"Planificación automática de un proyecto a partir de la limitación de las capacidades de los recursos y la duración de las fases, considerando como prioridades las fechas de finalización exigidas y la máxima ocupación"

Autores:
Carlos Proensa Mora
Santiago Martín López
Pedro Rizaldos Pareja

Profesor director:
Miguel Ángel Blanco

Curso académico:
2006-2007

Proyecto de Sistemas Informáticos
Facultad de Informática
Universidad Complutense de Madrid

Índice

1. Resumen

El objetivo de este proyecto consiste en desarrollar una herramienta capaz de planificar automáticamente un proyecto dado siguiendo para ello varias formas de planificación.
La herramienta se compone de una base de datos que almacena toda la información necesaria de un proyecto, como son las tareas y los recursos propios, y unos algoritmos de planificación que utilizan esa base de datos para ofrecer resultados adecuados.
Para la creación de la base de datos, en primer lugar, definimos un modelo entidad-relación que posteriormente se convertirá en el modelo relacional. Sobre ella se sustentan los diferentes métodos de planificación: hacia delante, hacia atrás y por costes.
A partir de las diferentes formas de planificación de un mismo proyecto se puede observar cual de ellas es la más adecuada para el cliente que solicite los servicios de esta herramienta.

Summary

The objective of this project consists of developing a tool able to plan automatically a given project following for it several forms of planning.
The tool is made up of a data base that stores all the necessary information of a project, like they are the own tasks and resources, and scheduling algorithms that use that data base to offer suitable results.
For the creation of the data base, in the first place, we defined a organization-relation model that later will become the relational model. On it the different methods from planning are sustained: towards ahead, backwards and by costs.
To start off of the different forms of planning from a same project can be observed as of them it is adapted for the client who asks for the services of this tool.

2. Palabras Clave

- Base de datos
- Gestión de proyectos
- Planificación automática
- Planificador
- Gestión
- Proyecto
- Recursos
- Tareas
- SQL
- Dependencias

3. Actas del proceso de diseño.

Los diferentes aspectos de diseño que se trataron en cada reunión se recogieron en este formato de acta.

Acta 31 Octubre 2006. Proyecto

Fecha

31 de Octubre de 2006

Asistentes

- Pedro Rizaldos
- Santiago Martín
- Carlos Proensa
- Miguel Ángel Blanco (profesor)

Lugar

Despacho de Miguel Ángel Blanco en la Facultad de Informática

---

Temas Tratados

1. Actas de reuniones

2. Especificación de elementos del dominio

3. Consulta sobre la denominación oficial del proyecto

Temas Acordados

1. Actas de reuniones

Las actas deberán incluir la siguiente información:

- fecha
- asistentes
- temas tratados
- temas acordados

Cada reunión quedará reflejada en un acta, que deberá ser aprobada por el profesor supervisor del proyecto.

2. Especificación de elementos del dominio

En el desarrollo de la aplicación se tendrán en cuenta los siguientes conceptos

Proyectos

- Objetivo es gestionar recursos.
- Planificación se compone de fases (se profundizará más adelante)
- Un proyecto para un cliente
- Un proyecto para un cliente puede tener varios presupuestos con distintas planificaciones (versiones)
- Se definirán estados para el proyecto: ofertado, aceptado, en curso, terminado
- Implementar un histórico de proyectos para: estadísticas, reutilizar partes o estimar presupuestos.
- Un proyecto tiene un diario de recursos, con información detallada de consumos de cada recurso
- Estadísticas del proyecto: incluye un resumen de recursos usados con información de

Recursos

- Información que define un recurso:
    - tipo: si es persona, material, etc
    - categoría (clasificación):
    - capacidad: disponibilidad de cada recurso
    - coste

  - unidad en que se mide: unidades materiales, de tiempo, etc
  - datos personales, localización, descripción, etc

- Clasificar familias de recursos:
  - Poder usar las familias como un recurso genérico (para planificación en vacío)

- Tener un calendario de recursos, con información detallada de capacidad.

3. Consulta sobre la denominación oficial del proyecto

El profesor desconocía el código y la descripción con la que se ha registrado el proyecto, por lo que nos lo comunicará en la siguiente reunión.

Observaciones

La especificación de conceptos en el punto 2 consiste en un resumen de los elementos que formarán parte del dominio de la aplicación. En las sucesivas sesiones, con una visión más global del sistema, se irá ampliando el detalle de la especificación.

Acta 14 Noviembre 2006. Proyecto

Fecha

14 de Noviembre de 2006

Asistentes

- Miguel Ángel Blanco (profesor)
- Santiago Martín
- Pedro Rizaldos
- Carlos Proensa

Lugar

Despacho de Miguel Ángel Blanco en la Facultad de Informática

---

Temas Tratados

1. Necesidad de una buena especificación de funcionalidades del proyecto

2. Demostración por parte del profesor de diversos requisitos funcionales que se pueden incluir en el proyecto

---

Temas Acordados

1. Necesidad de una buena especificación de funcionalidades del proyecto

¿Cuál es el objetivo de nuestro proyecto?

A partir de un proyecto deseamos generar automáticamente la planificación detallando: fechas de las fases y distribución de recursos. La planificación puede realizarse con distintos criterios: planificación hacia delante y hacia atrás o con restricciones en el coste final.

¿Qué datos son necesarios para calcular esa planificación?

Un proyecto contiene cierta información que lo define. Hay que conocer cuáles son las fases del proyecto, de qué tareas consiste cada fase, y que recursos necesita cada tarea. Estos datos son específicos de cada proyecto y ya estarán definidos previamente.

¿Qué necesitamos saber de los recursos para poder hacer la planificación?

Hay que clasificarlos en familias de recursos según características comunes entre ellos y conocer sus capacidades y coste asociado. Una familia de recursos tendrá una cierta habilidad para desarrollar una tarea, mientras que cada miembro de esa familia tendrá su propia dedicación al trabajo.

¿Como se produce la relación entre el proyecto y el cliente?

En primer lugar un cliente hace una petición del trabajo que quiere. El cliente puede especificar ciertas restricciones como fechas límite o de costes. A partir de esa información se elabora un proyecto y una planificación provisional que tendrá un coste que se utilizará para componer un presupuesto. Para una misma petición se pueden hacer diferentes proyectos, con su correspondiente planificación, que producen varios presupuestos. Estos proyectos provisionales se encuentran en estado “ofertado”.

Cuando el cliente ha aceptado alguno de los presupuestos, el proyecto elegido pasará a estado “aceptado” y se realizará una planificación real.

Características de los recursos:

Necesitamos conocer todos los recursos en su totalidad, incluyendo tanto los que están asignados a proyectos como los que están disponibles. Usaremos un concepto denominado *almacén de recursos*. Aquí se encuentran clasificados con toda la información que los define:

- Familia(s) a la que pertenece
- Tipo de recurso: si es persona, material, maquina...
- Categoría: que permite diferenciar recursos dentro de una misma familia
- Capacidad: define la disponibilidad del recurso para ser utilizado, especificando días y horas.
- Coste: coste asociado al uso del recurso, dado en euros, por una unidad de tiempo. Existe la posibilidad de definir costes adicionales para un recurso en casos concretos como horas extras de un trabajador, sobre uso de una maquina...
- Unidades de medida: pueden ser distintas para recursos tipo persona, maquina, etc., y pueden ser horas, días, semanas.
- Datos descriptivos del recurso:
    - para personas serían: nombre, dirección, teléfono, etc.
    - para maquinas y materiales serian: nombre, descripción, características, localización, etc.

Recursos genéricos

Un recurso genérico es un recurso que pertenece a una familia pero no contiene información de ningún recurso concreto. Estos recursos se usarán en las planificaciones no definitivas de un proyecto, como puede ser la que se realiza para ofertar un presupuesto.

En principio no tiene limitaciones de disponibilidad, aunque en la planificación se pueda tener en cuenta la disponibilidad de los recursos reales en particular para ajustar mejor el coste. El coste de un recurso genérico será similar al de los recursos

reales de su misma familia, por la misma razón que antes. El resto de atributos no difiere de los de un recurso normal.

Características de proyecto

Un proyecto se define con la siguiente información:

- una identificación
- un nombre
- un responsable
- un estado (ofertado, aceptado, en curso, finalizado...)
- una descripción, de manera que sus características lo identifiquen perfectamente.

El proyecto debe especificar:

- todas las tareas que lo componen
- la agrupación de las tareas en fases
- los recursos que requieren cada tarea
- dependencias temporales entre tareas
- restricciones de tareas, como fechas de inicio o finalización.

Características de tarea

Una tarea se define con la siguiente información:

- Nombre
- Descripción
- Duración estimada
- Necesidad de recursos:
    - la tarea puede requerir recursos de diferentes familias.
    - adicionalmente, de cada familia puede especificar un mínimo y un máximo de recursos que se le puede asignar
- Dependencias: con otras tareas
    - no comenzar hasta que otra finalice
- Restricciones

- de tiempo: limites en fechas de inicio o final, o de duración
- de coste: si existe alguna condición para el coste que genera esta tarea

¿Cómo se asignan los recursos a las tareas del proyecto?

La asignación se realiza siguiendo diferentes estrategias, como son “minimización de costes” y “minimización de tiempo”, pudiendo además dividir ésta última en otras dos subestratégicas, que serian “fecha inicio” y “fecha limite”.

Durante la planificación, los recursos se asignarán a las tareas de forma que cumplan las restricciones que nos dice la tarea. En caso de no haber recursos disponibles las fechas de comienzo y final de la tarea se modificaran dependiendo de los recursos que se han podido asignar.

Por ejemplo, una tarea que necesita 2 personas y no hay recursos disponibles, la fecha de inicio de la tarea se retrasará hasta que haya 2 personas disponibles. Por otro lado, si una tarea especifica que dura 2 días usando 2 personas, si solo tenemos una disponible y la tarea lo permite, podemos asignarla, pero la duración de la tarea se ampliará a 4 días.

¿Cómo se realiza un buen seguimiento del proyecto?

Tendremos a nuestra disposición herramientas tales como las estadísticas, que nos permitirán conocer tanto el tiempo empleado en las fases, subfases, tareas, etc.; como el coste, ya sea individual de cada recurso, o planificado por tareas o fases, y, por supuesto, el coste total del proyecto. Otra buena herramienta de seguimiento será el histórico de proyectos, en el que iremos guardando información relevante sobre lo que se está haciendo, lo que nos ha costado hacerlo y el tiempo que se ha tardado.

Estadísticas de un proyecto

Con las estadísticas el usuario podrá obtener una visión global del proyecto con información sobre tiempos:

- Tareas que se han realizado hasta el momento
- Tareas que faltan
- Diferencias sobre la planificación inicial

También información sobre costes, en total y por recursos:

- Importe de venta: A lo que vamos a venderlo
- Importe neto o de coste: Lo que nos va a costar adquirirlo
- Beneficio bruto, que será la diferencia entre los anteriores.

Histórico de proyectos

Ofrecerá al usuario una visión global de varios proyectos ya finalizados. Se mostrará la siguiente información:

- Tareas realizadas
- Coste de cada una de las tareas realizadas
- Tiempo que ha llevado realizar cada tarea

Se podrá realizar consultas de un proyecto en curso hasta su parte ya completada, para dar una visión de ejecución del proyecto. Estos datos estarán relacionados con las estadísticas de la aplicación, ya que se podrán visualizar todos los datos para que el usuario pueda comparar la ejecución de proyectos parecidos, basándose en estadísticas incluso de un conjunto de proyectos.

Acta 19 Diciembre 2006. Proyecto

Fecha

19 de Diciembre de 2006

Asistentes

- Miguel Ángel Blanco (profesor)
- Santiago Martín
- Pedro Rizaldos
- Carlos Proensa

Lugar

Despacho de Miguel Ángel Blanco en la Facultad de Informática

---

Temas Tratados

1. Revisión de los requisitos funcionales

2. Explicación para comenzar la realización del modelo entidad relación

---

Temas Acordados

1. Revisión de los requisitos funcionales

Después de la discusión sobre la especificación de los requisitos funcionales de la pasada reunión, hemos decidido realizar modificaciones en algunos de los temas tratados:

- **Recursos** (*Capacidad*): Distinguiremos la capacidad del recurso entre disponible, asignada y actual (actual= disponible-asignada)
- **Recursos** (*Tipos de recursos*): Distinguiremos sólo 2 tipos de recursos: Personas y productos. Añadiremos familias de recurso y familias de productos.
- **Recursos** (*Unidades de asignación*): Las unidades de asignación tienen un mínimo de horas
- **Recursos genéricos**: La disponibilidad se debe calcular en el momento
- **Tareas y Recursos**: Una tarea necesita un número concreto de recursos, ni más ni menos
- **Tareas**: Ya están definidas, en horas
- **Tareas** (*Dependencias dentro de tareas*): No se contempla "encadenamiento en tareas" es decir, dentro de una tarea NO hay dependencias entre los recursos
- **Tareas** (*Dependencias entre tareas*): Existen dependencias entre tareas, pero sólo se tiene en cuenta cuándo una tarea se ha completado totalmente.
- **Planificaciones**: Al hacer planificación en vacío distinguiremos entre una posible planificación con disponibilidad total o comprometida, para tener en cuenta los recursos (genéricos) que se están empleando en proyectos ofertados

2. Explicación para comenzar la realización del modelo entidad relación

Teniendo en cuenta lo anteriormente acordado, describiremos el modelo entidad relación de la aplicación.

Entidades

PROYECTO

Atributos

- id proyecto

Tipo: ENTERO, autonumérico, >0
Es clave primaria.
No tiene significado propio, sólo identificador numérico.

- nombre

Tipo: TEXTO corto
Es un nombre corto del proyecto

- descripción

Tipo: TEXTO largo
Es una descripción concisa del proyecto

Relaciones

Relaciones que afectan a esta entidad:

- PROYECTO *tiene_responsable* RECURSO
- PROYECTO *en_estado* DESCR_ESTADO
- PROYECTO *formado_por* TAREA
- PROYECTO *asignado_presupuesto* PRESUPUESTO
- PROYECTO *de_expediente* EXPEDIENTE

DESCR_ESTADO

Atributos

- id_estado

Es clave primaria
Tipo: ENTERO, autonumérico, >0

- nombre

Tipo: TEXTO corto

Relaciones

Relaciones que afectan a esta entidad:

- PROYECTO *en_estado* DESCR_ESTADO

TAREA

Atributos

- id_tarea

Es clave primaria
Tipo: ENTERO, autonumérico, >0

- nombre

Tipo: TEXTO corto. Es un nombre corto de la tarea

- descripción

Tipo: TEXTO largo. Es una descripción concisa para la tarea

- fecha_inicio

Tipo: FECHA. Es la fecha de inicio calculada en la planificación

- fecha_fin

Tipo: FECHA. Es la fecha de finalización calculada en la planificación

- restr_fecha_inicio

Tipo: FECHA, puede ser NULL
Si existe, es una restricción para fecha de inicio de la tarea, impuesta de antemano en la creación del proyecto

- restr_fecha_fin

Tipo: FECHA, puede ser NULL
Si existe, es una restricción para fecha de finalización de la tarea, impuesta de antemano en la creación del proyecto

- restr_coste

Tipo: DECIMAL, puede ser NULL
Si existe, es una restricción para el coste máximo que puede acumular esta tarea.

Relaciones

Relaciones que afectan a esta entidad:

- PROYECTO *formado_por* TAREA
- TAREA *necesita* RECURSO
- TAREA *depende_de* TAREA

RECURSO

Descripción

Hereda de *Tipo_recurso*

Atributos

- nombre

Tipo: TEXTO corto. Nombre corto para el recurso

Relaciones

Relaciones que afectan a esta entidad:

- RECURSO *de_familia* FAMILIA_RECURSO
- RECURSO *es_un* TIPO_RECURSO
- RECURSO *disponible* CALENDARIO

FAMILIA_RECURSO

Descripción

Hereda de *Tipo_recurso*. Identifica los posibles recursos que tenemos (Ej.: Consultores, programadores)

Atributos

- descripción

Tipo: TEXTO corto

Relaciones

Relaciones que afectan a esta entidad:

- FAMILIA_RECURSO *es_un* TIPO_RECURSO
- RECURSO *de_familia* FAMILIA_RECURSO

PRODUCTO

Descripción

Hereda de *Tipo_recurso.* Entidad que almacena productos (almacenaremos las herramientas que utilicemos para el desarrollo).

Atributos

- Descripción

Relaciones

Relaciones que afectan a esta entidad:

- PRODUCTO *de_familia* FAMILIA_PRODUCTO
- PRODUCTO *es_un* TIPO_RECURSO
- PRODUCTO *disponible* CALENDARIO

FAMILIA_PRODUCTO

Descripción

Hereda de *Tipo_recurso* Identifica los posibles productos que podemos utilizar (Ej: Oracle tiene varios productos)

Atributos

- descripción

Tipo: TEXTO corto

Relaciones

Relaciones que afectan a esta entidad:

- FAMILIA_PRODUCTO *es_un* TIPO_RECURSO
- PRODUCTO *de_familia* FAMILIA_PRODUCTO

TIPO_RECURSO

Atributos

- id_tipo

Es clave primaria
Tipo: ENTERO, autonumérico, >0

- descripción

Tipo: TEXTO corto

Relaciones

Relaciones que afectan a esta entidad:

- TAREA *necesita* TIPO_RECURSO

CALENDARIO

Descripción

Es entidad débil

Atributos

- año
- dia001
- dia002
- dia003
- ...
- dia366

Cada atributo diaxxx contiene el numero de horas disponibles

Relaciones

Relaciones que afectan a esta entidad:

- RECURSO *disponible* CALENDARIO

EXPEDIENTE

Atributos

- num_exp

Es clave primaria
Tipo: ENTERO, autonumérico, >0

Relaciones

Relaciones que afectan a esta entidad:

- EXPEDIENTE *de_cliente* CLIENTE
- PROYECTO *de_expediente* EXPEDIENTE

CLIENTE

Atributos

- id_cliente

Es clave primaria
Tipo: ENTERO, autonumérico, >0

- CIF

Tipo: TEXTO corto
Número de identificación fiscal de la empresa o DNI de particular

- teléfono

Tipo: TEXTO corto

- dirección

Tipo: TEXTO corto

- ciudad

Tipo: TEXTO corto

- país

Tipo: TEXTO corto

- email

Tipo: TEXTO corto

Relaciones

Relaciones que afectan a esta entidad:

- EXPEDIENTE *de_cliente* CLIENTE

HISTORICO

Descripción

Entidad que va a almacenar la dedicación de cada recurso en un proyecto_determinado.

Cuando un proyecto acabe se realiza el volcado de los datos en esta tabla.

Atributos

- id_recurso

Es clave primaria

- dia

Tipo: FECHA

- Proyecto

Tipo: entero (identificador del proyecto)

- Horas

Tipo: decimal

PRESUPUESTO

Atributos

- id_presupuesto

Es clave primaria
Tipo: ENTERO, autonumérico, >0

- fecha_inicio

Tipo: FECHA

- fecha_fin

Tipo: FECHA

- coste

Tipo: Numerico

- Precio_venta

Tipo: Numerico

Relaciones

Relaciones que afectan a esta entidad:

- PROYECTO *asignado_presupuesto* PRESUPUESTO

RELACIONES

PROYECTO <tiene_responsable> RECURSO

PROYECTO * *tiene_responsable* 1 RECURSO

Cardinalidad: N a 1

PROYECTO <en_estado> DESCR_ESTADO

PROYECTO * *en_estado* 1 DESCR_ESTADO

Cardinalidad: N a 1

PROYECTO <formado_por> TAREA

PROYECTO 1 *formado_por* * TAREA

Cardinalidad: 1 a N

Atributos:

- fase:

Tipo: ENTERO, >0

PROYECTO <asignado_presupuesto> PRESUPUESTO

PROYECTO 1 *asignado_presupuesto* * PRESUPUESTO
Cardinalidad: 1 a N

EXPEDIENTE <de_cliente> CLIENTE

EXPEDIENTE * *de_cliente* 1 CLIENTE
Cardinalidad: N a 1

PROYECTO <de_expediente> EXPEDIENTE

PROYECTO * *de_expediente* 1 EXPEDIENTE
Cardinalidad: N a 1

TAREA <necesita> TIPO_RECURSO

TAREA 1 *necesita* * TIPO_RECURSO

Cardinalidad: 1 a N

Atributos

- Cantidad

Tipo: ENTERO, >0

TAREA <depende_de> TAREA

TAREA 1 *depende_de* * TAREA

Cardinalidad: 1 a N

RECURSO <de_familia> FAMILIA_RECURSO

RECURSO * *de_familia* 1 FAMILIA_RECURSO

Cardinalidad: N a 1

PRODUCTO <de_familia> FAMILIA_PRODUCTO

PRODUCTO * *de_familia* 1 FAMILIA_PRODUCTO

Cardinalidad: N a 1

RECURSO <disponible> CALENDARIO

RECURSO 1 *disponible* 1 CALENDARIO

Cardinalidad: 1 a 1

PRODUCTO <disponible> CALENDARIO

PRODUCTO 1 *disponible* 1 CALENDARIO

Cardinalidad: 1 a 1

4. Especificaciones funcionales.

¿Cuál es el objetivo de nuestro proyecto?
A partir de un proyecto deseamos generar automáticamente la planificación detallando: fechas de las fases y distribución de recursos. La planificación puede realizarse con distintos criterios: planificación hacia delante y hacia atrás o con restricciones en el coste final.

¿Qué datos son necesarios para calcular esa planificación?
Un proyecto contiene cierta información que lo define. Hay que conocer cuáles son las fases del proyecto, de qué tareas consiste cada fase, y que recursos necesita cada tarea. Estos datos son específicos de cada proyecto y ya estarán definidos previamente.

¿Qué necesitamos saber de los recursos para poder hacer la planificación?
Hay que clasificarlos en familias de recursos y familias de productos según características comunes entre ellos y conocer sus capacidades y coste asociado. Una familia de recursos tendrá una cierta habilidad para desarrollar una tarea, mientras que cada miembro de esa familia tendrá su propia dedicación al trabajo. Lo mismo ocurre con una familia de productos, cada miembro de esa familia tendrá asignada una capacidad en concreto.

¿Como se produce la relación entre proyecto y el cliente?
En primer lugar un cliente hace una petición del trabajo que quiere. El cliente puede especificar ciertas restricciones como fechas límite de entrega o de costes. A partir de esa información se elabora un proyecto y una planificación provisional que tendrá un coste que se utilizará para componer un presupuesto. Para una misma petición se pueden hacer diferentes proyectos, con su correspondiente planificación, que producen varios presupuestos. Estos proyectos provisionales se encuentran en estado “ofertado”.

Cuando el cliente ha aceptado alguno de los presupuestos, el proyecto elegido pasará a estado “aceptado” y se realizará una planificación real.

Características de los recursos

Necesitamos conocer todos los recursos en su totalidad, incluyendo tanto los que están asignados a proyectos con los que están disponibles. Usaremos un concepto denominado *almacén de recursos*. Aquí se encuentran clasificados con toda la información que los define:

- Familia a la que pertenece (Familia de recursos o Familia de productos)
- Tipo de recurso: si es persona, se incluirá dentro de la familia de recursos. Serán los denominados tipo Recurso. Si es material, máquinas...se relacionará con la familia de productos. Serán los tipo Producto.
- Categoría: que permite diferenciar recursos dentro de una misma familia
- Capacidad: define la disponibilidad del recurso para ser utilizado, especificando días y horas.
- Coste: coste asociado al uso del recurso, dado en euros, por una unidad de tiempo. Existe la posibilidad definir costes adicionales para un recurso en casos concretos como horas extras de un trabajador, mayor utilización de una maquina...
- unidades de medida: Vendrán definidas por el tiempo que se utilice un recurso, que será en horas.

Recursos genéricos

Un recurso genérico es un recurso que pertenece a una familia pero no contiene información de ningún recurso concreto. Estos recursos se usarán en las planificaciones no definitivas de un proyecto, como puede ser la que se realiza para ofertar un presupuesto.

En principio no tiene limitaciones de disponibilidad, aunque en la planificación se pueda tener en cuenta la disponibilidad de los recursos reales.

El coste de un recurso genérico será similar al de los recursos reales de su misma familia.
El resto de atributos no difiere de los de un recurso normal.

Características de proyecto
Un proyecto se define con la siguiente información:

- identificación
- un nombre
- un responsable
- un estado (ofertado, aceptado, en curso, finalizado...)
- una descripción, de manera que sus características lo identifiquen perfectamente.

El proyecto debe especificar:

- todas las tareas que lo componen
- la agrupación de las tareas en fases
- dependencias temporales entre tareas
- los recursos que requieren cada tarea
- restricciones de tareas como fechas de inicio o finalización.

Características de tarea
Una tarea se define con la siguiente información:

- Nombre
- Descripción
- Duración estimada
- Necesidad de recursos:
  - la tarea puede requerir recursos de diferentes familias.
- Dependencias: con otras tareas
  - no comenzar hasta que otra finalice
- Restricciones
  - de tiempo: limites en fechas de inicio o final, o de duración

¿Cómo se asignan los recursos a las tareas del proyecto?
La asignación se realiza siguiendo diferentes estrategias, como son “minimización de costes” y “minimización de tiempo”,

pudiendo además dividir ésta última en otras dos subestratégias, que serian “fecha inicio” y “fecha limite”.
Durante la planificación, los recursos se asignarán a las tareas de forma que cumplan las restricciones que nos dice la tarea. En caso de no haber recursos disponibles las fechas de comienzo y final de la tarea se modificaran dependiendo de los recursos que se han podido asignar.
Por ejemplo, una tarea que necesita 2 personas y no hay recursos disponibles, la fecha de inicio de la tarea se retrasará hasta que haya 2 personas disponibles. Por otro lado, si una tarea que especifica que dura 2 días usando 2 personas, si solo tenemos una disponible y la tarea lo permite, podemos asignarla, pero la duración de la tarea se ampliará a 4 días.

¿Cómo se realiza un buen seguimiento del proyecto?
Tendremos a nuestra disposición herramientas tales como las estadísticas, que nos permitirán conocer tanto el tiempo empleado en las fases, subfases, tareas, etc.; como el coste, ya sea individual de cada recurso, o planificado por tareas o fases, y, por supuesto, el coste total del proyecto. Otra buena herramienta de seguimiento será el histórico de proyectos, en el que iremos guardando información relevante sobre lo que se está haciendo, lo que nos ha costado hacerlo y el tiempo que se ha tardado.

Estadísticas de un proyecto
Con las estadísticas el usuario podrá obtener una visión global del proyecto con información sobre tiempos:

- Tareas que se han realizado hasta el momento
- Tareas que faltan
- Diferencias sobre la planificación inicial

También información sobre costes, en total y por recursos:

- Importe de venta: A lo que vamos a venderlo
- Importe neto o de coste: Lo que nos va a costar adquirirlo
- Beneficio bruto, que será la diferencia entre los anteriores.

Histórico de proyectos
Ofrecerá al usuario una visión global de varios proyectos ya finalizados. Se mostrará la siguiente información:

- Tareas realizadas.
- Coste de cada una de las tareas realizadas
- Tiempo que ha llevado realizar cada tarea

Se podrá realizar consultas de un proyecto en curso hasta su parte ya completada, para dar una visión de ejecución del proyecto. Estos datos estarán relacionados con las estadísticas de la aplicación, ya que se podrán visualizar todos los datos para que el usuario pueda comparar la ejecución de proyectos parecidos, basándose en estadísticas incluso de un conjunto de proyectos.

5. Modelo Entidad-Relación.

Entidades

PROYECTO

Descripción

Es la entidad principal de la base de datos.

Atributos

- id_proyecto

Tipo: ENTERO, autonumérico, >0
Es clave primaria
No tiene significado propio, sólo identificador numérico

- nombre

Tipo: TEXTO corto
Es un nombre corto del proyecto

- descripción

Tipo: TEXTO largo
Es una descripción concisa del proyecto

- fecha_inicio

Tipo: FECHA

- fecha_fin

Tipo: FECHA

Relaciones

Relaciones que afectan a esta entidad:

- PROYECTO *tiene_responsable* RECURSO

- PROYECTO *en_estado* DESCR_ESTADO
- PROYECTO *dividido_en* FASE
- PROYECTO *asignado_presupuesto* PRESUPUESTO
- PROYECTO *de_expediente* EXPEDIENTE

DESCR_ESTADO

Descripción

Almacena el estado en el que se encuentra el proyecto: ofertado, presupuestado, aceptado, concluido, etc.

Atributos

- id_estado

Es clave primaria
Tipo: ENTERO, autonumérico, >0

- nombre

Tipo: TEXTO corto

Relaciones

Relaciones que afectan a esta entidad:

- PROYECTO *en_estado* DESCR_ESTADO

FASE

Descripción

Cada proyecto se subdivide en diferentes fases que se planifican por separado, para después enlazar las fases en función de las dependencias.

Atributos

* id_fase
Es clave primaria
Tipo: ENTERO, autonumérico, >0

- nombre

Tipo: TEXTO corto

Relaciones

Relaciones que afectan a esta entidad:

- PROYECTO *dividido en* FASE
- FASE *formado por* TAREA

TAREA

Descripción

Cada fase se compone de diferentes tareas que se tratan de forma individual y se entrelazan, dentro siempre de una misma fase, en función de sus dependencias.

Atributos

- id_tarea

Es clave primaria
Tipo: ENTERO, autonumérico, >0

- nombre

Tipo: TEXTO corto. Es un nombre corto de la tarea

- descripción

Tipo: TEXTO largo. Es una descripción concisa para la tarea

- fecha_inicio

Tipo: FECHA. Es la fecha de inicio calculada en la planificación

- fecha_fin

Tipo: FECHA. Es la fecha de finalización calculada en la planificación.

- restr_fecha_inicio

Tipo: FECHA, puede ser NULL
Si existe, es una restricción para fecha de inicio de la tarea, impuesta de antemano en la creación del proyecto

- restr_fecha_fin

Tipo: FECHA, puede ser NULL
Si existe, es una restricción para fecha de finalización de la tarea, impuesta de antemano en la creación del proyecto

Relaciones

Relaciones que afectan a esta entidad:

- FASE *formada por* TAREA
- TAREA *asignada* TIPO_RECURSO
- TAREA *depende_de* TAREA

RECURSO

Descripción

Hereda de *Tipo_recurso*

Atributos

- nombre

Tipo: TEXTO corto. Nombre corto para el recurso

- estado

Tipo: TEXTO corto. Estado del recurso

- coste

Tipo: DECIMAL. Coste de utilización del recurso

Relaciones

Relaciones que afectan a esta entidad:

- RECURSO *de_familia* FAMILIA_RECURSO
- RECURSO *es_un* TIPO_RECURSO
- RECURSO *disponible* CALENDARIO

FAMILIA_RECURSO

Descripción

Hereda de *Tipo_recurso.* Identifica los posibles recursos que tenemos (Ej: Consultores, programadores)

Atributos

- descripción

Tipo: TEXTO corto

Relaciones

Relaciones que afectan a esta entidad:

- FAMILIA_RECURSO *es_un* TIPO_RECURSO
- RECURSO *de_familia* FAMILIA_RECURSO

PRODUCTO

Descripción

Hereda de *Tipo_recurso.* Entidad que almacena productos (almacenaremos las herramientas que utilicemos para el desarrollo).

Atributos

- Nombre

Tipo: TEXTO corto

- estado

Tipo: TEXTO corto. Estado del Producto

- coste

Tipo: DECIMAL. Coste de utilizacion del producto

Relaciones

Relaciones que afectan a esta entidad:

- PRODUCTO *de_familia* FAMILIA_PRODUCTO
- PRODUCTO *es_un* TIPO_RECURSO
- PRODUCTO *disponible* CALENDARIO

FAMILIA_PRODUCTO

Descripción

Hereda de *Tipo_recurso* Identifica los posibles productos que podemos utilizar (Ej: Oracle tiene varios productos)

Atributos

- descripción

Tipo: TEXTO corto.

Relaciones

Relaciones que afectan a esta entidad:

- FAMILIA_PRODUCTO *es_un* TIPO_RECURSO
- PRODUCTO *de_familia* FAMILIA_PRODUCTO

TIPO_RECURSO

Descripción

Las entidades Recurso, Familia_Recurso, Producto, Familia_Producto heredan de esta entidad. Es una entidad genérica para los recursos y los productos que forman parte de un mismo tipo.

Atributos

- id_tipo

Es clave primaria
Tipo: ENTERO, autonumérico, >0

- descripción

Tipo: TEXTO corto

- coste

Tipo: DECIMAL

Relaciones

Relaciones que afectan a esta entidad:

- TAREA *necesita* TIPO_RECURSO

CALENDARIO

Descripción

Nos da la información de disponibilidad en un año de un recurso.

Atributos

- año

- dia001
- dia002
- dia003
- ...
- dia366

Cada atributo diaxxx contiene el numero de horas disponibles

Relaciones

Relaciones que afectan a esta entidad:

- RECURSO *disponible* CALENDARIO

EXPEDIENTE

Descripción

Es la forma de relacionar un cliente con todos sus proyectos, independientemente del estado en el que se encuentren.

Atributos

- num_exp

Es clave primaria
Tipo: ENTERO, autonumérico, >0

Relaciones

Relaciones que afectan a esta entidad:

- EXPEDIENTE *de_cliente* CLIENTE
- PROYECTO *de_expediente* EXPEDIENTE

CLIENTE

Descripción

Identifica unívocamente a un cliente, con todos sus datos.

Atributos

- id_cliente

Es clave primaria
Tipo: ENTERO, autonumérico, >0

- CIF

Tipo: TEXTO corto
Número de identificación fiscal de la empresa o DNI de particular

- teléfono

Tipo: TEXTO corto

- dirección

Tipo: TEXTO corto

- ciudad

Tipo: TEXTO corto

- país

Tipo: TEXTO corto

- email

Tipo: TEXTO corto

Relaciones

Relaciones que afectan a esta entidad:

- EXPEDIENTE *de_cliente* CLIENTE

HISTORICO

Descripción

Entidad que va a almacenar la dedicación de cada recurso en un proyecto_determinado.
Cuando un proyecto acabe se realiza el volcado de los datos en esta tabla.

Atributos

- id_recurso

Es clave primaria

- Dia

Tipo: FECHA

- Proyecto

Tipo: entero (identificador del proyecto)

- Horas

Tipo: decimal

Relaciones

Relaciones que afectan a esta entidad:

PRESUPUESTO

Descripción

Almacena información relevante sobre un presupuesto relacionado con un proyecto determinado.

Atributos

- id_presupuesto

Es clave primaria
Tipo: ENTERO, autonumérico, >0

- fecha_inicio

Tipo: FECHA

- fecha_fin

Tipo: FECHA

- coste

Tipo: Numerico. Lo que nos cuesta a nosotros

- Precio_venta

Tipo: Numerico. El precio al que se vende al cliente

Relaciones

Relaciones que afectan a esta entidad:

- PROYECTO *asignado_presupuesto* PRESUPUESTO

Relaciones

PROYECTO <tiene_responsable> RECURSO

PROYECTO * *tiene_responsable* 1 RECURSO

Cardinalidad: N a 1

PROYECTO <en_estado> DESCR_ESTADO

PROYECTO * *en_estado* 1 DESCR_ESTADO

Cardinalidad: N a 1

PROYECTO <formado_por> FASE

PROYECTO 1 *dividido_en* * FASE

Cardinalidad: 1 a N

FASE <formado_por> TAREA

FASE *formado_por* * TAREA

Cardinalidad: 1 a N

PROYECTO <asignado_presupuesto> PRESUPUESTO

PROYECTO 1 *asignado_presupuesto* 1 PRESUPUESTO
Cardinalidad: 1 a 1

EXPEDIENTE <de_cliente> CLIENTE

EXPEDIENTE * *de_cliente* 1 CLIENTE
Cardinalidad: N a 1

PROYECTO <de_expediente> EXPEDIENTE

PROYECTO * *de_expediente* 1 EXPEDIENTE
Cardinalidad: N a 1

TAREA <necesita> TIPO_RECURSO

TAREA M *necesita* N TIPO_RECURSO

Cardinalidad: M a N

Atributos

- Cantidad

Cantidad de tiempo(horas) que va a estar dedicado la tarea un recurso concreto Tipo: ENTERO, >0

- Fecha inicio

Tipo: Fecha Cuando inicia esa tarea.

- Fecha fin

Tipo: Fecha Cuando finalice esa tarea

TAREA <depende_de> TAREA

TAREA * *depende_de* * TAREA

Cardinalidad: M a N

RECURSO <de_familia> FAMILIA_RECURSO

RECURSO * *de_familia* 1 FAMILIA_RECURSO

Cardinalidad: N a 1

PRODUCTO <de_familia> FAMILIA_PRODUCTO

PRODUCTO * *de_familia* 1 FAMILIA_PRODUCTO

Cardinalidad: N a 1

RECURSO <disponible> CALENDARIO

RECURSO 1 *disponible* 1 CALENDARIO

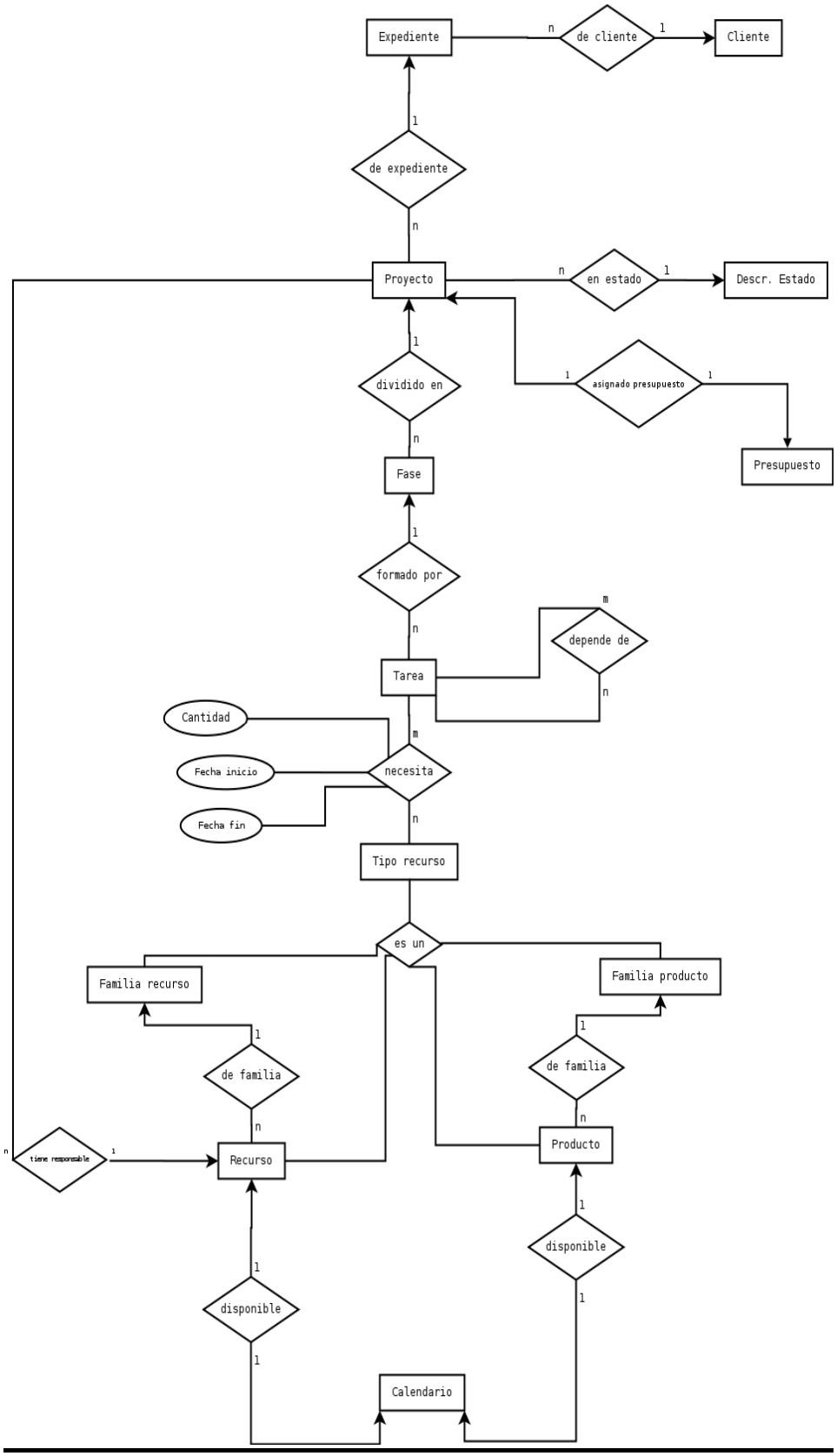
Cardinalidad: 1 a 1

PRODUCTO <disponible> CALENDARIO

PRODUCTO 1 *disponible* 1 CALENDARIO

Cardinalidad: 1 a 1

Diagrama Entidad-Relación.



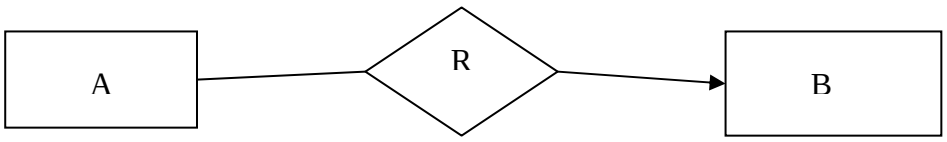
6. Paso de modelo E-R a modelo relacional.

El modelo entidad relación nos permite expresar con bastante precisión el esquema conceptual de nuestro proyecto, pero para la creación de nuestra de bases de datos vemos la necesidad de transformarlo a un esquema de relación de más bajo nivel, más cercano a la implementación soportada por los sistemas de gestión de bases de datos. Este esquema es el modelo relacional.

En este esquema, las entidades definidas como tales, no sufrirán alteración alguna. Los cambios vendrán en los atributos de las entidades y en las relaciones definidas en el modelo entidad-relación.

En el modelo entidad-relación básico se crea una tabla por cada entidad o relación definida. El modelo relacional nos permite la combinación de tablas dependiendo del tipo de relaciones que tengan sus entidades.

En nuestro esquema, la combinación de tablas que más vamos a utilizar será la que nos ofrezcan las relaciones 1 a N. Dichas relaciones tienen el siguiente esquema:



Dichas relaciones sólo podrán combinarse si la relación es total, es decir, que cada entidad A que posee la cardinalidad N participa en la relación R que queremos eliminar. Se combinarán las tablas de la entidad A con la tabla de la relación R.

Un ejemplo de ello es la siguiente relación:

(NOTA: Los atributos de las tablas que están en subrayado son las claves primarias de dichas tablas, mientras que los que están en cursiva son claves ajenas)

EXPEDIENTE <*de_cliente*> CLIENTE

La relación <de_cliente> es de N a 1. Como la relación es total, se puede combinar con la tabla expediente (que tiene cardinalidad N).

Por tanto, los campos de la tabla expediente serán:

EXPEDIENTE

Num_exp	***Id_cliente***

El mismo caso se presentará en las siguiente relaciones:

EXPEDIENTE <*de_expediente*> PROYECTO

La relación de <de_expediente> es de 1 a N. Se puede combinar con la tabla proyecto.

Los campos de la tabla proyecto serán:

PROYECTO

id_proyecto	Nombre	descripción	fecha_inicio	fecha_fin	***num_exp***	***responsable***	***id_estado***

PROYECTO <*en_estado*> DESCR. ESTADO

Relación de N a 1. Se combina con la tabla proyecto. Los campos son los mismos que en el apartado anterior.

PROYECTO <*dividido en*> FASE

Relación 1 a N. La relación se combina con la tabla fase. Los campos de la tabla fase son los siguientes:

FASE

id _ fase	***id_proyecto***	nombre

FASE <*formado por*> TAREA

Relación 1 a N. <formado por> se puede combinar con la tabla tarea. Los campos de tabla tarea son:

TAREA

Id _tarea	***Id _fase***	nombre	descripción	fecha inicio	fecha fin	restr_ fecha inicio	restr fecha fin

FAMILIA RECURSO <*de_familia*> RECURSO

Relación de 1 a N. La relación <de_familia> se combina con la tabla recurso. Los campos de la tabla recurso son:

RECURSO

id_tipo	***id_famrec***	nombre	estado	coste

FAMILIA PRODUCTO <*de_familia*> PRODUCTO

Relación de 1 a N. La relación <de_familia> se combina con la tabla producto. Los campos de la tabla producto son:

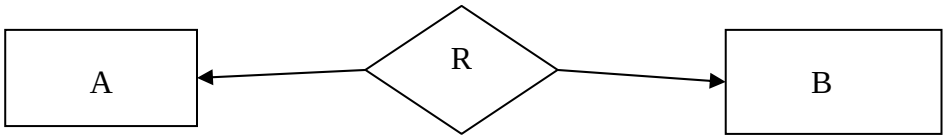
PRODUCTO

id_tipo	***id_fampro***	nombre	estado	coste

PROYECTO <*tiene_responsable*> RECURSO

Relación de N a 1. La relación <tiene_responsable> se combina con la tabla proyecto. Los campos de la tabla proyecto se han definido anteriormente.

Otro tipo de combinación de tablas que se da en el paso del modelo entidad-relación al modelo relacional se da en las relaciones 1 a 1. Es decir, las que tienen esta forma:



Los campos de la tabla relación R se pueden combinar con cualquiera de las tablas de las entidades de la relación.

Este tipo de combinación se puede aplicar en las siguientes relaciones:

PROYECTO <*asignado*> PRESUPUESTO

Relación 1 a 1. La relación <asignado> la vamos a combinar con la tabla presupuesto. Los campos de la tabla presupuesto son los siguientes:

PRESUPUESTO

id_presupuesto	***id_proyecto***	fecha_inicio	fecha_fin	coste	precio_venta

RECURSO <disponible> CALENDARIO

Relación 1 a 1. La relación <disponible> podemos combinarla con la tabla calendario. Los campos de la tabla calendario son los siguientes:

CALENDARIO

Id_tipo	año	dia001	dia002	…..	dia366

Otro paso es identificar las relaciones <es un> que nos indican herencia entre entidades. Se va a producir una *generalización*.

La relación <es un> no va a tener una tabla asociada en nuestro modelo relacional. Crearemos una tabla para la entidad de nivel más alto y una tabla por cada entidad de nivel más bajo en nuestro modelo entidad-relación añadiendo los atributos clave de las del nivel más alto.

Las entidades de nuestro modelo entidad-relación que tienen este comportamiento son:

TIPO RECURSO <*es un*> FAMILIA RECURSO

La tabla Familia Recurso coge los atributos clave de la tabla Tipo Recurso, produciéndose la generalización. Los campos de la tabla Familia Recurso son:

FAMILIA RECURSO

id_tipo	Descripción

TIPO RECURSO <*es un*> RECURSO

La tabla Recurso añade los atributos clave de la tabla Tipo Recurso mediante la generalización. Los campos de la tabla Recurso son los siguientes:

RECURSO

id_tipo	***id_famrec***	nombre	estado	coste

TIPO RECURSO <*es un*> FAMILIA PRODUCTO

La tabla Familia Producto coge los atributos clave de la tabla Tipo Recurso, produciéndose la generalización. Los campos de la tabla Familia Producto son:

FAMILIA PRODUCTO

id_tipo	Descripción

TIPO RECURSO <*es un*> PRODUCTO

La tabla Producto añade los atributos clave de la tabla Tipo Recurso mediante la generalización. Los campos de la tabla Producto son los siguientes:

PRODUCTO

id_tipo	***id_fampro***	nombre	estado	coste

Las entidades y relaciones restantes no necesitan transformación. Por tanto todas las tablas y sus campos se muestran a continuación, recordando que los nombres de los campos en subrayado se refieren a claves primarias, mientras que los que están en cursiva son claves ajenas.

CLIENTE

id_cliente	cif	telefono	direccion	ciudad	pais	email

EXPEDIENTE

Num_exp	***Id_cliente***

PROYECTO

id_proyecto	Nombre	descripción	fecha_inicio	fecha_fin	***num_exp***	***responsable***	***id_estado***

DESC. ESTADO

id_estado	nombre

PRESUPUESTO

id_presupuesto	***id_proyecto***	fecha_inicio	fecha_fin	coste	precio_venta

FASE

id _ fase	***id_proyecto***	nombre

TAREA

Id_tarea	***id_fase***	nombre	descripción	fecha_inicio	fecha_fin	restr_fecha_inicio	restr_fecha_fin

TIPO RECURSO

id_ _tipo	descripcion	coste

FAMILIA RECURSO

id_tipo	Descripción

FAMILIA PRODUCTO

id_tipo	Descripción

RECURSO

id_tipo	***id_famrec***	nombre	estado	coste

PRODUCTO

id_tipo	***id_fampro***	nombre	estado	coste

CALENDARIO

Id_tipo	año	dia001	dia002	…..	dia366

Las tablas de las relaciones que no se modifican son las siguientes:

DEPENDE DE

id_tarea1	**id_tarea2**

NECESITA

id_tarea	**id_tipo**	cantidad	fecha_inicio	fecha_fin

Las tablas que almacena la información de los proyectos una vez finalizados es el histórico que tiene como campos:

HISTORICO

id_recurso	**id_proyecto**	**dia**	horas

7. Diferentes tipos de planificación.

Para el correcto diseño de la herramienta, desarrollamos diferentes ejemplos de planificación:

Planificación hacia delante.

Para planificar hacia delante, vamos a tener en cuenta que esta planificación tiene que ser mínima en el tiempo, es decir, debemos situar todas las tareas y sus recursos en el menor tiempo posible.

Para ver cómo funciona nuestro algoritmo realizaremos un pequeño ejemplo de planificación de una fase de un proyecto con sus diferentes tareas.

Los datos de los que disponemos son los siguientes:

Recursos:

Id_tipo	Nombre	Coste
1	A	5
2	B	10
3	C	15

Disponibilidad de los recursos (en número de horas por día):

	RECURSOS		
	A	B	C
Dia001	8	8	8
Dia002	8	0	8
Dia003	4	8	8
Dia004	4	0	8
Dia005	8	8	8

Dia006	8	8	4
Dia007	8	8	4
Dia008	8	8	0
Dia009	4	8	0
Dia010	0	8	8
Dia011	8	8	8
Dia012	8	8	8
Dia013	4	8	4
Dia014	8	8	8
Dia015	4	8	8
Dia016	8	0	8
Dia017	8	8	0
Dia018	8	8	8
Dia019	8	4	8
Dia020	8	8	0

Tareas:

Id_tarea	**Id_fase**	**Nombre**	**Fecha_inicio**	**Fecha_fin**
1	1	Tarea1		
2	1	Tarea2		
3	1	Tarea3		
4	1	Tarea4		
5	1	Tarea5		
6	1	Tarea6		
7	1	Tarea7		
8	1	Tarea8		

Las tareas tendrán las siguientes dependencias entre sí:

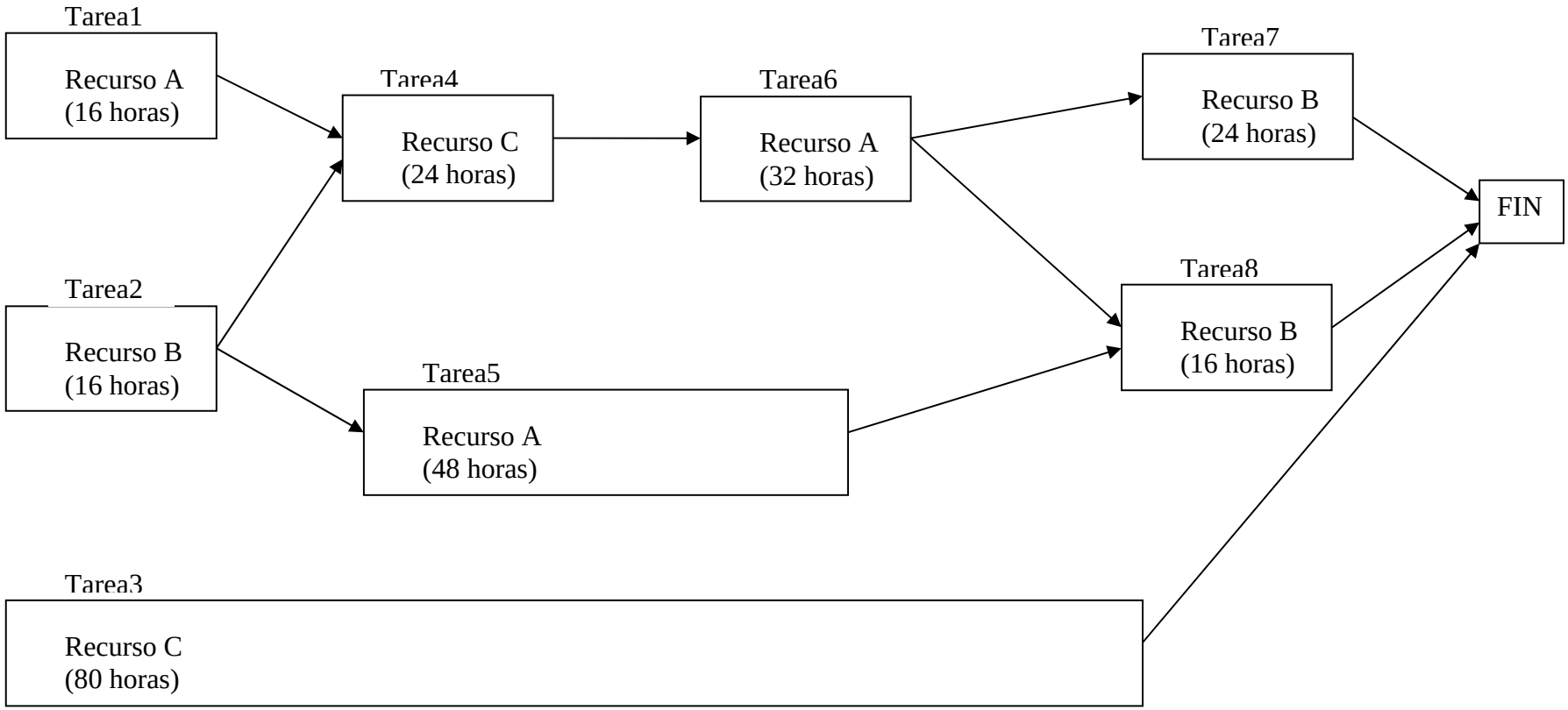
Relación Depende de

Id_tarea1	Id_tarea2
Tarea4	Tarea1
Tarea4	Tarea2
Tarea5	Tarea2
Tarea6	Tarea4
Tarea7	Tarea6
Tarea8	Tarea5
Tarea8	Tarea6

Asignación de tareas a recursos (cantidad es en horas):

Id_tarea	Id_tipo	Cantidad
1	1	16
2	2	16
3	3	80
4	3	24
5	1	48
6	1	32
7	2	24
8	3	16

La planificación de esta fase con las dependencias será de la siguiente manera:


Tarea1
Recurso A
(16 horas)
Tarea4
Recurso C
(24 horas)
Tarea6
Recurso A
(32 horas)
Tarea7
Recurso B
(24 horas)
FIN
Tarea2
Recurso B
(16 horas)
Tarea5
Recurso A
(48 horas)
Tarea8
Recurso B
(16 horas)
Tarea3
Recurso C
(80 horas)

La planificación se realizará de la siguiente manera:

Calculamos el camino crítico. En nuestro ejemplo el camino crítico será:

Tarea1, Tarea4, Tarea6, Tarea7

Seleccionaremos la primera tarea a planificar. Elegimos la tarea que antes comience en el tiempo. Comenzamos sacando los recursos de dicha tarea. En nuestro ejemplo tomamos la Tarea1, que tiene el recurso A.

Se observa la disponibilidad para ese recurso. Si hay suficiente disponibilidad, entonces se asigna el recurso a esos días. El recurso A tiene disponibilidad total, por tanto, lo asignamos:

Días	
1	**2**
A1	

Vamos calculando cada vez el camino crítico, dando prioridad a las tareas de camino crítico, y asignando recursos a las tareas comprobando su disponibilidad. Este proceso se repite hasta que no queden tareas sin planificar.

¿Qué pasa si el recurso no tiene disponibilidad o tiene disponibilidad pero no total en una fecha concreta?

Para el primer caso tenemos como ejemplo la asignación del recurso B a la tarea 2. Como el día 2 no dispone de ninguna hora, la asignación de las 8 horas restantes no se produce hasta el día siguiente (día 3), en el que tiene disponibilidad total.

Días		
1	**2**	**3**
B2		**B2**

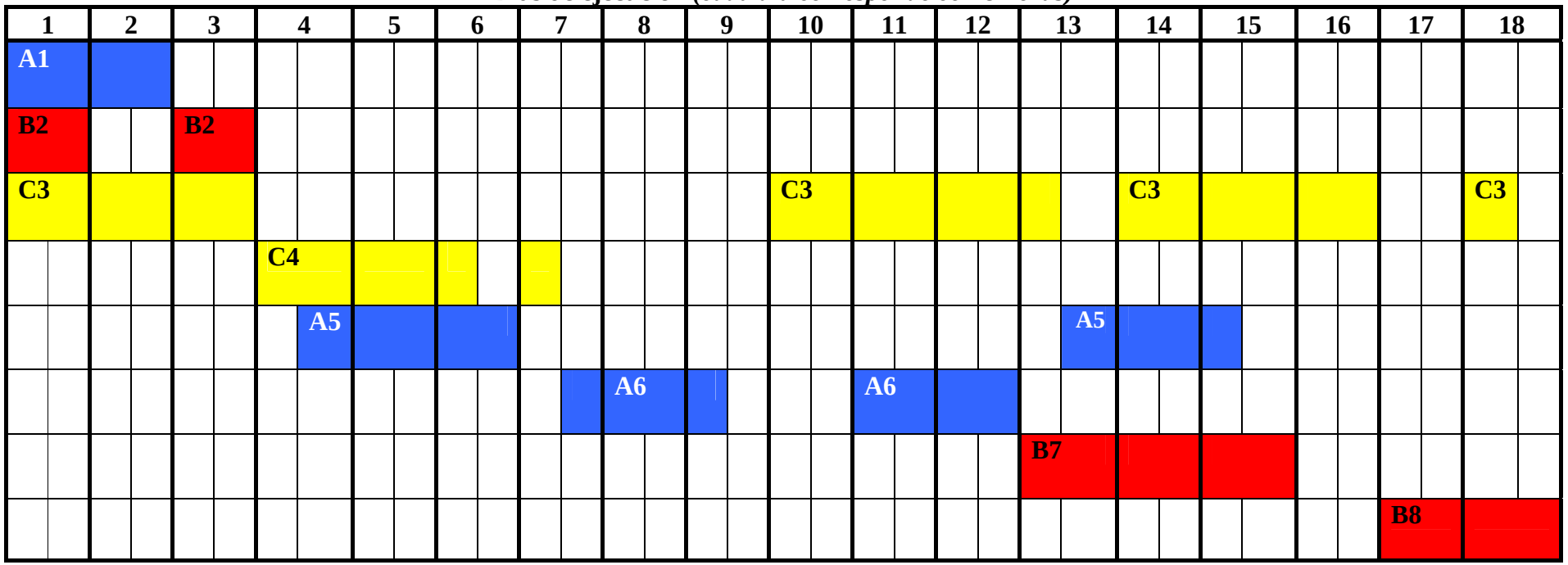
Para el segundo caso observamos la asignación del recurso C a la tarea 4 durante 24 horas. Vemos que el

día 6 y el día 7 el recurso sólo tiene disponibilidad de medio día (4 horas). Por tanto, la planificación del recurso se prolonga un día más:

Días					
4	**5**	**6**		**7**	
C4					

El resultado final del proceso es:

Días de ejecución ***(cada día corresponde con 8 horas)***



Planificación hacia atrás.

Para planificar hacia atrás, tenemos una fecha límite para la ejecución de todas las tareas del proyecto, así que nuestro algoritmo debe asignar los recursos a las tareas de manera que cumplan un plazo establecido.

Para ver cómo funciona nuestro algoritmo realizaremos un pequeño ejemplo de planificación de una fase de un proyecto con sus diferentes tareas.
Los datos de los que disponemos son los siguientes:

Recursos:

Id_tipo	Nombre	Coste
1	A	5
2	B	10
3	C	15

Disponibilidad de los recursos (en número de horas por día):

	RECURSOS		
	A	B	C
Dia001	8	8	8
Dia002	8	8	8
Dia003	8	8	8
Dia004	8	0	8
Dia005	4	8	8
Dia006	4	0	8
Dia007	8	8	8
Dia008	8	8	4
Dia009	8	8	4
Dia010	8	8	0
Dia011	4	8	0
Dia012	0	8	8
Dia013	8	8	8
Dia014	8	8	8
Dia015	4	8	4
Dia016	8	8	8

Dia017	4	8	8
Dia018	8	0	8
Dia019	8	8	0
Dia020	8	8	8
Dia021	8	4	8
Dia022	8	8	0

Tareas:

Id_tarea	**Id_fase**	**Nombre**	**Fecha_inicio**	**Fecha_fin**
1	1	Tarea1		
2	1	Tarea2		
3	1	Tarea3		
4	1	Tarea4		
5	1	Tarea5		
6	1	Tarea6		
7	1	Tarea7		
8	1	Tarea8		

Las tareas tendrán las siguientes dependencias entre sí:

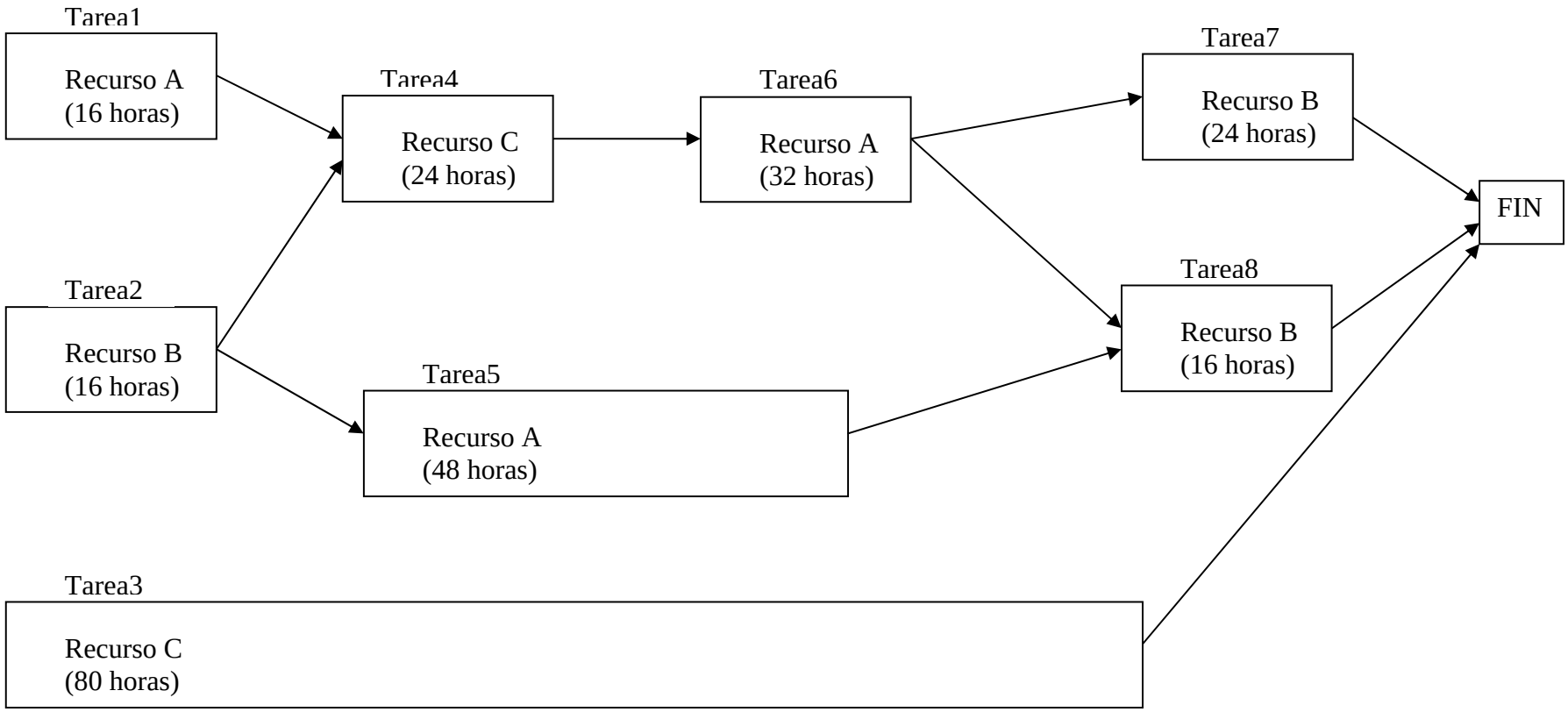
Relación Depende de

Id_tarea1	**Id_tarea2**
Tarea4	Tarea1
Tarea4	Tarea2
Tarea5	Tarea2
Tarea6	Tarea4
Tarea7	Tarea6
Tarea8	Tarea5
Tarea8	Tarea6

Asignación de tareas a recursos (cantidad es en horas):

Id_tarea	Id_tipo	Cantidad
1	1	16
2	2	16
3	3	80
4	3	24
5	1	48
6	1	32
7	2	24
8	2	16

La planificación de esta fase con las dependencias será de la siguiente manera:


Tarea1
Recurso A
(16 horas)
Tarea2
Recurso B
(16 horas)
Tarea4
Recurso C
(24 horas)
Tarea5
Recurso A
(48 horas)
Tarea6
Recurso A
(32 horas)
Tarea7
Recurso B
(24 horas)
Tarea8
Recurso B
(16 horas)
FIN
Tarea3
Recurso C
(80 horas)

La planificación se realizará de la siguiente manera:

Calculamos el camino crítico. En nuestro ejemplo el camino crítico será:

Tarea1, Tarea4, Tarea6, Tarea7

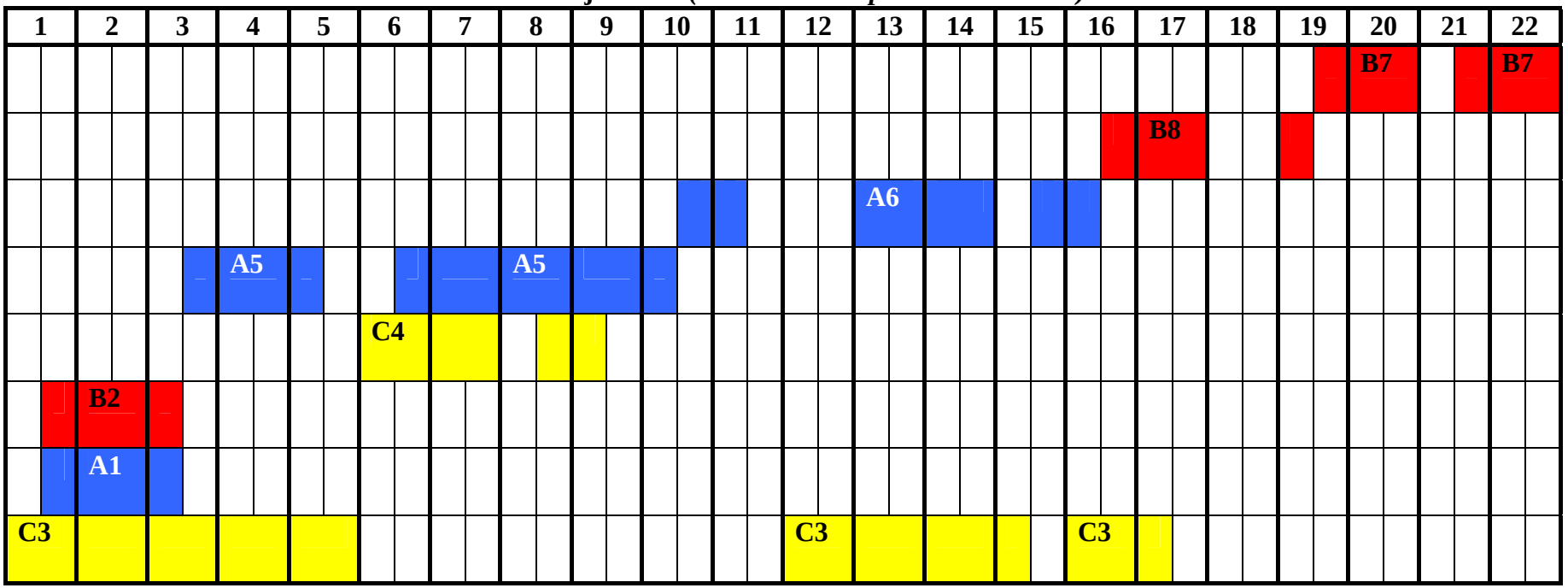
Seleccionaremos la primera tarea a planificar, que corresponderá con una tarea del camino crítico. Elegimos la tarea que más tarde comience en el tiempo. Comenzamos sacando los recursos de dicha tarea. En nuestro ejemplo tomamos la Tarea7, que tiene el recurso B.

Se observa la disponibilidad para ese recurso. Si hay suficiente disponibilidad, entonces se asigna el recurso a esos días. La manera de asignar la disponibilidad es la misma que en el ejemplo de la planificación hacia delante.

Vamos calculando cada vez el camino crítico, dando prioridad a las tareas de camino crítico, y asignando recursos a las tareas comprobando su disponibilidad. Este proceso se repite hasta que no queden tareas sin planificar.

El resultado final del proceso es:

Días de ejecución ***(cada día corresponde con 8 horas)***



Planificación por costes.

Para planificar por costes, tenemos en cuenta el coste por hora que tiene cada recurso. Nuestro objetivo será minimizar el coste de la planificación, es decir, crearemos una planificación de tareas que haga que nuestro proyecto tenga el menor coste.

Mediante un ejemplo de planificación de una fase de un proyecto con sus diferentes tareas comprenderemos mejor el funcionamiento de nuestro algoritmo.

Los datos de los que disponemos son los siguientes:

Recursos:

Id_tipo	Nombre	Coste
1	A	5
2	B	10
3	C	15

Disponibilidad de los recursos (en número de horas por día):

	RECURSOS		
	A	B	C
Dia001	8	8	8
Dia002	8	0	8
Dia003	4	8	8
Dia004	4	0	8
Dia005	8	8	8
Dia006	8	8	4
Dia007	8	8	4
Dia008	8	8	0
Dia009	4	8	0
Dia010	0	8	8
Dia011	8	8	8
Dia012	8	8	8
Dia013	4	8	4
Dia014	8	8	8
Dia015	4	8	8
Dia016	8	0	8

Dia017	8	8	0
Dia018	8	8	8
Dia019	8	4	8
Dia020	8	8	0
Dia021	8	4	8
Dia022	4	8	8
Dia023	8	8	8
Dia024	8	8	4
Dia025	8	8	0

Tareas:

Id_tarea	**Id_fase**	**Nombre**	**Fecha_inicio**	**Fecha_fin**
1	1	Tarea1		
2	1	Tarea2		
3	1	Tarea3		
4	1	Tarea4		
5	1	Tarea5		
6	1	Tarea6		
7	1	Tarea7		
8	1	Tarea8		

Las tareas tendrán las siguientes dependencias entre sí:

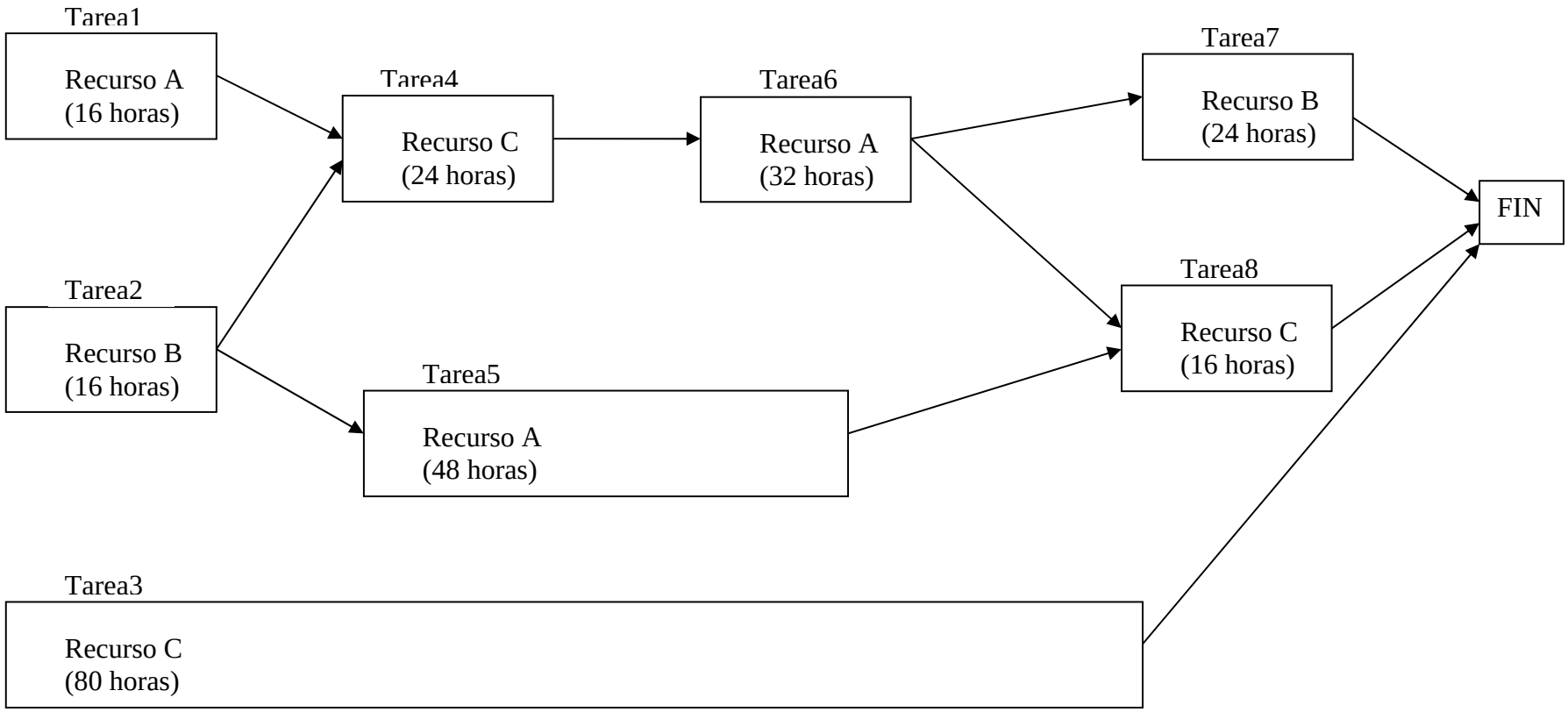
Relación Depende de

Id_tarea1	**Id_tarea2**
Tarea4	Tarea1
Tarea4	Tarea2
Tarea5	Tarea2
Tarea6	Tarea4
Tarea7	Tarea6
Tarea8	Tarea5
Tarea8	Tarea6

Asignación de tareas a recursos (cantidad es en horas):

Id_tarea	Id_tipo	Cantidad
1	1	16
2	2	16
3	3	80
4	3	24
5	1	48
6	1	32
7	2	24
8	3	16

La planificación de esta fase con las dependencias será de la siguiente manera:


Tarea1
Recurso A
(16 horas)
Tarea4
Recurso C
(24 horas)
Tarea6
Recurso A
(32 horas)
Tarea7
Recurso B
(24 horas)
FIN
Tarea2
Recurso B
(16 horas)
Tarea8
Recurso C
(16 horas)
Tarea5
Recurso A
(48 horas)
Tarea3
Recurso C
(80 horas)

La planificación por coste se realizará de la siguiente manera:

Calculamos el camino crítico. Para ello estudiamos qué tarea tiene mayor coste. En nuestro ejemplo el camino crítico será:

Tarea3

Ya que el recurso C es el más caro en coste por hora (15€/h). Tendríamos por tanto un coste de:

80 horas x 15 €/hora = 1200 €

Es la primera tarea que se planificaría. El único recurso que actúa en esta tarea es el recurso C. Miramos su disponibilidad y vamos asignando el recurso a esos días.

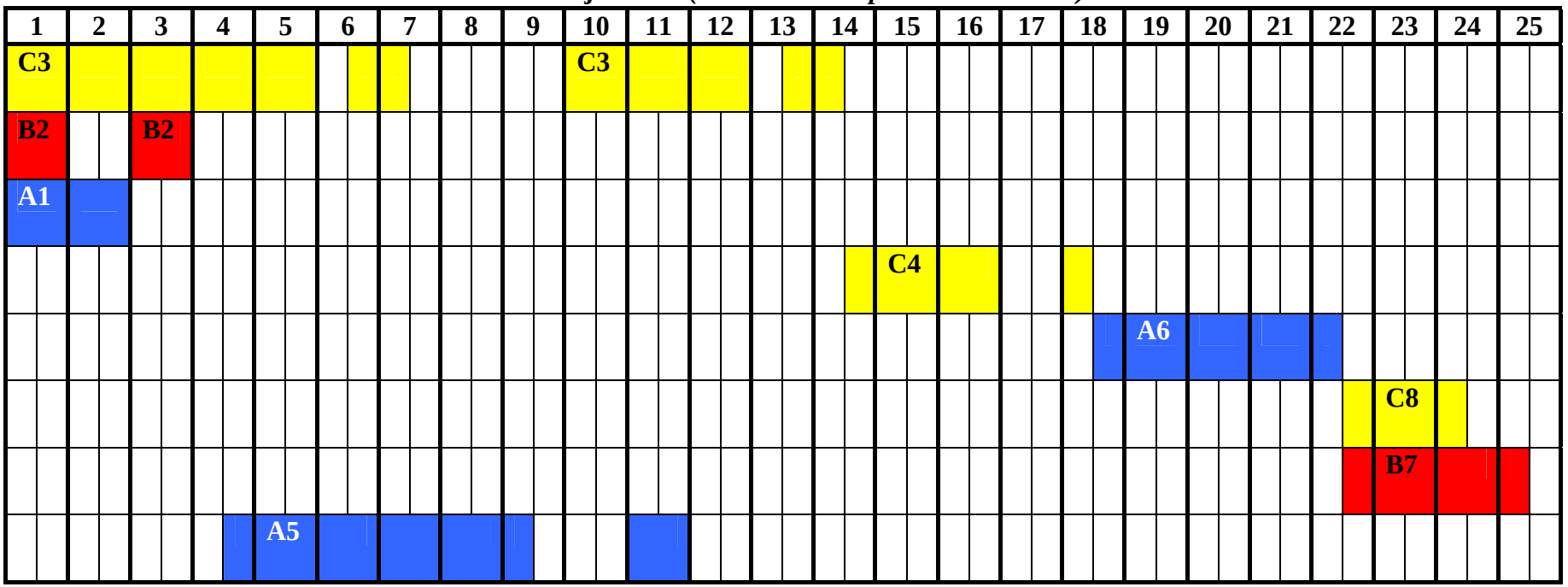
En nuestro ejemplo, el siguiente camino crítico por costes a tratar sería el formado por las tareas:

Tarea2, Tarea4, Tarea6, Tarea8

Vamos calculando cada vez el camino crítico, dando prioridad a las tareas de camino crítico, y asignando recursos a las tareas comprobando su disponibilidad. Este proceso se repite hasta que no queden tareas sin planificar.

El resultado final del proceso es:

Días de ejecución ***(cada día corresponde con 8 horas)***



8. Código SQL de la base de datos.

Para crear la base de datos hemos utilizado un lenguaje SQL de libre distribución llamado MySQL, por lo que la sintaxis es la propia de dicho lenguaje.

```
/* ARCHIVO CREACION BASE DE DATOS DEL PLANIFICADOR DE TAREAS
*/

create database if not exists Planificador;
/*
*/

use Planificador;

/*
 ENTIDAD CLIENTE
*/

drop table if exists Cliente;

create table Cliente(
id_cliente INT NOT NULL AUTO_INCREMENT,
cif VARCHAR(9) NOT NULL,
telefono VARCHAR(9),
direccion VARCHAR(30),
ciudad VARCHAR(20),
pais VARCHAR(20),
email VARCHAR(30),
primary key (id_cliente)
);

/*
 ENTIDAD EXPEDIENTE
*/

drop table if exists Expediente;

create table Expediente(
num_exp INT NOT NULL AUTO_INCREMENT,
id_cliente INT NOT NULL,
primary key(num_exp),
foreign key(id_cliente) REFERENCES Cliente(id_cliente)
);

/*
 ENTIDAD PROYECTO
*/
```

```
drop table if exists Proyecto;

create table Proyecto(
id_proyecto INT NOT NULL AUTO_INCREMENT,
nombre VARCHAR(20),
descripcion VARCHAR(50),
fecha_inicio DATE,
fecha_fin DATE,
num_exp INT NOT NULL,
responsable INT NOT NULL,
id_estado INT NOT NULL,
primary key(id_proyecto),
foreign key(responsable) REFERENCES Recurso(id_tipo),
foreign key(num_exp) REFERENCES Expediente(num_exp),
foreign key(id_estado) REFERENCES Descr_estado(id_estado)
);

/*
 ENTIDAD DESCRIPCION ESTADO
*/

drop table if exists Descr_estado;

create table Descr_estado(
id_estado INT NOT NULL AUTO_INCREMENT,
nombre VARCHAR(20),
primary key(id_estado)
);

/*
 ENTIDAD PRESUPUESTO
*/

drop table if exists Presupuesto;

-- Incluimos la relacion asignado presupuesto, ya que es una relacion 1 a 1

create table Presupuesto(
id_presupuesto INT NOT NULL AUTO_INCREMENT,
id_proyecto INT NOT NULL,
fecha_inicio DATE,
fecha_fin DATE,
coste FLOAT,
precio_venta FLOAT,
primary key(id_presupuesto),
foreign key (id_proyecto) REFERENCES Proyecto(id_proyecto)
);

/*
 ENTIDAD FASE
*/

drop table if exists Fase;
```

```
create table Fase(
id_fase INT NOT NULL AUTO_INCREMENT,
id_proyecto INT NOT NULL,
nombre VARCHAR(20),
primary key(id_fase),
foreign key (id_proyecto) REFERENCES Proyecto(id_proyecto)
);

/*
  ENTIDAD TAREA

*/

drop table if exists Tarea;

create table Tarea(
id_tarea INT NOT NULL AUTO_INCREMENT,
id_fase INT NOT NULL,
nombre VARCHAR(20),
descripcion VARCHAR(50),
fecha_inicio DATE,
fecha_fin DATE,
restr_fecha_inicio DATE,
restr_fecha_fin DATE,
primary key(id_tarea),
foreign key (id_fase) REFERENCES Fase(id_fase)
);

/*
 ENTIDAD TIPO RECURSO
*/

drop table if exists Tipo_recurso;

create table Tipo_recurso(
id_tipo INT NOT NULL AUTO_INCREMENT,
descripcion VARCHAR(50),
coste FLOAT,
primary key(id_tipo)
);

/*
 ENTIDAD FAMILIA RECURSO
*/

-- Hereda de Tipo_Recurso. Generalización

drop table if exists Familia_recurso;

create table Familia_recurso(
id_tipo INT NOT NULL,
```

```
descripcion VARCHAR(50),
primary key(id_tipo),
foreign key (id_tipo) REFERENCES Tipo_recurso(id_tipo)
);



/*
 ENTIDAD FAMILIA PRODUCTO
*/

-- Hereda de Tipo_Recurso. Generalización

drop table if exists Familia_producto;

create table Familia_producto(
id_tipo INT NOT NULL,
descripcion VARCHAR(50),
primary key(id_tipo),
foreign key (id_tipo) REFERENCES Tipo_recurso(id_tipo)
);


/*
 ENTIDAD RECURSO
*/

-- Hereda de Tipo_Recurso. Generalización

drop table if exists Recurso;

create table Recurso(
id_tipo INT NOT NULL,
id_famrec INT NOT NULL,
nombre VARCHAR(20),
estado VARCHAR(20),
coste FLOAT,
primary key(id_tipo),
foreign key (id_tipo) REFERENCES Tipo_recurso(id_tipo),
foreign key (id_famrec) REFERENCES Familia_recurso(id_tipo)
);


/*
 ENTIDAD PRODUCTO
*/

-- Hereda de Tipo_Recurso. Generalización

drop table if exists Producto;
```

```
create table Producto(
id_tipo INT NOT NULL,
id_fampro INT NOT NULL,
nombre VARCHAR(20),
estado VARCHAR(20),
coste FLOAT,
primary key(id_tipo),
foreign key (id_tipo) REFERENCES Tipo_recurso(id_tipo),
foreign key (id_fampro) REFERENCES Familia_producto(id_tipo)
);


/*
 Calendario
*/

drop table if exists Calendario;

create table Calendario(
id_tipo INT NOT NULL,
año INT,
dia001 DECIMAL (4,2),
dia002 DECIMAL (4,2),
dia003 DECIMAL (4,2),
dia004 DECIMAL (4,2),
dia005 DECIMAL (4,2),
dia006 DECIMAL (4,2),
dia007 DECIMAL (4,2),
dia008 DECIMAL (4,2),
dia009 DECIMAL (4,2),
dia010 DECIMAL (4,2),
dia011 DECIMAL (4,2),
dia012 DECIMAL (4,2),
dia013 DECIMAL (4,2),
dia014 DECIMAL (4,2),
dia015 DECIMAL (4,2),
dia016 DECIMAL (4,2),
dia017 DECIMAL (4,2),
dia018 DECIMAL (4,2),
dia019 DECIMAL (4,2),
dia020 DECIMAL (4,2),
dia021 DECIMAL (4,2),
dia022 DECIMAL (4,2),
dia023 DECIMAL (4,2),
dia024 DECIMAL (4,2),
dia025 DECIMAL (4,2),
dia026 DECIMAL (4,2),
dia027 DECIMAL (4,2),
dia028 DECIMAL (4,2),
dia029 DECIMAL (4,2),
dia030 DECIMAL (4,2),
dia031 DECIMAL (4,2),
dia032 DECIMAL (4,2),
dia033 DECIMAL (4,2),
dia034 DECIMAL (4,2),
dia035 DECIMAL (4,2),
dia036 DECIMAL (4,2),
dia037 DECIMAL (4,2),
dia038 DECIMAL (4,2),
dia039 DECIMAL (4,2),
dia040 DECIMAL (4,2),
dia041 DECIMAL (4,2),
dia042 DECIMAL (4,2),
dia043 DECIMAL (4,2),
dia044 DECIMAL (4,2),
dia045 DECIMAL (4,2),
dia046 DECIMAL (4,2),
dia047 DECIMAL (4,2),
dia048 DECIMAL (4,2),
dia049 DECIMAL (4,2),
dia050 DECIMAL (4,2),
dia051 DECIMAL (4,2),
dia052 DECIMAL (4,2),
dia053 DECIMAL (4,2),
dia054 DECIMAL (4,2),
dia055 DECIMAL (4,2),
dia056 DECIMAL (4,2),
dia057 DECIMAL (4,2),
dia058 DECIMAL (4,2),
dia059 DECIMAL (4,2),
dia060 DECIMAL (4,2),
dia061 DECIMAL (4,2),
dia062 DECIMAL (4,2),
dia063 DECIMAL (4,2),
dia064 DECIMAL (4,2),
```

dia065 DECIMAL (4,2),
dia066 DECIMAL (4,2),
dia067 DECIMAL (4,2),
dia068 DECIMAL (4,2),
dia069 DECIMAL (4,2),
dia070 DECIMAL (4,2),
dia071 DECIMAL (4,2),
dia072 DECIMAL (4,2),
dia073 DECIMAL (4,2),
dia074 DECIMAL (4,2),
dia075 DECIMAL (4,2),
dia076 DECIMAL (4,2),
dia077 DECIMAL (4,2),
dia078 DECIMAL (4,2),
dia079 DECIMAL (4,2),
dia080 DECIMAL (4,2),
dia081 DECIMAL (4,2),
dia082 DECIMAL (4,2),
dia083 DECIMAL (4,2),
dia084 DECIMAL (4,2),
dia085 DECIMAL (4,2),
dia086 DECIMAL (4,2),
dia087 DECIMAL (4,2),
dia088 DECIMAL (4,2),
dia089 DECIMAL (4,2),
dia090 DECIMAL (4,2),
dia091 DECIMAL (4,2),
dia092 DECIMAL (4,2),
dia093 DECIMAL (4,2),
dia094 DECIMAL (4,2),
dia095 DECIMAL (4,2),
dia096 DECIMAL (4,2),
dia097 DECIMAL (4,2),
dia098 DECIMAL (4,2),
dia099 DECIMAL (4,2),
dia100 DECIMAL (4,2),
dia101 DECIMAL (4,2),
dia102 DECIMAL (4,2),
dia103 DECIMAL (4,2),
dia104 DECIMAL (4,2),
dia105 DECIMAL (4,2),
dia106 DECIMAL (4,2),
dia107 DECIMAL (4,2),
dia108 DECIMAL (4,2),
dia109 DECIMAL (4,2),
dia110 DECIMAL (4,2),
dia111 DECIMAL (4,2),
dia112 DECIMAL (4,2),
dia113 DECIMAL (4,2),
dia114 DECIMAL (4,2),
dia115 DECIMAL (4,2),
dia116 DECIMAL (4,2),
dia117 DECIMAL (4,2),
dia118 DECIMAL (4,2),
dia119 DECIMAL (4,2),
dia120 DECIMAL (4,2),
dia121 DECIMAL (4,2),
dia122 DECIMAL (4,2),
dia123 DECIMAL (4,2),
dia124 DECIMAL (4,2),
dia125 DECIMAL (4,2),
dia126 DECIMAL (4,2),
dia127 DECIMAL (4,2),
dia128 DECIMAL (4,2),
dia129 DECIMAL (4,2),
dia130 DECIMAL (4,2),
dia131 DECIMAL (4,2),
dia132 DECIMAL (4,2),
dia133 DECIMAL (4,2),
dia134 DECIMAL (4,2),
dia135 DECIMAL (4,2),
dia136 DECIMAL (4,2),
dia137 DECIMAL (4,2),
dia138 DECIMAL (4,2),
dia139 DECIMAL (4,2),
dia140 DECIMAL (4,2),
dia141 DECIMAL (4,2),
dia142 DECIMAL (4,2),
dia143 DECIMAL (4,2),
dia144 DECIMAL (4,2),
dia145 DECIMAL (4,2),
dia146 DECIMAL (4,2),
dia147 DECIMAL (4,2),
dia148 DECIMAL (4,2),
dia149 DECIMAL (4,2),
dia150 DECIMAL (4,2),
dia151 DECIMAL (4,2),
dia152 DECIMAL (4,2),
dia153 DECIMAL (4,2),
dia154 DECIMAL (4,2),
dia155 DECIMAL (4,2),
dia156 DECIMAL (4,2),
dia157 DECIMAL (4,2),
dia158 DECIMAL (4,2),
dia159 DECIMAL (4,2),
dia160 DECIMAL (4,2),
dia161 DECIMAL (4,2),
dia162 DECIMAL (4,2),
dia163 DECIMAL (4,2),
dia164 DECIMAL (4,2),
dia165 DECIMAL (4,2),
dia166 DECIMAL (4,2),
dia167 DECIMAL (4,2),
dia168 DECIMAL (4,2),
dia169 DECIMAL (4,2),
dia170 DECIMAL (4,2),

dia171 DECIMAL (4,2),
dia172 DECIMAL (4,2),
dia173 DECIMAL (4,2),
dia174 DECIMAL (4,2),
dia175 DECIMAL (4,2),
dia176 DECIMAL (4,2),
dia177 DECIMAL (4,2),
dia178 DECIMAL (4,2),
dia179 DECIMAL (4,2),
dia180 DECIMAL (4,2),
dia181 DECIMAL (4,2),
dia182 DECIMAL (4,2),
dia183 DECIMAL (4,2),
dia184 DECIMAL (4,2),
dia185 DECIMAL (4,2),
dia186 DECIMAL (4,2),
dia187 DECIMAL (4,2),
dia188 DECIMAL (4,2),
dia189 DECIMAL (4,2),
dia190 DECIMAL (4,2),
dia191 DECIMAL (4,2),
dia192 DECIMAL (4,2),
dia193 DECIMAL (4,2),
dia194 DECIMAL (4,2),
dia195 DECIMAL (4,2),
dia196 DECIMAL (4,2),
dia197 DECIMAL (4,2),
dia198 DECIMAL (4,2),
dia199 DECIMAL (4,2),
dia200 DECIMAL (4,2),
dia201 DECIMAL (4,2),
dia202 DECIMAL (4,2),
dia203 DECIMAL (4,2),
dia204 DECIMAL (4,2),
dia205 DECIMAL (4,2),
dia206 DECIMAL (4,2),
dia207 DECIMAL (4,2),
dia208 DECIMAL (4,2),
dia209 DECIMAL (4,2),
dia210 DECIMAL (4,2),
dia211 DECIMAL (4,2),
dia212 DECIMAL (4,2),
dia213 DECIMAL (4,2),
dia214 DECIMAL (4,2),
dia215 DECIMAL (4,2),
dia216 DECIMAL (4,2),
dia217 DECIMAL (4,2),
dia218 DECIMAL (4,2),
dia219 DECIMAL (4,2),
dia220 DECIMAL (4,2),
dia221 DECIMAL (4,2),
dia222 DECIMAL (4,2),
dia223 DECIMAL (4,2),
dia224 DECIMAL (4,2),
dia225 DECIMAL (4,2),
dia226 DECIMAL (4,2),
dia227 DECIMAL (4,2),
dia228 DECIMAL (4,2),
dia229 DECIMAL (4,2),
dia230 DECIMAL (4,2),
dia231 DECIMAL (4,2),
dia232 DECIMAL (4,2),
dia233 DECIMAL (4,2),
dia234 DECIMAL (4,2),
dia235 DECIMAL (4,2),
dia236 DECIMAL (4,2),
dia237 DECIMAL (4,2),
dia238 DECIMAL (4,2),
dia239 DECIMAL (4,2),
dia240 DECIMAL (4,2),
dia241 DECIMAL (4,2),
dia242 DECIMAL (4,2),
dia243 DECIMAL (4,2),
dia244 DECIMAL (4,2),
dia245 DECIMAL (4,2),
dia246 DECIMAL (4,2),
dia247 DECIMAL (4,2),
dia248 DECIMAL (4,2),
dia249 DECIMAL (4,2),
dia250 DECIMAL (4,2),
dia251 DECIMAL (4,2),
dia252 DECIMAL (4,2),
dia253 DECIMAL (4,2),
dia254 DECIMAL (4,2),
dia255 DECIMAL (4,2),
dia256 DECIMAL (4,2),
dia257 DECIMAL (4,2),
dia258 DECIMAL (4,2),
dia259 DECIMAL (4,2),
dia260 DECIMAL (4,2),
dia261 DECIMAL (4,2),
dia262 DECIMAL (4,2),
dia263 DECIMAL (4,2),
dia264 DECIMAL (4,2),
dia265 DECIMAL (4,2),
dia266 DECIMAL (4,2),
dia267 DECIMAL (4,2),
dia268 DECIMAL (4,2),
dia269 DECIMAL (4,2),
dia270 DECIMAL (4,2),
dia271 DECIMAL (4,2),
dia272 DECIMAL (4,2),
dia273 DECIMAL (4,2),
dia274 DECIMAL (4,2),
dia275 DECIMAL (4,2),
dia276 DECIMAL (4,2),

```
dia277 DECIMAL (4,2),
dia278 DECIMAL (4,2),
dia279 DECIMAL (4,2),
dia280 DECIMAL (4,2),
dia281 DECIMAL (4,2),
dia282 DECIMAL (4,2),
dia283 DECIMAL (4,2),
dia284 DECIMAL (4,2),
dia285 DECIMAL (4,2),
dia286 DECIMAL (4,2),
dia287 DECIMAL (4,2),
dia288 DECIMAL (4,2),
dia289 DECIMAL (4,2),
dia290 DECIMAL (4,2),
dia291 DECIMAL (4,2),
dia292 DECIMAL (4,2),
dia293 DECIMAL (4,2),
dia294 DECIMAL (4,2),
dia295 DECIMAL (4,2),
dia296 DECIMAL (4,2),
dia297 DECIMAL (4,2),
dia298 DECIMAL (4,2),
dia299 DECIMAL (4,2),
dia300 DECIMAL (4,2),
dia301 DECIMAL (4,2),
dia302 DECIMAL (4,2),
dia303 DECIMAL (4,2),
dia304 DECIMAL (4,2),
dia305 DECIMAL (4,2),
dia306 DECIMAL (4,2),
dia307 DECIMAL (4,2),
dia308 DECIMAL (4,2),
dia309 DECIMAL (4,2),
dia310 DECIMAL (4,2),
dia311 DECIMAL (4,2),
dia312 DECIMAL (4,2),
dia313 DECIMAL (4,2),
dia314 DECIMAL (4,2),
dia315 DECIMAL (4,2),
dia316 DECIMAL (4,2),
dia317 DECIMAL (4,2),
dia318 DECIMAL (4,2),
dia319 DECIMAL (4,2),
dia320 DECIMAL (4,2),
dia321 DECIMAL (4,2),
dia322 DECIMAL (4,2),
dia323 DECIMAL (4,2),
dia324 DECIMAL (4,2),
dia325 DECIMAL (4,2),
dia326 DECIMAL (4,2),
dia327 DECIMAL (4,2),
dia328 DECIMAL (4,2),
dia329 DECIMAL (4,2),
dia330 DECIMAL (4,2),
dia331 DECIMAL (4,2),
dia332 DECIMAL (4,2),
dia333 DECIMAL (4,2),
dia334 DECIMAL (4,2),
dia335 DECIMAL (4,2),
dia336 DECIMAL (4,2),
dia337 DECIMAL (4,2),
dia338 DECIMAL (4,2),
dia339 DECIMAL (4,2),
dia340 DECIMAL (4,2),
dia341 DECIMAL (4,2),
dia342 DECIMAL (4,2),
dia343 DECIMAL (4,2),
dia344 DECIMAL (4,2),
dia345 DECIMAL (4,2),
dia346 DECIMAL (4,2),
dia347 DECIMAL (4,2),
dia348 DECIMAL (4,2),
dia349 DECIMAL (4,2),
dia350 DECIMAL (4,2),
dia351 DECIMAL (4,2),
dia352 DECIMAL (4,2),
dia353 DECIMAL (4,2),
dia354 DECIMAL (4,2),
dia355 DECIMAL (4,2),
dia356 DECIMAL (4,2),
dia357 DECIMAL (4,2),
dia358 DECIMAL (4,2),
dia359 DECIMAL (4,2),
dia360 DECIMAL (4,2),
dia361 DECIMAL (4,2),
dia362 DECIMAL (4,2),
dia363 DECIMAL (4,2),
dia364 DECIMAL (4,2),
dia365 DECIMAL (4,2),
dia366 DECIMAL (4,2),

foreign key (id_tipo) REFERENCES Tipo_Recurso(id_tipo)
);

/*
 Historico
*/
```

```
drop table if exists Historico;

create table Historico(
id_recurso INT NOT NULL,
id_proyecto INT,
dia DATE,
horas DECIMAL (4,2),
primary key(id_recurso,dia,id_proyecto)
);


/*
 RELACIONES
*/


/*
TAREA <depende de> TAREA
Relacion m a n
*/

drop table if exists DependeDe;

create table DependeDe(
id_tarea1 INT NOT NULL,
id_tarea2 INT NOT NULL,
primary key(id_tarea1,id_tarea2),
foreign key (id_tarea1) REFERENCES Tarea(id_tarea),
foreign key (id_tarea2) REFERENCES Tarea(id_tarea)
);

/*
TAREA <necesita> TIPO_RECURSO
Relacion m a n
*/

drop table if exists Necesita;

create table Necesita(
id_tarea INT NOT NULL,
id_tipo INT NOT NULL,
cantidad FLOAT,
fecha_inicio DATE,
fecha_fin DATE,
primary key(id_tarea,id_tipo),
foreign key (id_tarea) REFERENCES Tarea(id_tarea),
foreign key (id_tipo) REFERENCES Tipo_recurso(id_tipo)
);
```

9. Implementación de algoritmos.

Planificación hacia delante.

El proceso de “planificar hacia delante” se realiza de la siguiente forma:

En un principio se empieza con todas las tareas sin planificar, y termina cuando todas las tareas han sido planificadas. El algoritmo va seleccionando las tareas una a una y calculando su planificación optima.

El orden de selección de las tareas es decisivo para que el resultado final cumpla la estrategia. Para obtener el resultado en una sola iteración y evitar otros métodos como la *vuelta atrás*, se hacen las siguientes consideraciones:

- Una tarea se podrá planificar sólo si todas las tareas de las que depende ya han sido planificadas
- Se intentará planificar antes aquellas tareas que por su longitud estimada sean más decisivas en la duración total del proyecto. El conjunto de estas tareas son las que forman el *camino crítico*.
- En el momento del calculo de la planificación de una tarea se deberán tener en cuenta factores como la *fragmentación* del tiempo que los recursos dedicarán a ella, la *simultaneidad* de éstos.

En cada iteración por tanto se selecciona la tarea a tratar de la siguiente forma:

- Primero se identifican las tareas que todavía no están planificadas
- Se calcula el *camino crítico* en la situación actual, con la información de las tareas ya planificadas
- Las tareas no planificadas se separan en aquellas que pertenecen al camino crítico y las que no
- En los grupos de tareas resultantes se eliminan aquellas tareas que no cumplen la condición de depender totalmente de tareas ya planificadas
- Si hay tareas críticas, se seleccionará la menor de éstas.

Si no, la menor de las no críticas.
Se considerará la menos crítica aquella que tenga la fecha de comienzo más temprana

Una vez seleccionada la tarea, se deberán calcular los intervalos de fechas que se le asigna a cada recurso. El procedimiento es:

- Se crea una estructura de tuplas, una para cada recurso que necesita la tarea, que contienen información acerca de:

  · Recurso concreto asignado
  · Lista de fechas asignada al recurso
  · Variable interna: Fecha desde la que evalúa el siguiente intervalo de fechas
  · Variable interna: Cantidad de horas que quedan por asignar

- Se asignan las fechas en un bucle de *punto fijo* que asocia posibles intervalos de fechas al recurso con la tarea.
  En cada iteración se obtiene un intervalo de fechas con la llamada "calcular_disponibilidad", que será comprobado posteriormente para comprobar que cumple los criterios de fragmentación y simultaneidad. Si esta comprobación fallara se modificaría el intervalo obtenido o se eliminaría si es necesario.
  La comprobación de estas condiciones puede implementarse en base a ciertos parámetros configurados en el programa o usando interacción con el usuario para que pueda decidir la asignación más conveniente a cada circunstancia.

  El bucle termina cuando se ha completado la asignación de la tarea y las comprobaciones posteriores son válidas.

```
funcion planificar_hacia_delante (ListaTareas todas_tareas)
< precondicion: hay al menos una tarea >

mientras ( [hay tareas sin planificar] ) hacer

    //
    // selección de tarea a planificar
    //

    ListaTareas camino_critico= Calcula_camino_critico( todas_tareas )
    ListaTareas planificadas= [tareas ya planificadas]
    ListaTareas no_criticas= ( todas_tareas – camino_critico ) – planificadas
    ListaTareas criticas= camino_critico – planificadas
    no_criticas= Elimina_no_dependen_planificada( no_criticas )
    criticas= Elimina_no_dependen_planificada_fwd( criticas )
    Tarea tarea
    si (criticas == [vacio])     tarea= Minimo_tarea( no_criticas )
    sino tarea= Minimo_tarea( criticas )
    fsi
```

```
//
// asignacion
//

tarea->fecha_inicio= Maximo_fecha_fin( Depende_hacia_atras( tarea ) )
< Recurso recurso,
  Fecha desde,
 Int por_asignar,
  ListaFechas fechas > disp_recursos[]

/** inicializar recursos[]:
    un registro para cada recurso que use la tarea, inicializando:
       recurso: familia_recurso que pide la tarea
       desde: fecha_inicio
       por_asignar: cantidad total de horas
       fechas: vacio
**/

// La asignacion de recurso es un bucle que va modificando la asignacion hasta que
// se cumplen las restricciones (como haria una ecuacion de punto fijo)

bool ajustado = falso
mientras ( no ajustado ) hacer

      para cada recurso_tratando en disp_recursos[] hacer

            < ListaFechas disponibilidad, Recurso recurso_asignado > =
                  Calcula_disponibilidad_fwd ( recurso_tratando->recurso,
                                               recurso_tratando->desde,
                                               recurso_tratando->por_asignar )
            recurso_tratando->recurso= recurso_asignado
            recurso_tratando->fechas= recurso_tratando->fechas ++ disponibilidad
            recurso_tratando->desde= Ultima_fecha ( recurso_tratando->fechas )

      fpara

   fmientras

fmientras

ffuncion
```

Tipos usados:

Tarea
ListaTareas
Fecha
ListaFechas
Recurso

Funciones auxiliares:

ListaTareas Elimina_no_dependen_planificada_fwd (*ListaTareas*)

ListaTareas Calcula_camino_critico (*ListaTareas*)

Tarea Minimo_tarea (*ListaTareas*)

ListaTareas Depende_hacia_atras (*Tarea)*

Fecha Maximo_fecha_fin (*ListaTareas)*

<*ListaFechas*, *Recurso*> Calcula_disponibilidad (*Recurso*, *Fecha, Int*)

ListaFechas operador ++ (*ListaFechas*, *ListaFechas*)

ListaFechas operador – (*ListaFechas*, *ListaFechas*)

Fecha Ultima_fecha (*ListaFechas*)

```
funcion <ListaFechas, Recurso> Calcula_disponibilidad_fwd ( Recurso recurso, Fecha desde, Int cantidad )

    /**
      aquí hay que ver si "recurso" es una familia o un recurso concreto si es una familia debemos elegir un recurso concreto, el mas optimo y hacer los calculos con el si ya es un recurso concreto calculamos las fechas directamente

      para elegir un recurso si solo nos especifican una familia:
        - para cada recurso "R" de la familia
          aplicamos Calcula_disponibilidad (R, .. , .. )
          elegimos el que de un rango de fechas mas optimo
          devolvemos directamente los resultados porque ya se han calculado
    **/

    si ( [tipo(recurso) es "familia de recursos"] )
        ListaFechas resul_lista[]
        Recurso resul_recurso[]
        para cada Recurso R en [recursos que pertencen a la familia 'recurso'] hacer
            ListaFechas lista
            Recurso rec
            <lista, rec> = Calcula_disponibilidad_fwd( R, desde, cantidad )
            si ( resul_lista==[vacio] || Ultima_fecha(lista) < Ultima_fecha(resul_lista) )
                resul_lista= lista
                resul_recurso= rec
            fsi
        fpara
        devuelve <resul_lista, resul_recurso>
    fsi

    //
    // Para sacar el rango de fechas en un recurso concreto:
    //

    Fecha fecha_busqueda = desde
    ListaFechas lista_fechas

    mientras ( cantidad > 0 )

        Int horas= Disponible( recurso, fecha_busqueda )
        si ( horas > 0 )
            Int asignado
            si ( horas > cantidad ) asignado= cantidad
            sino asignado= horas
            fsi
            cantidad= cantidad – asignado
            lista_fechas= lista_fechas + <fecha_busqueda, asignado>
        fsi
        fecha= fecha '+' 1;

    fmientras

    devuelve < lista_fechas, recurso >

ffuncion
```

Tipos usados:

Fecha
ListaFechas
Recurso

Funciones auxiliares:

Int Disponible (*Recurso*, *Fecha*)

Fecha Ultima_fecha (*ListaFechas*)

ListaFechas operador + *(ListaFechas*, *<Fecha*, *Int>*)

```
// Funcion recursiva:
funcion <ListaTareas, Int> rec_Camino_critico ( ListaTareas tareas, Tarea ini, Tarea fin)
    /** caso base **/
    si ( ini == fin )
        devuelve <vacio, 0>
    fsi
    ListaTareas dependientes= Depende_hacia_delante( tarea_ini )
    ListaTareas camino
    Int dias= -1
    para cada Tarea siguiente en dependientes hacer
        ListaTareas tmp_camino
        Int tmp_dias
        <tmp_camino, tmp_dias> = rec_Camino_critico( tareas, siguiente, fin )
        si ( tmp_dias == dias)
            camino= camino (+) tmp_camino  // union sin repeticion
        sino si (tmp_dias > dias
            camino= tmp_camino     // se descarta el anterior
            dias= tmp_dias
        sino si ( dias < 0 )   // primera asignacion
            camino= tmp_camino
            dias= tmp_dias
        fsi
    fpara
    camino= tarea_ini ++ camino
    dias= dias + Convierte_dias( Max( tarea_ini->recursos[]->cantidad ) )
    devuelve <camino, dias>
ffuncion


// Llamada inicial:
funcion ListaTareas Calcula_camino_critico ( ListaTareas tareas )
< suponemos que existe unas tareas especiales que representan el comienzo
 y el final del proyecto, que cierran las depencencias de todas las demas >
    Tarea ini= [tarea inicio del proyecto]
    Tarea fin= [tarea final del proyecto]
    ListaTareas camino
    Int dias
    <camino, dias> = rec_Camino_critico( tareas, ini, fin )
    devuelve camino
ffuncion
```

Tipos usados:

Tarea
ListaTareas

Funciones auxiliares:

ListaTareas Depende_hacia_delante (Tarea)
ListaFechas operador (+) (ListaFechas, ListaFechas)
Int Convierte_dias (Int)

Planificación hacia atrás.

El proceso de “planificar hacia atrás” se realiza de manera análoga a la planificación hacia delante, únicamente varía en el la interpretación del recorrido temporal del algoritmo y funciones auxiliares, convenientemente modificadas.

Se invierte el sentido del recorrido, y usando el mismo criterio acerca de las tareas que pertenecen al camino crítico, se seleccionará la tarea cuya fecha de finalización sea mayor.

Una de las restricciones del proceso de selección cambiara ahora a que una tarea se podrá planificar sólo si ya han sido planificadas todas las tareas que dependen de ella por delante.

```
funcion planificar_hacia_atras (ListaTareas todas_tareas)
< precondicion: hay al menos una tarea >

mientras ( [hay tareas sin planificar] ) hacer

    //
    // selección de tarea a planificar
    //

    ListaTareas camino_critico= Calcula_camino_critico( todas_tareas )
    ListaTareas planificadas= [tareas ya planificadas]
    ListaTareas no_criticas= ( todas_tareas – camino_critico ) – planificadas
    ListaTareas criticas= camino_critico – planificadas
    no_criticas= Elimina_no_dependen_planificada( no_criticas )
    criticas= Elimina_no_dependen_planificada_bck( criticas )

    Tarea tarea
    si (criticas == [vacio])      tarea= Maximo_tarea( no_criticas )
    sino                tarea= Maximo_tarea( criticas )
    fsi

    //
    // asignacion
    //

    tarea->fecha_fin= Minimo_fecha_inicio( Depende_hacia_delante( tarea ) )
    < Recurso recurso,
      Fecha desde,
     Int por_asignar,
      ListaFechas fechas > disp_recursos[]

    /** inicializar recursos[]:
        un registro para cada recurso que use la tarea, inicializando:
            recurso: familia_recurso que pide la tarea
            desde: fecha_inicio
            por_asignar: cantidad total de horas
            fechas: vacio
    **/

    // La asignacion de recurso es un bucle que va modificando la asignacion hasta que
    // se cumplen las restricciones (como haria una ecuacion de punto fijo)
```

```
    bool ajustado = falso
    mientras ( no ajustado ) hacer

        para cada recurso_tratando en disp_recursos[] hacer

            < ListaFechas disponibilidad, Recurso recurso_asignado > =
                Calcula_disponibilidad_bck ( recurso_tratando->recurso,
                                             recurso_tratando->desde,
                                              recurso_tratando->por_asignar )
            recurso_tratando->recurso= recurso_asignado
            recurso_tratando->fechas= disponibilidad ++ recurso_tratando->fechas
            recurso_tratando->desde= Primera_fecha ( recurso_tratando->fechas )

        fpara

        /**
           hacer comprobaciones de fragmentacion y simultaneidad en los intervalos de
fechas
           asignados a los recursos
           modifica variable "ajustado" cuando sea correcto

           * hay que demostrar que el bucle finaliza *
        **/

    fmientras

fmientras

ffuncion
```

Tipos usados:

Tarea
ListaTareas
Fecha
ListaFechas
Recurso

Funciones auxiliares:

ListaTareas Elimina_no_dependen_planificada_bck (*ListaTareas*)

ListaTareas Calcula_camino_critico (*ListaTareas*)

Tarea Maximo_tarea (*ListaTareas*)

ListaTareas Depende_hacia_delante (*Tarea)*

Fecha Minimo_fecha_inicio (*ListaTareas)*

<*ListaFechas*, *Recurso*> Calcula_disponibilidad (*Recurso*, *Fecha, Int*)

ListaFechas operador ++ (*ListaFechas*, *ListaFechas*)

ListaFechas operador – (*ListaFechas*, *ListaFechas*)

Fecha Primera_fecha (*ListaFechas*)

funcion <*ListaFechas*, *Recurso*> **Calcula_disponibilidad_bck** (*Recurso* recurso*, Fecha* desde, *Int* cantidad)

```
    /**
      aquí hay que ver si "recurso" es una familia o un recurso concreto
      si es una familia debemos elegir un recurso concreto, el mas optimo y hacer los
calculos con el
      si ya es un recurso concreto calculamos las fechas directamente

      para elegir un recurso si solo nos especifican una familia:
        - para cada recurso "R" de la familia
```

```
        aplicamos Calcula_disponibilidad (R, .. , .. )
        elegimos el que de un rango de fechas mas optimo
        devolvemos directamente los resultados porque ya se han calculado
    **/

    si ( [tipo(recurso) es "familia de recursos"] )
        ListaFechas resul_lista[]
        Recurso resul_recurso[]
        para cada Recurso R en [recursos que pertencen a la familia 'recurso'] hacer
            ListaFechas lista
            Recurso rec
            <lista, rec> = Calcula_disponibilidad_bck( R, desde, cantidad )
            si ( resul_lista==[vacio] || Primera_fecha(lista) > Primera_fecha(resul_lista) )
                resul_lista= lista
                resul_recurso= rec
            fsi
        fpara
        devuelve <resul_lista, resul_recurso>
    fsi

    //
    // Para sacar el rango de fechas en un recurso concreto:
    //

    Fecha fecha_busqueda = desde
    ListaFechas lista_fechas

    mientras ( cantidad > 0 )

        Int horas= Disponible( recurso, fecha_busqueda )
        si ( horas > 0 )
            Int asignado
            si ( horas > cantidad ) asignado= cantidad
            sino asignado= horas
            fsi
            cantidad= cantidad – asignado
            lista_fechas= lista_fechas + <fecha_busqueda, asignado>
        fsi
        fecha= fecha '-' 1;

    fmientras

    devuelve < lista_fechas, recurso >

ffuncion
```

Tipos usados:

Fecha
ListaFechas
Recurso

Funciones auxiliares:

Int Disponible (*Recurso*, *Fecha*)

Fecha Ultima_fecha (*ListaFechas*)

ListaFechas operador + *(ListaFechas*, <*Fecha*, *Int*>)

Planificación por coste.

El proceso de “planificar con coste mínimo” es en parte similar a la planificación hacia delante. Sigue la misma estrategia de seleccionar las tareas una a una hasta que se han planificado todas. En particular, se modifican algunos conceptos:

- *Camino de coste crítico*: Análogamente al camino crítico, en el contexto de duración de las tareas, podemos interpretar un *camino de coste crítico* como aquel cuyas tareas presentan la mayor suma de costes de todos los caminos. Este camino es, a priori, el más decisivo en el coste total del proyecto.

Una vez seleccionada una tarea para calcular su planificación, se hacen llamadas a la función auxiliar “Calcula_disponibilidad_coste” para cada recurso, que devuelve los intervalos de fechas que se pueden asignar al recurso.

De la misma forma que en las otras estrategias, esa asignación de fechas y recursos se repite hasta que las fechas resultantes cumplan las condiciones de *fragmentación* y *simultaneidad*.

```
funcion planificar_coste_minimo (ListaTareas todas_tareas)
< precondicion: hay al menos una tarea >

mientras ( [hay tareas sin planificar] ) hacer

    //
    // selección de tarea a planificar
    //

    ListaTareas coste_critico= Calcula_camino_critico_coste( todas_tareas )
    ListaTareas planificadas= [tareas ya planificadas]
    ListaTareas no_criticas= ( todas_tareas – coste_critico ) – planificadas
    ListaTareas criticas= coste_critico – planificadas
    no_criticas= Elimina_no_dependen_planificada( no_criticas )
    criticas= Elimina_no_dependen_planificada( criticas )

    Tarea tarea
    si (criticas == [vacio])     tarea= Minimo_tarea( no_criticas )
    sino                         tarea= Minimo_tarea( criticas )
    fsi

    //
    // asignacion
    //

    tarea->fecha_inicio= Maximo_fecha_fin( Depende_hacia_atras( tarea ) )
    < Recurso recurso,
      Fecha desde,
     Int por_asignar,
      ListaFechas fechas > disp_recursos[]

    /** inicializar recursos[]:
        un registro para cada recurso que use la tarea, inicializando:
            recurso: familia_recurso que pide la tarea
```

```
        desde: fecha_inicio
        por_asignar: cantidad total de horas
        fechas: vacio
    **/

    // La asignacion de recurso es un bucle que va modificando la asignacion hasta que
    // se cumplen las restricciones (como haria una ecuacion de punto fijo)

    bool ajustado = falso
    mientras ( no ajustado ) hacer

        para cada recurso_tratando en disp_recursos[] hacer

            < ListaFechas disponibilidad, Recurso recurso_asignado > =
                Calcula_disponibilidad_coste ( recurso_tratando->recurso,
                                               recurso_tratando->desde,
                                               recurso_tratando->por_asignar )
            recurso_tratando->recurso= recurso_asignado
            recurso_tratando->fechas= recurso_tratando->fechas ++ disponibilidad
            recurso_tratando->desde= Ultima_fecha ( recurso_tratando->fechas )

        fpara

    fmientras

fmientras

ffuncion
```

Tipos usados:

Tarea
ListaTareas
Fecha
ListaFechas
Recurso

Funciones auxiliares:

ListaTareas Elimina_no_dependen_planificada (*ListaTareas*)

ListaTareas Calcula_camino_critico_coste (*ListaTareas*)

Tarea Minimo_tarea (*ListaTareas*)

ListaTareas Depende_hacia_atras (*Tarea)*

Fecha Maximo_fecha_fin (*ListaTareas)*

<*ListaFechas*, *Recurso*> Calcula_disponibilidad_coste (*Recurso*, *Fecha, Int*)

ListaFechas operador ++ (*ListaFechas*, *ListaFechas*)

ListaFechas operador – (*ListaFechas*, *ListaFechas*)

Fecha Ultima_fecha (*ListaFechas*)

funcion <*ListaFechas*, *Recurso*> **Calcula_disponibilidad_coste** (*Recurso* recurso*, Fecha* desde, *Int* cantidad)

```
    /**
      aquí hay que ver si "recurso" es una familia o un recurso concreto
      si es una familia debemos elegir un recurso concreto, el mas optimo y hacer los
calculos con el.
      si ya es un recurso concreto calculamos las fechas directamente

      para elegir un recurso si solo nos especifican una familia:
        - para cada recurso "R" de la familia
          aplicamos Calcula_disponibilidad_coste (R, .. , .. )
          elegimos el que de un rengo de fechas con coste mas optimo
```

```
        devolvemos directamente los resultados porque ya se han calculado
    **/

    si ( [tipo(recurso) es "familia de recursos"] )
        ListaFechas resul_lista[]
        Recurso resul_recurso[]
        para cada Recurso R en [recursos que pertencen a la familia 'recurso'] hacer
            ListaFechas lista
            Recurso rec
            <lista, rec> = Calcula_disponibilidad_coste( R, desde, cantidad )
            si ( resul_lista==[vacio] ||
                 Calcula_coste_fechas(rec, lista) < Calcula_coste_fechas(resul_recurso,
resul_lista) )
                 resul_lista= lista
                 resul_recurso= rec
            fsi
        fpara
        devuelve <resul_lista, resul_recurso>
    fsi

    //
    // Para sacar el rango de fechas en un recurso concreto:
    //

    Fecha limite_fin= desde + [numero maximo de dias que se permite alargar la
finalizacion de la tarea]

    ListaFechas lista_fechas= Intervalo_minimo_coste( recurso, desde, cantidad, limite_fin
)

    devuelve < lista_fechas, recurso >

ffuncion
```

Tipos usados:

Fecha
ListaFechas
Recurso

Funciones auxiliares:

Int Disponible (*Recurso*, *Fecha*)

Decimal Calcula_coste_fechas(*Recurso*, *ListaFechas*)

ListaFechas operador + *(ListaFechas*, <*Fecha*, *Int*>)

Fecha operador + (*Fecha*, *Int*)

ListaFechas Intervalo_minimo_coste(*Recurso*, *Fecha*, *Int*, *Fecha*)

```
funcion ListaFechas Intervalo_minimo_coste( Recurso recurso, Fecha desde, Int
cantidad, Fecha fin)

    ListaFechas disponibles= fechas_disponible( recurso, desde, fin )
    disponibles= Ordena_minino_coste( disponibles, recurso )

    ListaFechas resul_fechas
    Int pos= 0

    mientras (cantidad > 0) hacer
        si ( pos >= Tamaño(disponibles) ) ERROR "el recurso no tiene horas disponibles
en el intervalo [desde,fin]"
        Int horas= disponibles[pos]->horas_disponibles
        si (cantidad < horas) horas= cantidad
        resul_fechas= resul_fechas + <disponible[pos], horas>
```

```
        cantidad= cantidad – horas
        pos= pos + 1
    fmientras

    devuelve resul_fechas

ffuncion
```

Tipos usados:

Fecha
ListaFechas
Recurso

Funciones auxiliares:

ListaFechas Fechas_disponible (*Recurso*, *Fecha*, *Fecha*)

ListaFechas Ordena_minino_coste (*ListaFechas*, *Recurso*)

ListaFechas operador + *(ListaFechas*, <*Fecha*, *Int*>)

// Funcion recursiva:

funcion <*ListaTareas, Decimal*> **rec_Camino_critico_coste** (*ListaTareas* tareas, *Tare*a ini, *Tarea* fin)

```
    /** caso base **/

    si ( ini == fin )
        devuelve <vacio, 0>
    fsi

    ListaTareas dependientes= Depende_hacia_delante( ini )
    ListaTareas camino
    Decimal coste= -1

    para cada Tarea siguiente en dependientes hacer
        ListaTareas tmp_camino
        Decimal tmp_coste
        <tmp_camino, tmp_coste> = rec_Camino_critico( tareas, siguiente, fin )
        si ( tmp_coste == coste)
            camino= camino (+) tmp_camino  // union sin repeticion
        sino si (tmp_coste > coste
            camino= tmp_camino     // se descarta el anterior
            coste= tmp_coste
        sino si ( dias < 0 )    // primera asignacion
            camino= tmp_camino
            coste= tmp_coste
        fsi
    fpara

    camino= tarea_ini ++ camino
    dias= dias + Convierte_dias( Max( tarea_ini->recursos[]->cantidad ) )

    devuelve <camino, dias>

ffuncion
```

// Llamada inicial:

funcion *ListaTareas* **Calcula_camino_critico_coste** (*ListaTareas tareas*)

< suponemos que existe unas tareas especiales que representan el comienzo
y el final del proyecto, que cierran las depencencias de todas las demas >

```
    Tarea ini= [tarea inicio del proyecto]
    ListaTareas camino
```

Decimal coste

<camino, coste> = rec_Camino_critico_coste(tareas, ini)

devuelve camino

ffuncion

Tipos usados:

Tarea
ListaTareas

Funciones auxiliares:

ListaTareas Depende_hacia_delante (*Tarea*)

ListaFechas operador (+) (*ListaFechas*, *ListaFechas*)

Int Convierte_dias (*Int*)

Otras funciones.

ListaFechas operador **++** (*ListaFechas*, *ListaFechas*)

LR = L1 ++ L2

Devuelve LR como la concatenacionde L1 y L2

ListaFechas operador **(+)** (*ListaFechas*, *ListaFechas*)

LR = L1 (+) L2

Devuelve LR que contiene la union de los elementos de L1 y L2, sin incluir las repeticiones

ListaFechas operador **+** *(ListaFechas*, <*Fecha*, *Int*>)

LR = L + E

Devuelve LR que es la lista L añadiendole el elemento E

ListaFechas operador **–** (*ListaFechas*, *ListaFechas*)

LR = L1 – L2

Devuelve LR, resultado de eliminar de L1 las apariciones de los elementos contenidos en L2

ListaTareas **Depende_hacia_delante** (*Tarea*)

LR = Depende_hacia_delante (T)

LR es una lista de todas las tareas que tienen dependecias directas por delante con T

ListaTareas **Depende_hacia_atras** (*Tarea)*

LR = Depende_hacia_atras (T)

LR es una lista de todas las tareas que tienen dependecias directas por detras con T

ListaTareas **Elimina_no_dependen_planificada_fwd** (*ListaTareas*)

LR = Elimina_no_denden_planificada(LT)

LR es la lista de tareas resultante de eliminar de LT aquellas tareas cuyas dependencias hacia atrás presenta alguna tarea que no está planificada.

ListaTareas **Elimina_no_dependen_planificada_bck** (*ListaTareas*)

LR = Elimina_no_denden_planificada(LT)

LR es la lista de tareas resultante de eliminar de LT aquellas tareas cuyas dependencias hacia delante presenta alguna tarea que no está planificada.

Fecha operador **+** (*Fecha*, *Int*)

FR = FI + ND

FR es la fecha resultante de añadir ND dias a la fecha inicial FI

Fecha **Minimo_fecha_inicio** (*ListaTareas)*

F = Minimo_fecha_inicio (LT)

Devuelve la menor fecha de inicio de las tareas contenidas en LT

Fecha **Maximo_fecha_fin** (*ListaTareas)*

F = Maximo_fecha_fin (LT)

Devuelve la mayo fecha de finalizacion de las tareas contenidas en LT

Tarea **Maximo_tarea** (*ListaTareas*)

T = Maximo_tarea (LT)

Devuelve la tarea T contenida en LT que tiene fecha de inicio más tardía

Tarea **Minimo_tarea** (*ListaTareas*)

T = Minimo_tarea (LT)

Devuelve la tarea T contenida en LT que tiene fecha de inicio más temprana

Fecha **Ultima_fecha** (*ListaFechas*)

F = Ultima_fecha (LF)

Devuelve la fecha más tardia de las contenidas en LF

Fecha **Primera_fecha** (*ListaFechas*)

F = Primera_fecha (LF)

Devuelve la fecha más temprana de las contenidas en LF

Int **Disponible** (*Recurso*, *Fecha*)

NH = Disponible (R,F)

Devuelve el número de horas que el recurso R tiene disponibles en la fecha F

Decimal **Calcula_coste_fechas** (*Recurso*, *ListaFechas*)

C = Calcula_coste_fechas (R,LF)

Devuelve el coste de utilizar el recurso R en las fechas y horas indicadas en LF

ListaFechas **Fechas_disponible** (*Recurso*, *Fecha*, *Fecha*)

LR = Fechas_disponible (FI,FF)

Devuelve una lista de fechas en que el recurso esta disponible, entre las fechas inicial y final indicadas por FI y FF respectivamente

ListaFechas **Ordena_minino_coste** (*ListaFechas*, *Recurso*)

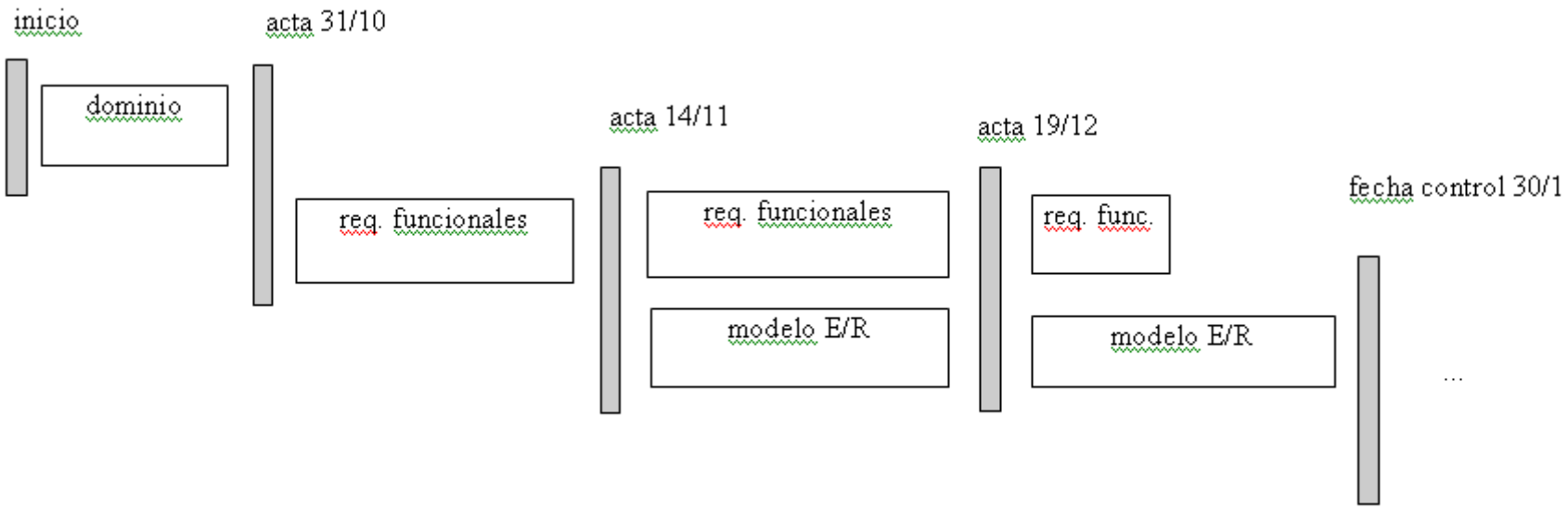
LR = Ordena_minimo_coste (LF,R)

Devuelve LR que es la lista de fechas LF ordenada en función del coste que tiene el recurso R en cada fecha, de menor a mayor coste

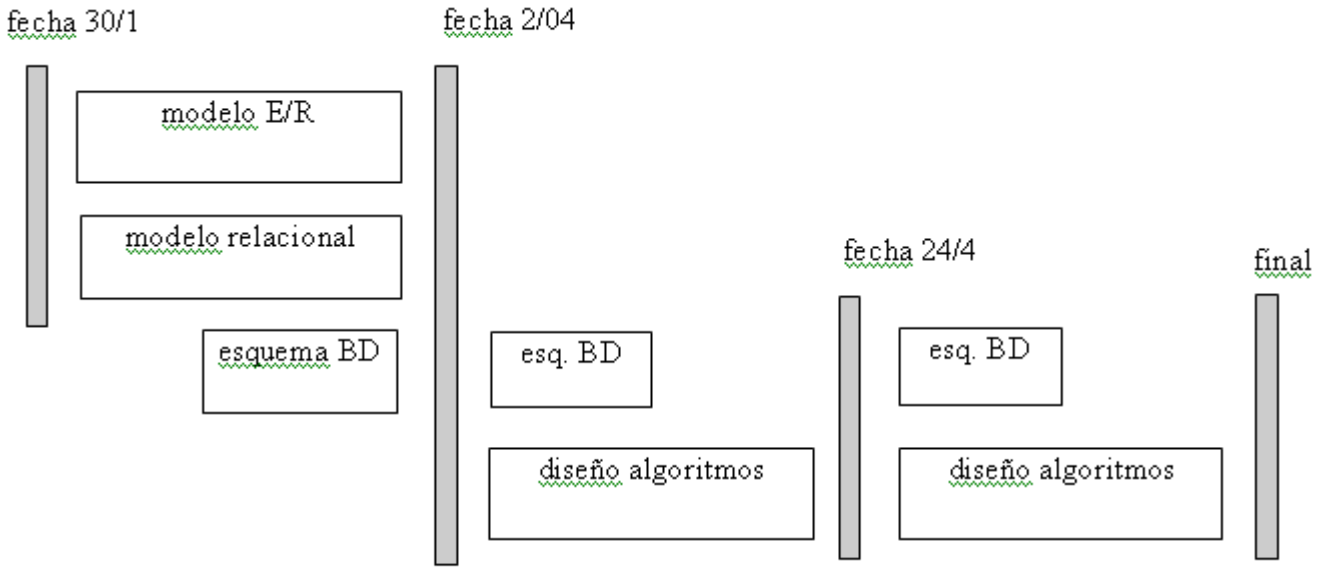
Int **Convierte_dias** (*Int*)

ND = Convierte_dias (NH)

Devuelve el número de dias equivalentes al número de horas NH



La primera etapa del trabajo se centró en la captura de requisitos de la herramienta, así como en el diseño de la base de datos que da sustento a la misma (modelo entidad-relación).



A partir de aquí, se refina el modelo relacional, se implementa la base de datos y entramos en la fase de diseño de los diferentes algoritmos, así como su implementación en pseudocódigo.

11. Relación de objetivos que perseguimos con nuestro proyecto.

Objetivos cumplidos.

Nuestro proyecto se ha desarrollado casi en su totalidad de manera teórica. Inicialmente tomamos las especificaciones guiados por el profesor, que actuaba como un cliente real. Las actas de reunión dejaron constancia de esos datos, que después se han convertido en el diseño de nuestro planificador de proyecto.

El primer objetivo cumplido fue crear el modelado, diseño y posterior implementación de la base de datos que sostiene toda la información para almacenar datos de proyectos de manera genérica.

Posteriormente se crearon los algoritmos de planificación de los recursos del proyecto. Hemos realizado tres maneras de planificación:

-Hacia Delante: Asignación de recursos a unas tareas concretas partiendo de una fecha de inicio de un proyecto, dependiendo de la disponibilidad del recurso en el menor tiempo posible. El cliente que utilice este planificador suele exigir que la planificación hacia delante se realice en el menor tiempo posible.

-Hacia Atrás: Asignación de recursos a unas tareas de un proyecto partiendo de una fecha de finalización. El cliente marca la fecha de fin de su proyecto y esta planificación tiene que hacer cumplir que el proyecto no supere esa duración.

-Costes: Los recursos tienen un coste asociado por hora. Esta planificación se encargará de la asignación de recursos a las tareas de un proyecto con el mínimo coste.

Hemos dado soporte para que un cliente tenga varios proyectos contratados, e incluso se le puedan realizar ofertas de un mismo proyecto con diferentes planificaciones y, por supuesto, varios presupuestos.

Los recursos se han separado en familias (familias de recurso y familias de productos), dando, de esta manera, capacidades de elección de unos recursos u otros en las planificaciones, sobre todo en la planificación por costes (búsqueda del recurso más barato).

Finalmente damos un soporte en la base de datos para que, finalizado un proyecto, se pueda guardar la información referente a un proyecto concreto. Es lo que denominamos histórico de proyectos.

Objetivos futuros.

Un aspecto que se recoge en las especificaciones funcionales de nuestro proyecto y que no se ha tratado es el seguimiento de nuestro proyecto por medio de las estadísticas de cada uno en particular. Estadísticas referentes al tiempo empleado en las fases, a sus costes…dependiendo de la planificación que se haya escogido. Esto nos permitirá observar la evolución de nuestro proyecto y compararlo con otros proyectos similares o el mismo proyecto pero con diferentes planificaciones, para ofrecer al cliente una visión más exacta e intuitiva de cómo planificar su proyecto.

Como trabajos adicionales se podría estudiar la manera de optimizar los algoritmos de planificación, si es posible, para su posterior implementación en un lenguaje de programación SQL.

Resolviendo todas estas características pendientes, el paso a una interfaz gráfica que interactúe con la base de datos se realizaría con más facilidad, dando a esta idea inicial de proyecto aspecto de aplicación profesional.

Bibliografía

SILBERSCHATZ, A., KORTH, H.F., SUDARSHAN, S. ; *Database System Concepts (Fundamentos de bases de datos);* 5ª edición, McGraw-Hill, 2006;

ELMASRI, R., NAVATHE, S.B. ; *Fundamentals of Database Systems(Fundamentos de sistemas de bases de datos);* 4ª edición, Addison-Wesley, 2004;

ULLMAN, J.D. ; *Principles of Databases and Knowledge Base Systems;* Computer Science Press, 1998;

ABBEY, M., COREY, M.J.; *ORACLE8 : guía de aprendizaje;* Osborne/McGraw-Hill, 1998;

MySQL 5.0 Reference Manual
dev.mysql.com/doc/refman/5.0/es/tutorial.html

MySQL Hispano - La comunidad de usuarios de MySQL
www.mysql-hispano.org/

Los autores del presente proyecto, Santiago Martín López con DNI-50749384-E, Carlos Proensa Mora con DNI-47493108-V y Pedro Rizaldos Pareja con DNI-00838691-L autorizamos a la Universidad Complutense a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a sus autores, tanto la propia memoria, como el código, la documentación y/o el prototipo desarrollado.

Firmado,

Carlos Proensa

Santiago Martín Pedro Rizaldos