceMateria**, que permite obtener las materias que tienen cruces.

DataSourceDetail = Consulta **QuCruceMateriaDetail**, que permite obtener de cada materia, el paralelo y el período del horario laborable en que presenta cruces en caso de haberlos.

Servicio del sistema:  Presentar.

Servicio interno: Presentar un reporte.

Método: TMasterDetailEditorForm.btn97ShowClick.

Acción: Presentar un reporte con los cruces.

Formulario que muestra el cruce de aulas

Cruce de aulas del mismo tipo

Tipo aula	Día	Hora	Usadas	Cruces
► COMPUTAC	Martes	Quinta	2	1
► COMPUTAC	Miércoles	Sexta	2	1

Materia	Nivel	Espec.	Par.
► Lab. Programac.	5	INFO	D
Lab. Programac.	6	INFO	D

QuCruceAula: localiza 13:2

Clase a la que pertenece el formulario: TMasterDetailEditorForm.

Argumentos que usan los procesos del formulario:

DataSource = Consulta **QuCruceAula**, que permite obtener las aulas que tienen cruces y en qué período del horario laborable.

DataSourceDetail = Consulta **QuCruceProfesorDetail**, que permite obtener de cada aula, los paralelos y las materias en que presenta cruces en caso de haberlos.

Servicio del sistema:  Presentar.

Servicio interno: Presentar un reporte.

Método: TMasterDetailEditorForm.btn97ShowClick.

Acción: Presentar un reporte con los cruces.

Formulario que muestra la prohibición de materias no respetadas

Prohibiciones de materia No Respetadas

Tipo prohib. mat.	Materia	Día	Hora	Nivel	Espec.	Par.
Inadecuado	Contabilidad	Lunes	Sexta	5	CONTA	A
► Inadecuado	Contabilidad	Lunes	Sexta	6	CONTA	A
Inadecuado	Contabilidad	Lunes	Sexta	6	CONTA	C
Inadecuado	Contabilidad	Martes	Sexta	5	CONTA	A
Inadecuado	Contabilidad	Miércoles	Sexta	5	CONTA	B
Inadecuado	Contabilidad	Jueves	Sexta	4	CONTA	A
Inadecuado	Contabilidad	Jueves	Sexta	5	CONTA	C
Inadecuado	Contabilidad	Jueves	Sexta	6	CONTA	C
Inadecuado	Contabilidad	Jueves	Sexta	6	SECRE	S
Inadecuado	Contabilidad	Viernes	Sexta	4	CONTA	C
Inadecuado	Lab. Programac.	Lunes	Segunda	5	INFO	D
Inadecuado	Lab. Programac.	Lunes	Tercera	5	INFO	D
Inadecuado	Lab. Programac.	Miércoles	Primera	6	INFO	D

RxQuHorarioDetalleMateriaProhibicion: localiza 2:29

Clase a la que pertenece el formulario: TSsingleEditorForm.

Argumentos que usan los procesos del formulario:

DataSource = Consulta QuMateriaProhibicionNoRespetada, que retorna las prohibiciones de materia no respetadas.

Servicio del sistema:  Presentar

Servicio interno: Presentar un reporte.

Método: TSsingleEditorForm.btn97ShowClick.

Acción: Presentar un reporte con las prohibiciones de materia no respetadas.

Formulario que muestra la prohibición de profesores no respetadas



Tipo prohib. prof.	Apellido	Nombre	Día	Hora	Nivel	Espec.	Par.
No Gusta	Albuja	Ramiro	Miércoles	Sexta	2	BASICO	A
No Gusta	Baquero	Zuly	Miércoles	Sexta	3	BASICO	A
No Gusta	Barragán	Miriam	Martes	Quinta	6	INFO	D
No Gusta	Barragán	Miriam	Jueves	Quinta	3	BASICO	A

Clase a la que pertenece el formulario: TSsingleEditorForm.

Argumentos que usan los procesos del formulario:

DataSource = Consulta QuMateriaProhibicionNoRespetada, que retorna las prohibiciones de materia no respetadas.

Servicio del sistema:  Presentar

Servicio interno: Presentar un reporte.

Método: TSsingleEditorForm.btn97ShowClick.

Acción: Presentar un reporte con los profesores.

4.2 Acción: Presentar los horarios por profesor

Servicio del sistema: Presentar los horarios por profesor.

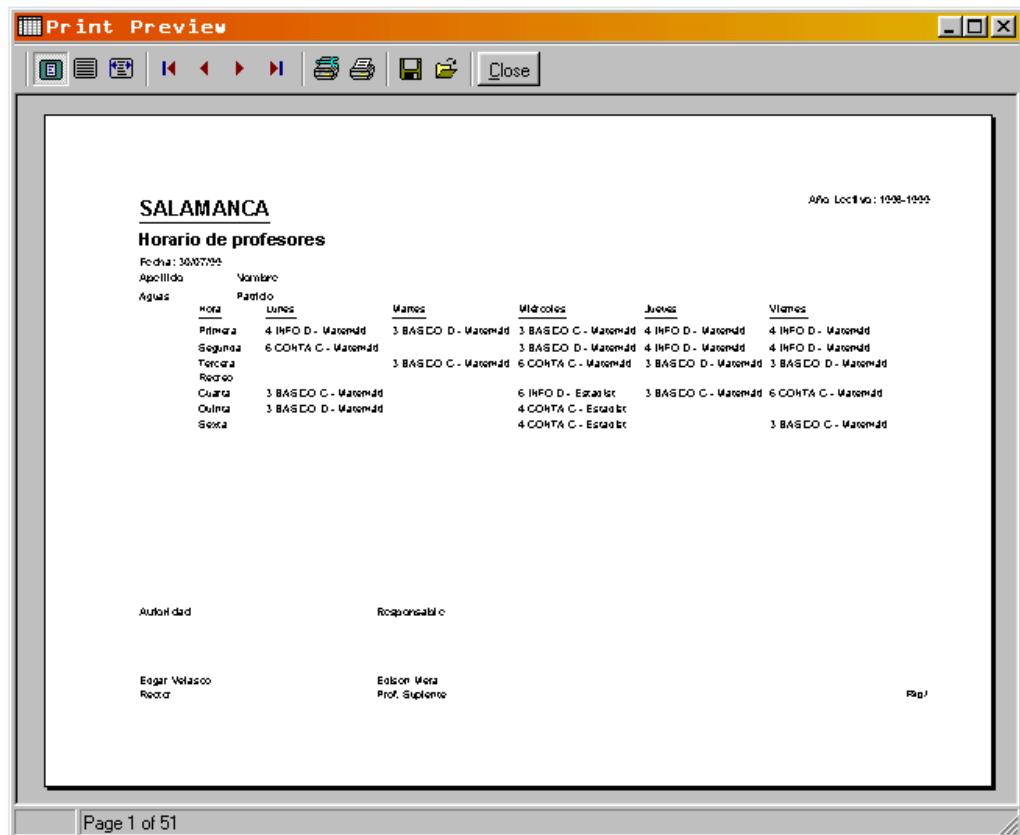
Servicio solicitado: Mostrar un reporte con los horarios por profesor.

Método: TMainForm.actPresentarProfesorHorarioClick.

Acción: Presentar los horarios por profesor.

Servicio en: Reporte.

Reporte



4.3 Acción: Presentar los horarios por paralelo

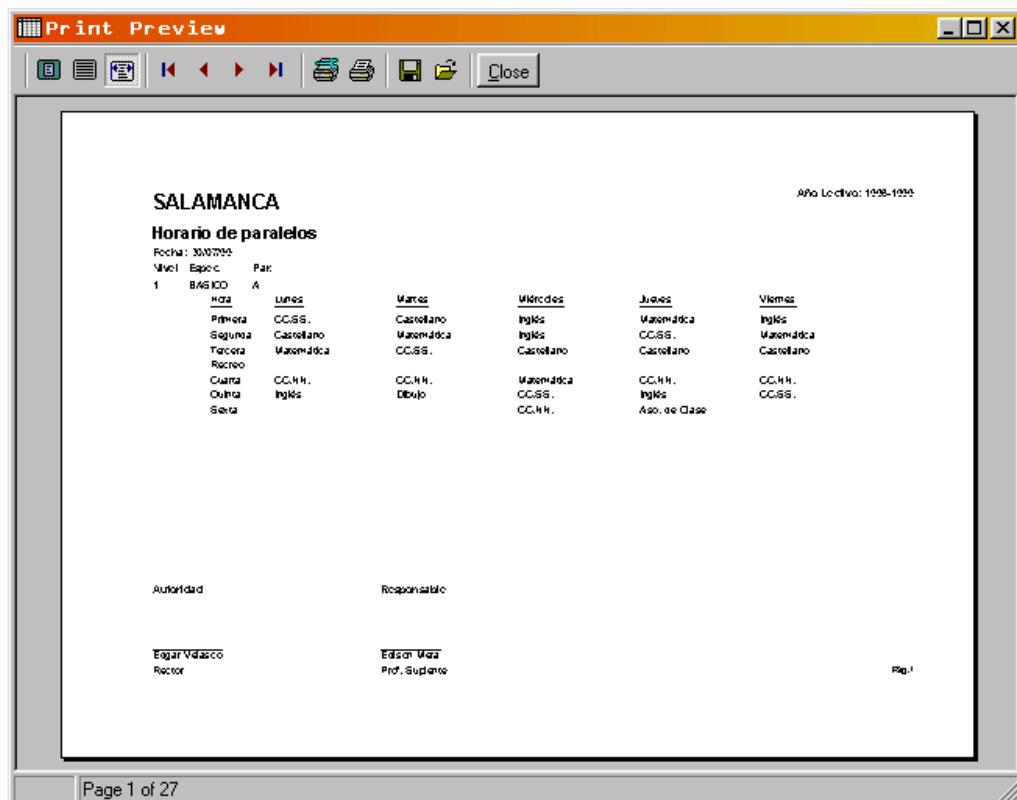
Servicio del sistema: Presentar los horarios por paralelo.

Servicio solicitado: Mostrar un reporte con los horarios por paralelo.

Método: TMainForm.actPresentarParaleloHorarioExecute.

Servicio en: Reporte.

Reporte



4.4 Acción: Presentar prohibición de profesores

Servicio del sistema: Presentar prohibición de profesores.

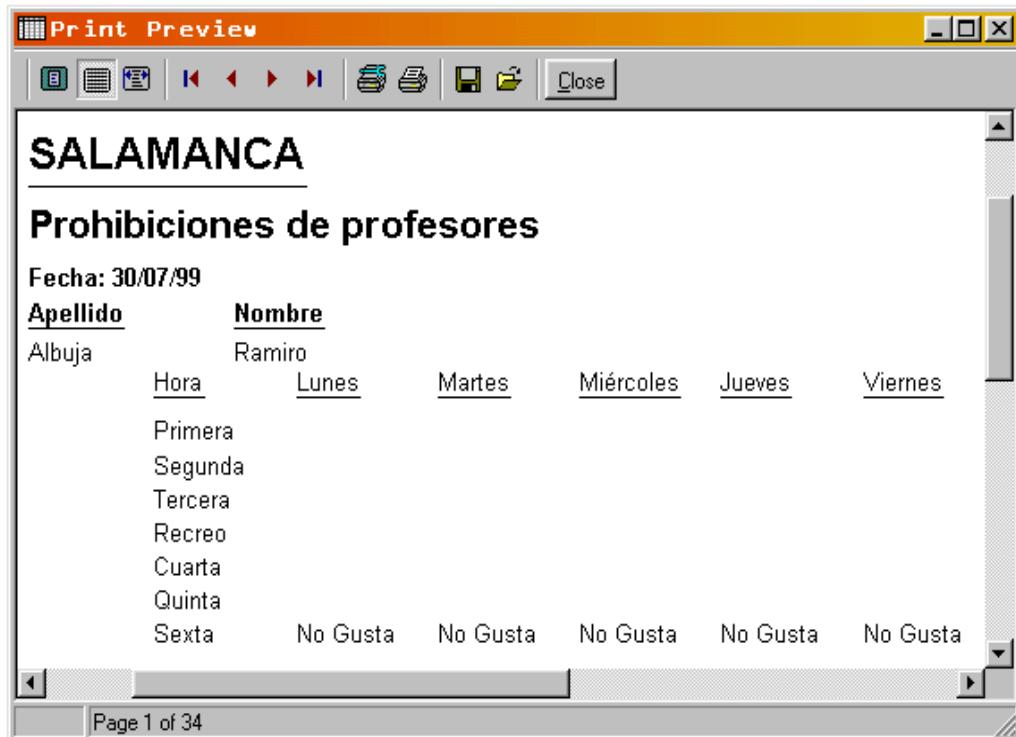
Servicio solicitado: Mostrar un reporte con las prohibiciones de profesores.

Método: TMainForm.actPresentarProfesorProhibicionExecute.

Acción: Presentar prohibición de profesores.

Servicio en: Reporte.

Reporte



4.5 Acción: Presentar prohibición de materias

Servicio del sistema: Presentar prohibición de materias.

Servicio solicitado: Mostrar un reporte con las prohibiciones de materia.

Método: TMainForm.actPresentarMateriaProhibicionExecute.

Acción: Presentar prohibición de materias.

Servicio en: Reporte.

Reporte



4.6 Acción: Presentar la carga académica por materias

Servicio del sistema: Presentar carga académica por materias.

Servicio solicitado: Mostrar un reporte con la carga académica por materias.

Método: TMainForm.actPresentarCargaAcademicaMateriaClick.

Acción: Presentar la carga académica por materias.

Servicio en: Reporte.

Reporte

SALAMANCA

Carga académica por materias

Fecha: 30/07/99

Nombre

Administración

Nivel	Espec.	Par.	Profesor	Duración
4	CONTA	A	Guerrero Edwin	1
4	CONTA	B	Guerrero Edwin	1
4	CONTA	C	Montenegro Beatriz	1
4	SECRE	S	Espinosa Fabián	2
5	CONTA	A	Esquivel Mercy	1
5	CONTA	B	Salgado Rosario	1
5	CONTA	C	Baca Patricio	1
5	SECRE	S	Espinosa Fabián	1
6	SECRE	S	Espinosa Fabián	1

SubTotal = 10

Page 1 of 29

4.7 Acción: Presentar la carga académica por profesores

Servicio del sistema: Presentar carga académica por profesores.

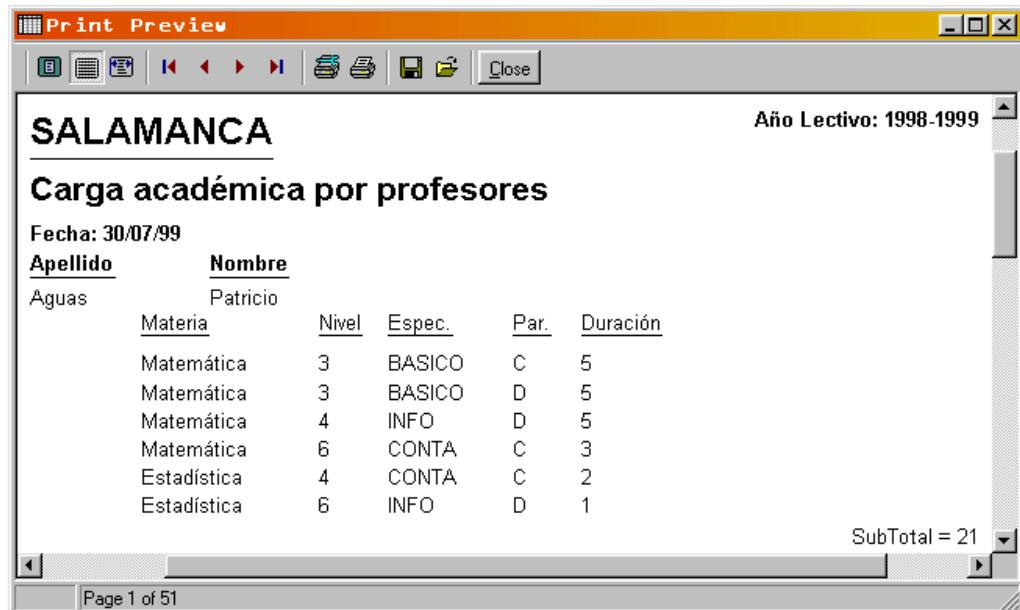
Servicio solicitado: Mostrar un reporte con la carga académica por profesores.

Método: TMainForm.actPresentarCargaAcademicaProfesorClick.

Acción: Presentar la carga académica por profesores.

Servicio en: Reporte.

Reporte



SALAMANCA Año Lectivo: 1998-1999

Carga académica por profesores

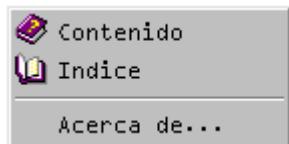
Fecha: 30/07/99

Apellido	Nombre	Materia	Nivel	Espec.	Par.	Duración
Aguas	Patricio	Matemática	3	BASICO	C	5
		Matemática	3	BASICO	D	5
		Matemática	4	INFO	D	5
		Matemática	6	CONTA	C	3
		Estadística	4	CONTA	C	2
		Estadística	6	INFO	D	1

SubTotal = 21

Page 1 of 51

5. Submenú Ayuda



Submenú Ayuda | Contenido

Muestra el contenido de la ayuda.

Submenú Ayuda | Índice

Muestra el índice de la ayuda.

Submenú Ayuda | Acerca de...

Muestra el acerca de... de la aplicación.

5.1 Acción: Contenido

Servicio del sistema: Mostrar el contenido de la ayuda.

Servicio solicitado: Mostrar el contenido de la ayuda.

Método: TMainForm.actContentsExecute.

Servicio en: Diálogo que muestra el contenido de la ayuda.

5.2 Acción: Índice

Servicio del sistema:  Mostrar el índice de la ayuda.

Servicio solicitado: Mostrar el índice de la ayuda.

Método: TMainForm.actIndexExecute.

Servicio en: Diálogo que muestra el índice de la ayuda.

5.3 Acción: Acerca de

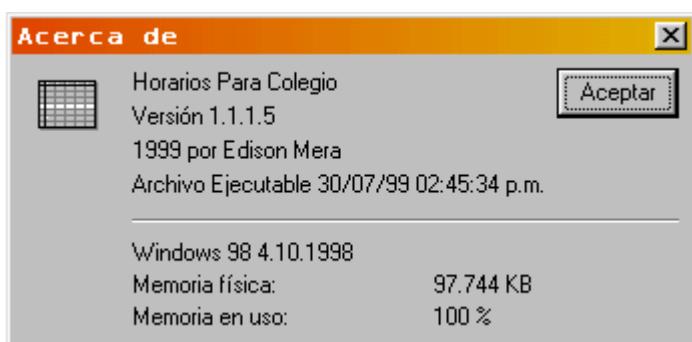
Servicio del sistema: Mostrar el acerca de... de la aplicación.

Servicio solicitado: Mostrar el acerca de... de la aplicación.

Método: TMainForm.actAboutClick.

Servicio en: Diálogo mostrando el acerca de... de la aplicación.

Diálogo mostrando el acerca de... de la aplicación



5 PRUEBAS REALIZADAS

El sistema fue empleado con éxito en el Colegio Nacional Mixto Nocturno Salamanca, obteniendo mejores resultados que cuando se elaboran los horarios manualmente. Las pruebas se basaron en los datos del Año Lectivo 1998 – 1999 ya que al momento de efectuarlas no estaba lista la carga académica del Año Lectivo 1999 – 2000. Sin embargo vale mencionar que el horario del presente año lectivo ha sido elaborado con ayuda de este sistema.

El presente colegio tiene 51 profesores, que se distribuyen en 27 paralelos, y en cada paralelo se deben cubrir entre 27 y 30 horas semanales, distribuidas en 6 horas diarias separadas por un recreo, entre la tercera y cuarta horas.

El rendimiento del algoritmo se probó en las siguientes configuraciones de computadores:

- 1) Computador Cyrix P166+ con 32 MB Ram, Win 95.
- 2) Computador Cyrix P200+ con 32 MB Ram, Win 95.
- 3) Computador Intel Celeron 333MHz con 64 MB Ram, Win 95.
- 4) Computador Intel Celeron 333MHz con 96 MB Ram, Win 98.

Estas pruebas permitieron poner en evidencia lo conveniente de llevar a cabo la elaboración del horario en el mejor computador disponible.

Para realizar el ajuste de los parámetros de configuración, se utilizó el computador 4.

Valor mínimo que toma la función objetivo en función del tiempo:

Configuración utilizada:

Semillas del GNA:

Semilla 1:	135987981
Semilla 2:	-4965498
Semilla 3:	165465
Semilla 4:	98798948

Pesos:

Cruce de profesores: 1447

Cruce de aulas:	1507
Horas huecas desubicadas:	805
Materias cortadas:	343
Materias no dispersas:	23
Prohibiciones de materia:	(99, 2000)
Prohibiciones de profesor:	(123, 2000)

Parámetros del algoritmo evolutivo:

Tamaño de la población:	5
Máximo de generaciones:	200
Probabilidad de cruzamiento:	0,30
Probabilidad de mutación 1:	0,25
Orden de la mutación 1:	3
Probabilidad de mutación 2:	0,20
Probabilidad de reparación:	0,15

Gráfico:

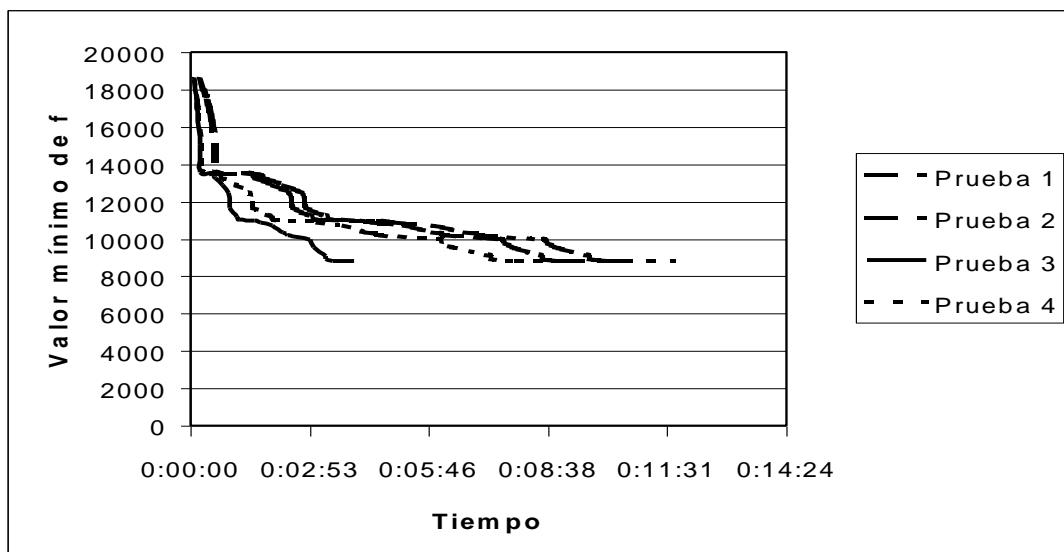


Gráfico 1: Valor mínimo de la función objetivo en función del tiempo, para 4 pruebas similares realizadas en diferentes máquinas.

A pesar de que la prueba 4 se llevó a cabo en el mejor computador, arrojó malos resultados, debido probablemente a que el computador tenía más programas precargados, por lo que se recomienda efectuar la elaboración del horario en un computador lo menos sobrecargado posible.

La prueba 3 resultó ser la mejor, debido a que se realizó en la segunda mejor máquina y no tiene muchos programas precargados.

Variación de la función objetivo y del tiempo de ejecución en función de los parámetros del algoritmo evolutivo:

Semillas del GNA:

Semilla 1:	135987981
Semilla 2:	-4965498
Semilla 3:	165465
Semilla 4:	98798948

Pesos:

Cruce de profesores:	1447
Cruce de aulas:	1507
Horas huecas desubicadas:	805
Materias cortadas:	343
Materias no dispersas:	23
Prohibiciones de materia:	(99, 2000)
Prohibiciones de profesor:	(123, 2000)

Tabla de los valores encontrados:

No.	Parámetros del A. Evolutivo							Mejores valores de la función objetivo Tiempo					Promedio	Desviación							
	1	2	3	4	5	6	7	1	2	3	4	5									
1	5	200	0,3	0,25	3	0,2	0,15	9866	00:03:57	9155	00:04:17	8804	00:03:59	10247	00:03:43	9298	00:04:02	9474	00:04:00	577,2	00:00:12
2	10	100	0,3	0,25	3	0,2	0,15	9880	00:06:11	10032	00:05:04	9674	00:05:35	8754	00:06:55	9834	00:05:33	9634,8	00:05:52	508,7	00:00:43
3	15	70	0,3	0,25	3	0,2	0,15	9487	00:06:33	9742	00:05:54	10181	00:05:43	10583	00:05:56	11036	00:05:33	10205,8	00:05:56	625,3	00:00:23
4	5	200	0,1	0,25	3	0,2	0,15	8055	00:03:07	8925	00:03:21	9718	00:03:15	8832	00:03:55	10585	00:03:20	9223	00:03:24	962,5	00:00:18
5	5	200	0,7	0,25	3	0,2	0,15	10371	00:06:24	8345	00:07:22	9261	00:06:46	8874	00:07:16	9431	00:08:32	9256,4	00:07:16	750,0	00:00:49
6	5	200	0,3	0,7	3	0,2	0,15	8980	00:07:02	9674	00:05:16	8234	00:04:37	9426	00:04:33	10725	00:04:35	9407,8	00:05:13	917,5	00:01:04
7	5	200	0,3	0,25	7	0,2	0,15	9305	00:04:17	9869	00:04:34	9619	00:04:35	8907	00:04:43	8956	00:03:21	9331,2	00:04:18	416,4	00:00:33
8	5	200	0,3	0,25	3	0,7	0,15	8488	00:06:58	9800	00:05:36	9161	00:06:45	8749	00:06:54	9720	00:07:02	9143,6	00:06:39	530,7	00:00:36
9	5	350	0,3	0,25	3	0,2	0,05	9483	00:02:28	10448	00:02:37	11078	00:02:11	9636	00:03:02	9928	00:02:18	10114,6	00:02:31	652,3	00:00:20
10	5	150	0,3	0,25	3	0,2	0,2	9726	00:04:31	9225	00:04:09	9542	00:03:35	10745	00:04:11	9443	00:04:21	9736,2	00:04:09	592,2	00:00:21

Como se puede observar, el mejor valor obtenido de la función objetivo es de 8055 en aproximadamente 3 minutos. Los valores de los parámetros deben ajustarse tomando en cuenta los mejores resultados, observando el tiempo que tarda el algoritmo y el valor de la función objetivo. En este caso, no resulta muy provechoso detenerse a realizar demasiadas pruebas de ajuste de los parámetros, puesto que en muy poco tiempo (de 3 a 10 minutos) se obtienen horarios de buena calidad.

6 CONCLUSIONES Y RECOMENDACIONES

Conclusiones

La complejidad del problema de la elaboración de un horario es extremadamente grande, esta depende de los siguientes factores:

- Número de paralelos.
- Número de horas laborables en una semana.

Para tener una idea del tamaño del espacio de búsqueda, se puede pensar en todas las combinaciones posibles de horario, que aproximadamente tiene un orden acotado por $O(P * L!)$ (P es el número de paralelos y L el de horas laborables), por ejemplo, en un colegio pequeño con 27 paralelos y 30 horas laborables el orden es de aproximadamente 10^{34} , un número del que ni siquiera es posible imaginar su magnitud.

Debido a esto, aunque en teoría debe alguna vez alcanzarse el óptimo global, en la práctica por falta de tiempo esto no sucede, entonces es conveniente realizar varias ejecuciones del algoritmo con distintos valores de los parámetros hasta encontrar una solución óptima.

Para ajustar los pesos de la función objetivo, se deben probar combinaciones que produzcan un mayor descenso de los cruces y prohibiciones. Debido a que gran parte de las restricciones fuertes fueron relajadas y pasadas a la función objetivo, para aquellas deben asignarse pesos más grandes. Por ejemplo, para el colegio considerado en las pruebas los siguientes pesos dieron resultados bastante buenos:

Cruce de profesores:	1447
Cruce de aulas:	1507
Horas huecas desubicadas:	805
Materias cortadas:	343
Materias no dispersas:	23
Prohibiciones de materia:	(No adecuado=99, Imposible=2000)
Prohibiciones de profesor:	(No gusta=123, No puede=2000)

Nótese el alto valor que se asigna a las prohibiciones más restrictivas, mientras que restricciones menos problemáticas como son las materias no dispersas, las prohibiciones de materia del tipo *no adecuado* y las prohibiciones de profesor del tipo *no gusta* tienen un peso mucho menor.

Indudablemente, el algoritmo evolutivo muestra una seria ventaja respecto de ciertas técnicas de búsqueda local, ya que el que se implementó para este problema utiliza el descenso rápido como parte de su funcionamiento, esto se puso en evidencia por las pruebas realizadas (ver gráfico 1), puesto que a los individuos iniciales se les aplicó un descenso rápido, y partiendo de un valor de la función objetivo de 18529 se llegó a un valor de 8804, lo que representa un descenso del 52%, respecto de utilizar únicamente descenso rápido.

En los problemas combinatorios es muy recomendable utilizar algoritmos evolutivos híbridos, por que si no la búsqueda no se acelera. Esto se puede comprobar en el sistema cuando la probabilidad de reparación es cero.

Recomendaciones

Con poco esfuerzo en programación, el sistema puede ser paralelizado. Sería bastante interesante que posteriormente se lleve a cabo esta experiencia. Sugerencia: En el algoritmo existe una función llamada *Elitista*, aquí habría que programar una rutina de exportación e importación de soluciones entre distintas máquinas, cada máquina exportaría la mejor solución al resto de máquinas, e importaría las mejores soluciones de las otras máquinas, y una vez recolectadas, las compararía y seguiría trabajando con la mejor de todas [24]p. 332-333.

Como detalle técnico, puesto que el sistema fue programado para Windows 95/98/NT, se puede trabajar utilizando un servidor NT, encapsular los objetos del algoritmo evolutivo en objetos DCOM (Distributed Component Object Model), configurar las otras máquinas con Windows 95/98 como servidores de objetos DCOM y la aplicación presente en el servidor coordinaría el trabajo del resto de máquinas.

Otro aspecto digno de mejorar sería el ocultar al usuario ciertos detalles técnicos como la fijación de los pesos en la función objetivo y el ajuste de los parámetros del algoritmo evolutivo. Existen técnicas *metaheurísticas* que permiten integrar en el algoritmo evolutivo el ajuste de los

parámetros y de los pesos, sin embargo, debido a que esto escapaba de los alcances de la presente tesis, no se lo realizó.

En cuanto a los parámetros del algoritmo evolutivo, el tamaño de la población y la probabilidad de reparación tienen una incidencia más directa sobre el rendimiento, por lo que deben ser ajustados con sumo cuidado. El valor sugerido para la probabilidad de reparación es de 0,15 [24]p. 321, sin embargo, parece ser que este parámetro es dependiente del problema, por lo que para cada caso debe estimarse el mejor valor a usar. Para el tamaño de la población, dado lo pesado del algoritmo, resulta adecuado tomar entre 5 y 10 individuos.

7 BIBLIOGRAFÍA

- [1] A. Hertz. “Tabu search for large scale timetabling problems”. European Journal of Operational Research 54 (1991) 39 – 47.
- [2] Alberto Colomni, Marco Dorigo and Vittorio Maniezzo. “A Genetic Algorithm to solve the Timetable Problem”. technical report no 90-060, Politecnico di Milano, Italy.
- [3] Alberto Colomni, Marco Dorigo and Vittorio Maniezzo. “Genetic Algorithms: A New Approach to the Time-Table Problem”. Politecnico di Milano, Dipartamento di Elettronica. 1992.
- [4] Alberto Colomni, Marco Dorigo and Vittorio Maniezzo. “Genetic Algorithms and Highly Constrained Problems: The Timetable Case”, proceedings of the 1st International Workshop on Parallel Problem Solving from Nature (Dortmund, Germany), Lecture Notes in Computer Science, vol 496, pp 55-59, Springer-Verlag, 1991.
- [5] Borland Delphi 4 <http://www.borland.com/delphi>
- [6] Borland Delphi 4. “Developer’s Guide”. Inprise Corporation, Scott Valley, CA. 1998.
- [7] C. Caux, H. Pierreval, M.C. PortMann. “Les algorithmes Génétiques et leur Application Aux Problèmes D’Ordonnancement”. Journées d’Etude sur l’Ordonnancement et entreprise, Toulouse 1994, 5 – 45.
- [8] D Abramson and J Abela. “A Parallel Genetic Algorithm for Solving the School Timetabling Problem”. 15 Australian Computer Science Conference, Hobart, February 1992.
- [9] Dainel Costa. “A Tabu Search algorithm for computing an operational timetable”. European Journal of Operational Research 76 (1994) 98 – 110.
- [10] Dan Haught, Jim Ferguson. “Microsoft Jet Database Engine, Programmer’s Guide”. Microsoft Press. Edición 1995 por Microsoft.
- [11] Data Architect <http://www.powersoft.com>
- [12] Edwar J. Anderson, Michael C. Ferris. “Genetic Algorithms for Combinatorial Optimization: The Assebly Line Balancing Problem”. ORSA Journal on Computing Vol. 6 No. 2, Spring 1994, Pág. 181-189.
- [13] Fred Glover. “Genetic Algorithms and Scatter Search: Unsuspected Potentials”. School of Busines, CB 419. University of Colorado Boulder, CO 80309, August 1993.
- [14] Jackes A. Ferland, Alain Hertz and Alain Lavoie. “An Object Oriented Methodology for solving assigment type problems with neightborhood search techniques”. Ecole Polytechnique Fédérale de Lausanne. March 1994.
- [15] James C. Bean. “Genetic Algorithms and Random Keys for Sequencing and Optimization”. ORSA Journal on Computing Vol. 6 No. 2, Spring 1994, Pág. 154-155.
- [16] James Martin / James J. Odell. “Análisis y diseño orientado a Objetos”.

- [17] Jean Aubin and Jacques A. Ferland. “A Large Scale Timetabling Problem”. Computers Opns Res. Vol. 16 No. 1. Pp 67 – 77, 1989.
- [18] Parra Oscar / Torres Luis. “Modelización y diseño de una heurística de solución para el problema de horarios en Universidades, y su implementación”. Tesis de Ingeniero Matemático. Escuela Politécnica Nacional. Quito, 1998.
- [19] Powersoft Process Analyst <http://www.powersoft.com>
- [20] Rupert Weare, Edmund Burke and Dave Elliman. “A Hybrid Genetic Algorithm for Highly Constrained Timetabling Problems”. Department of Computer Science, University of Nottingham. Nottingham, UK. Technical Report NOTTCS-TR-95-8, January 1995.
- [21] Salazar Rubio Luis. “Prototipo de un software para la construcción óptima de rutas de vehículos”. Tesis de Maestría en Informática. Escuela Politécnica Nacional. Quito, 1997.
- [22] Werner Junginger. “Timetabling in Germany – A Survey”. Interfaces 16: 4 July August 1986 (pp. 66 – 74).
- [23] Witold Salwach. “Genetic Algorithms in Solving Constraint Satisfaction Problems: The Timetable Case”. Institute of Industrial Engineering and Management, Technical University of Wroclaw. 1997.
- [24] Zbigniew Michalewicz. “Genetic Algorithms + Data Structures = Evolution Programs”. Springer Berlag 1996, Third Edition.