

GPI - Sistemas de Información

Gestor de Proyectos con Interfaz

Navarro Alfonso, Paula
Soriano Pérez, Carles Salvador

Tabla de contenido

1.	INTRODUCCIÓN.....	6
1.1	Objetivo.....	6
1.2	Ámbito.....	6
1.3	Bibliografía	6
2	Características Aplicación.....	7
2.1	Requerimientos del sistema	7
2.2	Ventajas frente a otros programas.....	7
3	Interfaz de Usuario	8
3.1	Entorno de desarrollo.....	8
3.1.1	Sobre QT	8
3.2	Mapa Conceptual Interfaz	11
3.3	Funcionamiento Interfaz	11
3.4	Casos de uso	12
3.4.1	Vista Ventana Principal.....	12
3.4.2	Añadir Actividad.	13
3.4.3	Añadir Recurso.	14
3.4.4	Añadir Festivos y Laborables.	15
3.4.5	Generar Proyecto.	16
3.4.6	Informe.....	17
3.4.7	Generar Análisis.....	20
3.5	Diagrama de clases.....	21
4	Desarrollo.....	22
4.1	Entorno de Desarrollo	22
4.2	Estructuras de Datos utilizadas.....	23
4.3	Recurso.....	23
4.3.1	Actividad.....	24
4.3.2	Camino	27
4.3.3	Proyecto	28
4.3.4	Otros.....	31
4.4	Desarrollo de Algoritmos.....	32
4.4.1	Camino Crítico	32
4.4.2	Mínimo coste mínima duración	37
4.4.3	Limitación de recursos.....	52

4.4.4	Atraso/Adelanto Asignación de recursos.....	65
4.4.5	Nivelación de recursos.....	76
4.5	ANALISIS	81
4.5.1	Flexibilidad.....	81
4.5.2	Probabilidad	85
4.5.3	LIBRERÍA DISTRIBUCION NORMAL.....	86
4.5.4	Librería de reglas	88
5	Ejemplos.....	91
5.1	Ejemplo 1.....	91
5.1.1	Camino Crítico	92
5.1.2	Limitación de recursos con regla LFT y esquema serie	94
5.1.3	Retraso adelanto	98
5.1.4	Nivelación recursos.....	103
5.1.5	Análisis.....	106
5.2	Ejemplo 2.....	107
5.2.1	Camino critico.....	109
5.2.2	Minima duración minimo coste	113
5.2.3	Limitación con esquema paralelo y regla NSUC.....	122
5.2.4	Retraso con esquema paralelo y adelanto con esquema serie	130
5.2.5	Nivelación de todos los recursos	147
5.2.6	Análisis.....	158
6	Conclusiones Finales	159
7	Anexos.....	160
7.1	Código Interfaz	161
7.1.1	ActivityView.h.....	161
7.1.2	ActivityView.cpp	162
7.1.3	Calendar.h	166
7.1.4	Calendar.cpp.....	167
7.1.5	GenerateAnalysis.h.....	169
7.1.6	GenerateAnalysis.cpp	170
7.1.7	GenerateReport.h.....	172
7.1.8	GenerateReport.cpp	173
7.1.9	InsertDate.h.....	177
7.1.10	InsertDate.cpp	178

7.1.11	Report.h.....	179
7.1.12	Report.cpp.....	180
7.1.13	ResourcesView.h	183
7.1.14	ResourcesView.cpp.....	183
7.1.15	VentanaPrincipal.h	186
7.1.16	VentanaPrincipal.cpp.....	187
7.2	Código Completo Algoritmos.....	193
7.2.1	Activity.h.....	193
7.2.2	Activity.cc	194
7.2.3	Exceptions.h	197
7.2.4	Exceptions.cc	197
7.2.5	Path.h	198
7.2.6	Path.cc	198
7.2.7	Probability.h	199
7.2.8	Project.h	200
7.2.9	Project.cc	201
7.2.10	NivelationResouces.cc	208
7.2.11	Limitación de Recursos	209
7.2.12	MinCostMinDuratio.cc.....	214
7.2.13	Rules.h	216
7.2.14	Rules.cc.....	217
7.2.15	Resource.h.....	218
7.2.16	Resource.cc.....	218
7.3	Manual del usuario.....	219
7.3.1	Ventana Principal.....	220
7.3.2	Añadir Actividad	222
7.3.3	Gestionar Recursos.....	223
7.3.4	Gestionar Fechas	224
7.3.5	Generar Informe	225
7.3.6	Informe	225
7.3.7	Análisis.....	226
7.3.8	Atajos de teclado	226

“La satisfacción es la única señal de la sinceridad del placer.” - Andre Gidé

1. INTRODUCCIÓN

1.1 *Objetivo*

El proyecto ‘Gestor de Proyectos’ pretende desarrollar una aplicación con interfaz gráfica capaz de administrar las actividades, recursos y tiempo de trabajo de cualquier proyecto en el ámbito empresarial . Esta herramienta pretende ser un substitutivo de otras herramientas como CA SuperProject, las cuales son demasiado complicadas tanto de utilizar para introducir un proyecto como de leer a la hora de analizar los informes. De esta forma nuestra aplicación es una herramienta directa y de fácil uso para el usuario.

1.2 *Ámbito*

Este documento tiene el propósito de definir y hacer entender al lector como hemos dividido el desarrollo del ‘Gestor de Proyectos’. Diferenciándolo, en dos partes, el entorno gráfico y el entorno de los algoritmos. Su desarrollo se ha llevado de forma independiente precisamente para tener una interfaz que se pueda modificar a gusto del usuario fácilmente y por otro lado una base algorítmica que permita añadir o revisar cualquier algoritmo de este programa.

Por un lado, la sección dedicada a la interfaz mostrará que motivos nos han llevado a estructurar la aplicación como se encuentra actualmente (casos de uso), y cuál es el funcionamiento de esta.

Finalmente tenemos el desarrollo del programa, el cual tiene dos secciones, una encargada de explicar cómo hemos aplicado los algoritmos teóricos a nuestro entorno de programación, y otra en la que se podrá ver las estructuras de datos que hemos utilizado.

De forma adicional hemos querido incluir nuestra valoración personal junto a varios anexos con los códigos completos del programa.

1.3 *Bibliografía*

- Referencias al código de QT <http://doc.qt.digia.com>
- QT Plotting widget: <http://www.workslikeclockwork.com/index.php/components/qt-plotting-widget>
- Teoría y Prácticas Gestión de Proyectos Informáticos.
- <http://www.richelbilderbeek.nl/CppGetCumulativeDensityNormal.htm>

2 Características Aplicación

La aplicación que hemos desarrollado permite al usuario obtener los mismos resultados que los obtenidos a partir de otras herramientas de Gestión de proyectos como SuperProject, con la ventaja de que nuestro Software es mucho más fácil y sencillo de utilizar. La interfaz de nuestra aplicación es intuitiva y permite guardar y cargar proyectos sin problema alguno. Esta aplicación pretende abarcar todo el temario de Gestión de Proyectos Informáticos visto en clase, y está enfocada a que el usuario obtenga de una forma más directa los resultados esperados. Destacamos ahora sus funcionalidades:

- Añadir /modificar / eliminar actividades
- Añadir /eliminar Recursos
- Gestionar fechas
- Asignar recursos.
- Asignar Predecesores.
- Informes basados en :
 - Camino Crítico
 - Mínimo Coste mínima duración
 - Limitación de recursos
 - Atraso/Adelanto
 - Nivelación de recursos
- Vista del calendario con días de trabajo del proyecto.
- Diagrama de Gantt en el informe.
- Histograma asignación de Recursos en el informe.
- Análisis de flexibilidad sobre las Actividades.
- Análisis de probabilidad sobre el proyecto.
- Guardar y Cargar proyectos con el formato exclusivo *.gpi

2.1 Requerimientos del sistema

Esta aplicación final ha sido desarrollada para utilizar bajo la plataforma Windows. Además se puede ejecutar desde un CD o un USB gracias a que han sido utilizadas librerías estáticas para evitar que el usuario final requiera instalación alguna.

2.2 Ventajas frente a otros programas.

- Software portable, no requiere instalación.
- Informes completos paso por paso.
- Licencia libre.
- Gestión rápida y sencilla de fechas laborales.
- Facilidad de uso para cualquier tipo de usuario.

3 Interfaz de Usuario

3.1 Entorno de desarrollo

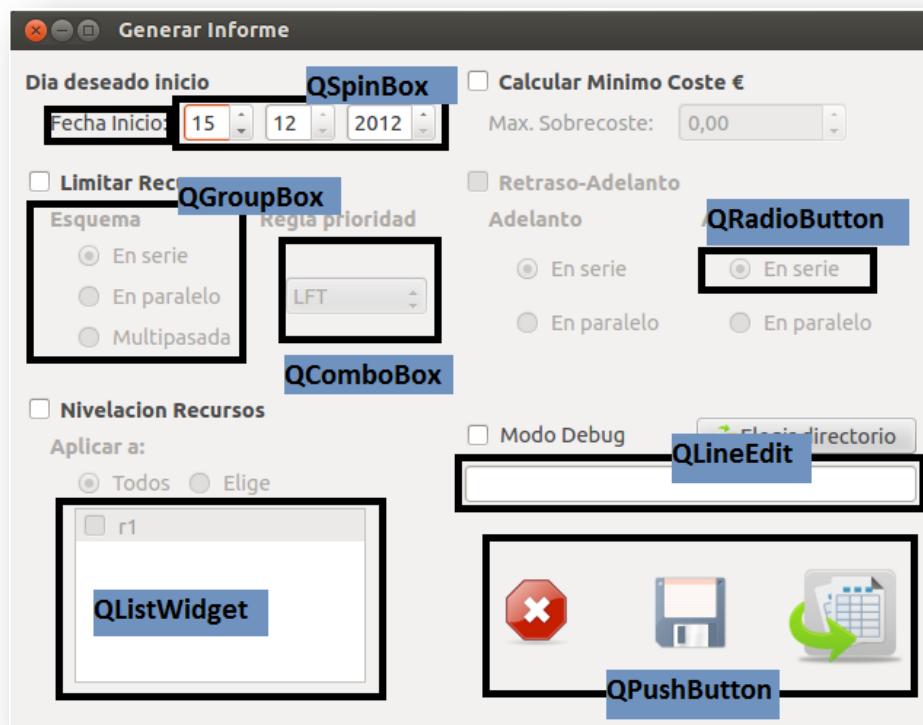
Para generar nuestra interfaz de usuario hemos hecho uso del entorno de desarrollo QT Creator 2.4.1, basado en QT 4.8 propiedad de Nokia Corporation. Este entorno nos permite gestionar los ficheros de programación en C++ y poder generar un entorno gráfico de forma fácil y sencilla gracias a sus librerías. De forma adicional, para poder generar el histograma de asignación de recursos hemos hecho uso de los ficheros qcustomplot.cpp y qcustomplot.h, que nos permiten mostrar gráficas gracias al aporte de la comunidad de desarrollo QT.

3.1.1 Sobre QT

QT nos permite crear ventanas mediante el uso de clases que se mostrarán como distintos objetos en la interfaz final. Gracias a la programación en C++ podemos enlazar y crear acciones entre estos objetos, de forma que al pulsar un botón se accionen un conjunto de tareas en segundo plano. Esto lo haremos gracias al comando connect, uno de los pilares fundamentales de este programa, el cual se sirve de un objeto, una tipo de acción, y una función privada donde realizar las tareas que corresponden a esta acción.

```
connect(submit, SIGNAL(clicked()), this, SLOT(generate()));
```

A continuación enumeramos un poco las distintas clases de objetos que QT proporciona con el ejemplo de la siguiente ventana:



- **QLabel:** añade una etiqueta.

```
date_iniLa = new QLabel(tr("Fecha Inicio: "));
```

- **QSpinBox / QDoubleSpinBox:** añade un cuadro de texto numérico.

```
date_dd = new QSpinBox;
date_dd->setMaximum(31);date_dd->setMinimum(1);
date_dd->setMaximumWidth(50);
```

- **QRadioButton:** añade un botón exclusivo de selección.

```
ser = new QRadioButton(tr("En serie"));
```

- **QGroupBox:** permite agrupar distintos objetos en un mismo campo mediante QLayouts, de esta forma por ejemplo, agrupamos los QRadioButton.

```
QVBoxLayout *vbox1 = new QVBoxLayout;
vbox1->addWidget(ser);
vbox1->addWidget(par);
vbox1->addWidget(multi);
```

- **QCheckBox:** añade un cuadro seleccionable para habilitar opciones. También se puede habilitar esta opción para que un QGroupBox pase a ser, también, un QCheckBox.

```
NivRec = new QGroupBox(tr("Nivelacion Recursos"));
NivRec->setCheckable(true);
NivRec->setChecked(false);
```

- **QLineEdit:** crea un cuadro del texto en el cual podemos habilitar o no la escritura por parte del usuario.

```
linePath = new QLineEdit();
linePath->setReadOnly(true);
```

- **QListWidget:** crea una lista de los objetos que queramos .

```
resList = new QListWidget();
resList->setEnabled(false);
vector<Resource*> *vecRes = pro->getResources();
int nvecRes = vecRes->size();
for(int i=0;i<nvecRes;i++){
    string name = vecRes->at(i)->getName();
    QListWidgetItem *checkbox1 = new QListWidgetItem(QString::fromStdString(name));
    resList->addItem(checkbox1);
    resList->item(i)->setFlags(resList->item(i)->flags() |Qt::ItemIsUserCheckable);
    resList->item(i)->setCheckState(Qt::Unchecked);
}
```

- **QTableWidget:** crea una tabla de una forma similar a las listas.

- **QPushButton:** crea un botón en el que se permite hacer click.

```
submit = new QPushButton();
submit->setFixedSize(100,108); submit->setFlat(true);
submit->setIcon(QPixmap("./icons/generate.png")); submit->setIconSize(QSize(80,80));
```

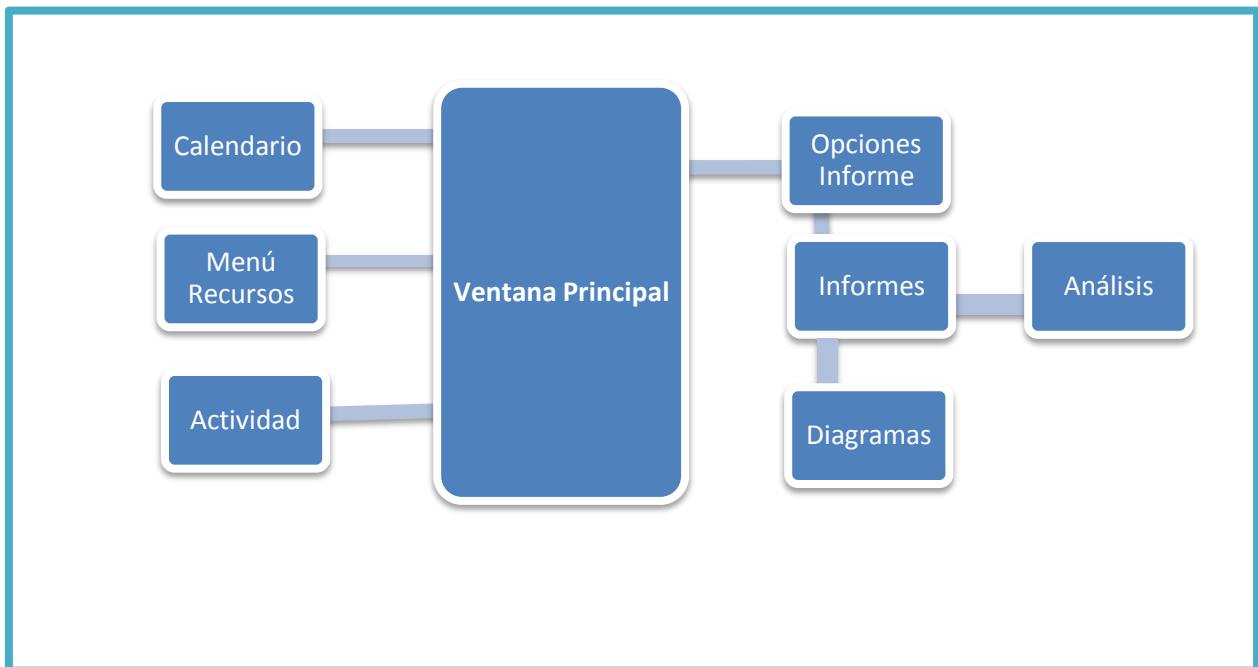
- **QDialog:** muestra diálogos del sistema, como errores, o información adicional para el usuario.

```
mes.warning(0,QString(tr("Error")),QString(tr("Fecha no valida")));
```

- **QFileDialog:** abre un dialogo en el cual podemos seleccionar un directorio para guardar nuestros archivos.

```
QDir directory;  
debPath = QFileDialog::getExistingDirectory(this,tr("Directorio"),directory.path());
```

3.2 Mapa Conceptual Interfaz

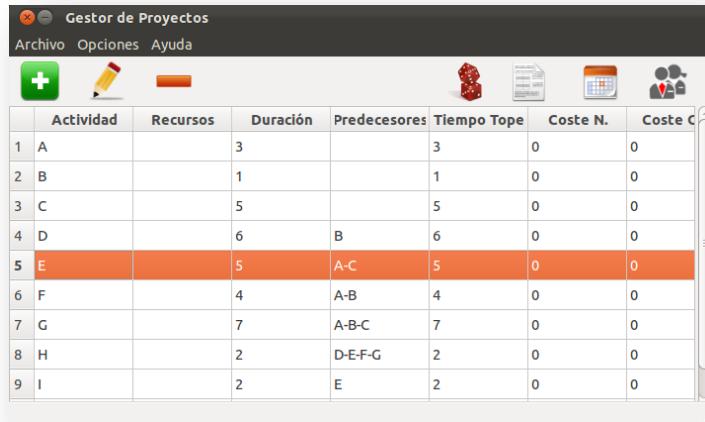


3.3 Funcionamiento Interfaz

La interfaz recibe del calendario las fechas festivas y aquellos días que la administración haya decidido marcar como Laborables. El menú recursos permite crear o eliminar recursos y determinar su capacidad de uso máximo. Por último podremos crear actividades a las que asignar recursos. Será posible además determinar qué algoritmo aplicar gracias a un menú que mostrará todas las opciones posibles a realizar, estas opciones podrán ser guardadas para seguir realizando cambios en las actividades, recursos o fechas y generar finalmente un informe. Este informe consta de dos partes, el informe plano, que podrá ser imprimido, y además un diagrama de Gantt y un histograma de utilización de recursos. Hemos incluido además un manual de uso de la aplicación al que se puede acceder desde esta misma en la barra de herramientas/ayuda/manual.

3.4 Casos de uso

3.4.1 Vista Ventana Principal



The screenshot shows the 'Gestor de Proyectos' application interface. At the top, there is a menu bar with 'Gestor de Proyectos', 'Archivo', 'Opciones', and 'Ayuda'. Below the menu is a toolbar with icons for adding (+), editing (pencil), deleting (-), and other project management functions. The main area is a table with the following data:

	Actividad	Recursos	Duración	Predecesores	Tiempo Tope	Coste N.	Coste O.
1	A		3		3	0	0
2	B		1		1	0	0
3	C		5		5	0	0
4	D		6	B	6	0	0
5	E		5	A-C	5	0	0
6	F		4	A-B	4	0	0
7	G		7	A-B-C	7	0	0
8	H		2	D-E-F-G	2	0	0
9	I		2	E	2	0	0

Desde la vista principal vemos todas las actividades que han sido introducidas en el proyecto con todas sus propiedades. Esta vista puede ser cambiada mediante un simple cuadro de diálogo el cual nos permite mostrar las propiedades que queramos. Las duraciones y tiempos máximos se muestran en Instantes por una simple razón, el usuario al introducir los datos de una actividad, no sabe cuando se llevará a cabo, el sólo debe de introducir los datos pertinentes de cada actividad y dejar que la aplicación genere los informes donde se reflejan las fechas del proyecto.

Desde la vista completa podemos ver:

- Nombre de Actividad
- Recursos Asignados
- Duración
- Predecesores
- Tiempo Tope
- Coste Normal
- Coste de Oportunidad

Desde la vista básica:

- Nombre de Actividad
- Recursos asignados
- Duración
- Predecesores

3.4.2 Añadir Actividad.



Gracias a este cuadro de diálogo que hemos creado podemos introducir tantas actividades como queramos. Asignarle los predecesores de actividades que ya existan, y asignarle tantas unidades de recurso como disponibles tenga. Así, si tan solo queremos crear una actividad sin asignar predecesores o recursos añadiremos esta nueva actividad al proyecto de la siguiente forma:

```
void ActivityView::SaveDataClose()
{
    ...
    //tn = duracio int ; tp = duracio - retrasoMax; cnrmal = cost activitat cuoprt=sobrecost
    pro->addActivity((ui->NomAct->text()).toStdString(),tn,tp,ui->PriceSpin->value(),ui->OverRunSpin->value());
    table->setItem(aux,4,new QTableWidgetItem(QString::number(tp)));
    table->setItem(aux,5,new QTableWidgetItem(ui->PriceSpin->text()));
    table->setItem(aux,6,new QTableWidgetItem(ui->OverRunSpin->text()));
    //Añadimos los elementos a la tabla.
    ...
}
```

3.4.2.1 Asignar Predecesores.



Como hemos comentado, podemos añadir predecesores, para ello tenemos un cuadro de diálogo como el que aparece a la izquierda para facilitarnos esta opción.

Este diálogo se genera de forma dinámica mediante checkboxes. O lo que es lo mismo, mostrará tantos posibles predecesores como actividades tenga nuestro proyecto.

A la hora de guardar estas asignaciones tan solo tenemos que comprobar si el checkbox correspondiente a una actividad ha sido marcado de la siguiente forma:

```
if(firstPred){
    for(int i=0;i<nSize;i++){
        if(chcks[i]->isChecked()){
            if(any) textPred->append("-.");
            any=true;
            textPred->append(chcks[i]->text());
            pro->addRelation(chcks[i]->text().toStdString(),ui->NomAct->text().toStdString());
        }
    }
}
```

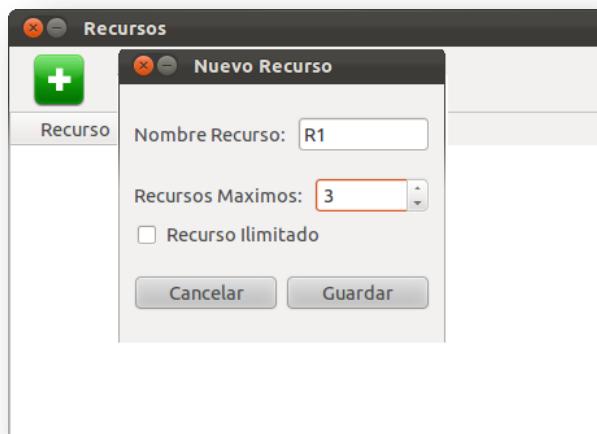
3.4.2.2 Asignar Recursos.

Al igual que para añadir predecesores, cuando queremos añadir recursos se abre un diálogo con tantos recursos como tengamos en nuestro proyecto, mediante un spinbox podemos asignar las unidades que queramos de un recurso a una actividad.

```
for(int i=0;i<resSize;i++){
    if(spiners[i]->value()>0{
        if(any) textRes->append(", ");
        textRes->append(labels[i]->text().append("("));
        textRes->append(spiners[i]->text().append(") "));
        string nameRes = labels[i]->text().toStdString();
        string nameAct = ui->NomAct->text().toStdString();
        int units = spiners[i]->value();
        pro->allocateResourceActivity(nameRes,nameAct,units);
        any=true;
    }
}
```



3.4.3 Añadir Recurso.



Cualquier proyecto necesita definir recursos para poder llevarlo a cabo. Estos recursos pueden desde el tipo humano, en el que se requieren ‘horas de trabajo’ por parte del recurso, o del tipo material, como por ejemplo, las máquinas que se necesitan en una cadena de montaje. Nosotros hemos querido representar los recursos de forma que el usuario pueda darle nombre y asignarle un número de unidades máximas disponibles, en las cuales también tenemos en cuenta un supuesto recurso ilimitado.

```
if(resUnlimited->isChecked()){
    ui->ResourceTable->setItem(aux,1,new QTableWidgetItem(QString("Ilimitado")));
    pro->addResource(resName->text().toStdString(),99999999); //Los recursos 'Ilimitados' tienen un valor muy alto.
} else{
    ui->ResourceTable->setItem(aux,1,new QTableWidgetItem(resAvailable->text()));
    pro->addResource(resName->text().toStdString(),resAvailable->text().toInt());
}
```

3.4.3.1 Actualizar recurso

Los recursos, a veces, necesitan ser modificados por demandas de la empresa o del mismo proyecto, aumentando o disminuyendo sus unidades. Así que con una simple modificación conseguimos que también se pueda modificar un recurso desde la misma ventana.

```
if(resUnlimited->isChecked()){
    ui->ResourceTable->setItem(aux,1,new QTableWidgetItem(QString("Ilimitado")));
    pro->modifyResource(resName->text().toStdString(),99999999);
} else{
    ui->ResourceTable->setItem(aux,1,new QTableWidgetItem(resAvailable->text()));
    pro->modifyResource(resName->text().toStdString(),resAvailable->text().toInt());
}
```

3.4.4 Añadir Festivos y Laborables.



Uno de los factores más importantes a la hora de definir un proyecto es saber en qué fechas se va a llevar a cabo. Para realizar esta labor hemos creado nuestro propio gestor de fechas divido en dos conceptos Festivos y laborables extras. El programa viene cargado con una lista de los festivos nacionales de España, esta lista no puede ser borrada, para poder asignar un festivo como día laboral el cliente debe de crear un laborable Extra en su lista correspondiente, adicionalmente se pueden crear nuevos días festivos pudiendo ser estos eliminados más adelante. Por otro lado tenemos los laborables extra, por norma general suelen ser días festivos en los que, como hemos explicado antes, el cliente quiere que se tengan en cuenta para desarrollar el proyecto, no obstante, también tenemos la posibilidad de fijar los sábados o domingos (o ambos), como días laborables durante todo el año.

3.4.4.1 Cómo se gestionan

A la hora de generar un proyecto (*ver apartado 3.4.4 Generar Proyecto*), se requiere que el usuario introduzca una fecha de inicio del mismo. Una vez hecho esto nuestra aplicación se encarga de buscar aquellos días que se adaptan a nuestro calendario laboral con un pequeño algoritmo basado en expresiones condicionales mediante dos funciones:

```
void Window::getWorkDays(){
    int auxDura = pro->getEnd()->getTMin(); //Obtenemos la duración en instantes de tiempo del proyecto.
    int i=0;
    workDays->clear();
    while(!dayValid(startingDate->addDays(i))){ //Asignamos un día válido para el día de inicio.
        i++;
    }
    workDays->append(startingDate->addDays(i));
    auxDura=auxDura-1;
    int j=1;
    for(int i=0;i<auxDura;i++){ //Obtenemos el resto de días válidos
        j=1;
        while(!dayValid(workDays->at(i).addDays(j))) j++;
        workDays->append(workDays->at(i).addDays(j));
    }
}
```

```

bool Window::dayValid(QDate date)
{
    //If it's a work day just return TRUE
    for(int i=0;i

```

3.4.5 Generar Proyecto.

El menú de opciones no es más que una vista rápida para saber que algoritmos aplicar sobre el proyecto. Gracias al uso de GroupBoxes, dividimos la ventana en distintas secciones que permiten diferenciar claramente los algoritmos que hemos incluido para generar un proyecto. Además del algoritmo de camino crítico, que se genera siempre, podemos elegir entre nivelación de recursos, mínimo coste, limitación de recursos y atraso adelanto. Detallamos a continuación el ejemplo de condiciones que se cumplen para generar uno de ellos, mínimo coste:

```

//Mínimo Coste
if(CalcMinCos->isChecked()){ //Si ha sido marcado para generar este informe, entra aquí.

    pro->startDebug("minCostMinDuration");
    if(maxOverrun->value()>0){ //Comprobamos si se ha de tener en cuenta el sobrecoste.
        pro->calculPathMinTimeMinCost(maxOverrun->value());
    }else{
        pro->calculPathMinTimeMinCost();
    }
    pro->finishDebug();
    getWorkDays(); // Ahora que ya tenemos la duración en instantes de tiempo del proyecto
                    // calculamos el vector de días REALES del proyecto.
    rep = new Report(workDays,pro,0);      //Generamos el informe.
    rep->setWindowTitle(QString("Informe Mínimo Coste"));
    rep->show();
}

```

3.4.6 Informe.



Los informes están divididos en tres partes, informe de actividades, calendario y los informes de Gantt e Histograma de Recursos.

3.4.6.1 Informe Actividades

El informe de actividades no es más que una tabla en la que hemos añadido las actividades y sus datos obtenidos a partir de los informes generados. Los datos de la tabla se completan con una simple función:

```
for(int i=0;i<nAct;i++){
    actTable->setItem(i,0,new QTableWidgetItem(QString::fromStdString(vecAct.at(i)->getName())));
    actTable->setItem(i,1,new QTableWidgetItem(workDays->at(vecAct.at(i)->getEarlyStart()).toString("dd/MM/yyyy")));
    actTable->setItem(i,2,new QTableWidgetItem(workDays->at(vecAct.at(i)->getLateStart()).toString("dd/MM/yyyy")));
    actTable->setItem(i,3,new QTableWidgetItem(workDays->at(vecAct.at(i)->getEarlyEnd()).toString("dd/MM/yyyy")));
    actTable->setItem(i,4,new QTableWidgetItem(workDays->at(vecAct.at(i)->getLateEnd()).toString("dd/MM/yyyy")));

    actTable->setItem(i,5,new QTableWidgetItem(QString::number(vecAct.at(i)->getHTotal())));
    actTable->setItem(i,6,new QTableWidgetItem(QString::number(vecAct.at(i)->getHFree())));
    actTable->setItem(i,7,new QTableWidgetItem(QString::fromStdString(vecAct.at(i)->sucesorsToString())));
}
```

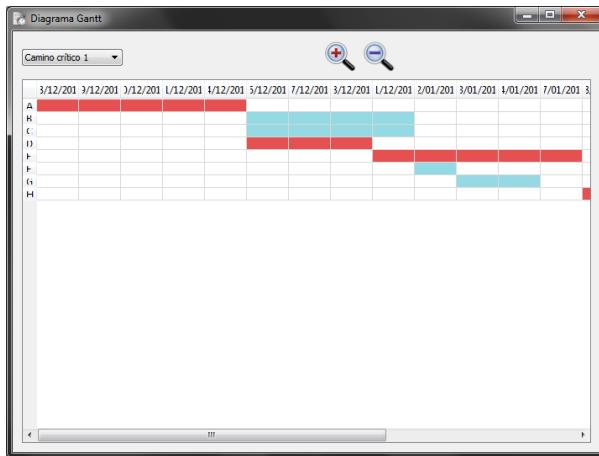
3.4.6.2 Calendario

El calendario nos muestra pintados los días en los que se lleva a cabo el proyecto. Para esto, hacemos uso de QCalendarWidget, una función nativa de QT que nos devuelve un calendario real, sobre este pintamos las celdas de aquellos días que el informe considera como laborables.

```
int i=-1;
do{
    i++;
    brus.setColor(Qt::black);
    format.setForeground(brus);
    format.setBackground(QColor(25,125,35,127));
    Cal->setDateTextFormat(workDays->at(i).format);
    calList->addItem(new QListWidgetItem(workDays->at(i).toString("dd/MM/yyyy")));
}while(workDays->at(i)!=workDays->last());
```

3.4.6.3 Gantt e Histograma

3.4.6.3.1 Gantt.



Una vez más, gracias al uso de las QTableWidget conseguimos mostrar un diagrama de Gantt que nos mostrará la duración de las actividades a lo largo del proyecto además de remarcar los posibles caminos críticos.

```

for(int i=0;i<camCrits;i++){           //Creamos el ComboBox con los caminos críticos
    comboCrit->addItem(QString("Camino crítico ").append(QString::number(i+1)));
}
connect(comboCrit,SIGNAL(currentIndexChanged(int)),this,SLOT(refreshCCrit(int)));   //Si cambia el camino crítico, cambia el dibujo

for(int i=0;i<nAct;i++){
    actList.append(QString::fromStdString(vecAct.at(i)->getName()));
    if(pro->getCriticalPaths()->at(0)->includesActivity(vecAct.at(i))) paintActivity(i,vecAct.at(i)->getTMin(),vecAct.at(i)->getTNormal(),true);
    else      paintActivity(i,vecAct.at(i)->getTMin(),vecAct.at(i)->getTNormal(),false);
}

```

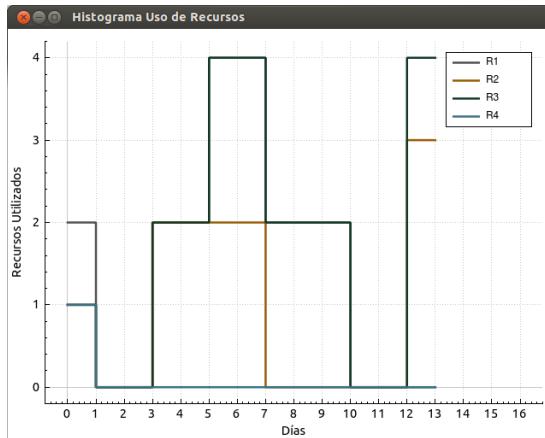
Al crear el ComboBox, hemos tenido que asociar una señal al cambio de Camino Crítico elegido por el usuario que se hará repitiendo prácticamente las mismas funciones que las usadas anteriormente:

```

void refreshCCrit(int op){
    int nAct = pro->sizeActivities();
    vector<Activity*> vecAct = pro->getActivities();
    for(int i=0;i<nAct;i++){
        if(pro->getCriticalPaths()->at(op)->includesActivity(vecAct.at(i))) paintActivity(i,vecAct.at(i)->getTMin(),vecAct.at(i)->getTNormal(),true);
        else      paintActivity(i,vecAct.at(i)->getTMin(),vecAct.at(i)->getTNormal(),false);
    }
    ganttTable->update();
}

```

3.4.6.3.2 Histograma.



Cómo se ha mencionado en las primeras páginas de esta memoria, para realizar el histograma hemos hecho uso de los ficheros QCustomPlot.h y QCustomPlot.cpp proporcionados de forma libre por la página <http://www.workslikeclockwork.com/index.php/components/qt-plotting-widget>. Esta librería nos permite pasarle los datos de uso de recursos en un vector y modificar los ejes X e Y a nuestro gusto, pudiendo poner en el eje X los días reales del proyecto. El código (reducido) que hemos utilizado es el siguiente:

```

int nvecRes = pro->sizeResources();
vector<Resource*> *vecRes = pro->getResources();

customPlot= new QCustomPlot();
// give the axes some labels:
customPlot->xAxis->setLabel("Días");
customPlot->yAxis->setLabel("Recursos Utilizados");
customPlot->legend->setVisible(true);
customPlot->legend->setFont(QFont("Helvetica", 9));
customPlot->legend->setPositionStyle(QCPLegend::psTopRight);
QPen pen;    QStringList lineNames;

int xLength = pro->getEnd()->getTMin();
int max1=0;
for(double i=0;i<nvecRes;i++){
    lineNames<< QString::fromStdString(vecRes->at(i)->getName());
    //generate data
    QVector<double> yValRes = QVector<double>::fromStdVector(pro->getAllocationResourcePerDay(vecRes->at(i)->getName()));
    QVector<double> xVal(xLength);
    //paint plot for this resource
    customPlot->addGraph();
    for(int j=0;j<xLength;j++){
        if(yValRes.toStdVector().at(j)>max1) //Asignamos los valores de uso del recurso al Vector
            max1=yValRes.toStdVector().at(j);
        xVal[j]=j;
    }
    customPlot->graph()->setData(xVal,yValRes);
    customPlot->graph()->rescaleAxes(true);

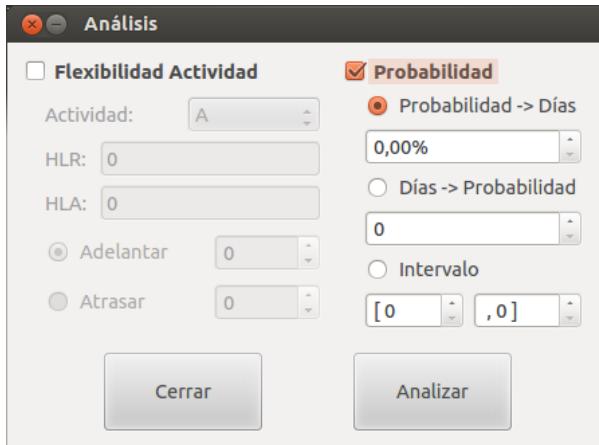
}
QVector<QString> dayLabel;
customPlot->xAxis->setAutoTickStep(false); customPlot->xAxis->setTickStep(2);
customPlot->yAxis->setAutoTickStep(false); customPlot->yAxis->setTickStep(2);

for(int i=0;i<workDays->size();i++) dayLabel.append(workDays->at(i).toString("dd/MM/yyyy"));
customPlot->xAxis->setAutoTickLabels(false);
customPlot->xAxis->setTickVectorLabels(dayLabel);

customPlot->xAxis->setTickLabelRotation(70);

```

3.4.7 Generar Análisis.



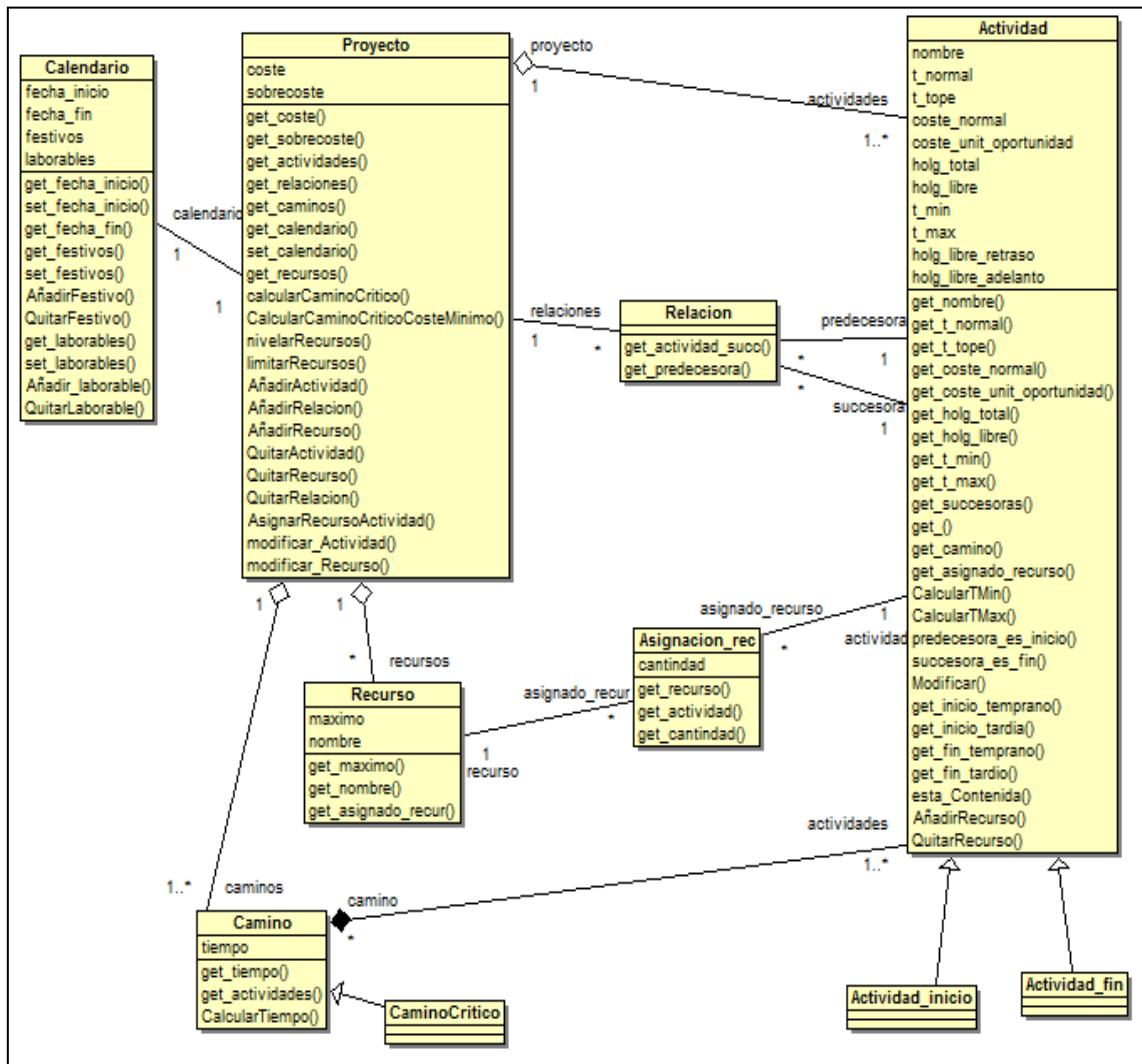
Una vez generado el informe, el usuario puede querer ver las posibles variaciones realizables sobre su proyecto, para ello la aplicación consta de un pequeño cuadro de diálogo el cual permite mostrar un análisis de probabilidad sobre el proyecto, y de flexibilidad sobre la actividad que elija el usuario. El resultado de este análisis se muestra en un simple diálogo de fácil comprensión para el usuario final. La forma de llevar a cabo este análisis desde la interfaz se muestra a continuación:

```

void GenerateAnalysis::submit(){
    QMessageBox *mes = new QMessageBox();
    mes->setWindowTitle(QString("Análisis"));
    QString analisisMes;
    if(!groupFlex->isChecked() && !groupProb->isChecked()){
        analisisMes.append(QString("No ha sido marcaado ningún tipo de análisis"));
    }
    if(groupFlex->isChecked()){ //Hay que realizar un análisis de actividad
        QString flexMes("Flexibilidad Actividad "); flexMes.append(QString::fromStdString(act->getName()));
        flexMes.append(QString(": "));
        if(forRadio->isChecked()){
            flexMes.append(QString::fromStdString(pro->analizeAnticipateTMin(forSpinner->value(),act->getName())));
        }else{
            flexMes.append(QString::fromStdString(pro->analizeDelayTMin(backSpinner->value(),act->getName())));
        }
        analisisMes+=flexMes;
    }
    int num=0;
    if(groupProb->isChecked()){ //Hay que realizar análisis de probabilidad sobre el proyecto
        QString probMes("\nProbabilidad Proyecto: ");
        if(radioProb1->isChecked())// Probabilidad-> Días
            num=pro->getDurationByProbability(probProb->value(),varian->value());
        QString proDate("");
        proDate.append(QString::number(num)).append(" Fecha Fin: "); proDate.append(datesVec->at(num).toString("dd/MM/yyyy"));
        probMes.append(proDate);
        if(radioProb2->isChecked()) // Días -> probabilidad.
            probMes.append(QString::number(pro->getProbabDurationLessThan(durProb->value(),varian->value())));
        if(radioProb3->isChecked()) // Rango días -> probabilidad.
            probMes.append(QString::number(pro->getProbabDurationBetween(int1Prob->value(),int2Prob->value(),varian->value())));
        analisisMes+=probMes;
    }
    mes->information(0,QString("Análisis"),analisisMes);
}

```

3.5 Diagrama de clases



4 Desarrollo

4.1 Entorno de Desarrollo

Hemos decidido realizar la implementación usando C++ por las siguientes razones:

- Muy conocido y sería fácil seguir desarrollando el proyecto en un futuro si fuese necesario, porque su programación es muy intuitiva.
- Control sobre el programa: como programamos a más bajo nivel que otros lenguajes como Java, podemos controlar mejor lo que programamos y diseñar todo lo que podamos necesitar sin problema, aunque eso implique mayor dificultad y líneas de código.
- Orientado a objetos: conceptualmente es más intuitivo ver los componentes que forman el proyecto como objetos de sus respectivas clases, y nos proporciona la abstracción que nos dan los métodos. Por ejemplo: si representamos una actividad como un objeto, es más fácil interpretarla y trabajar con ella, que si la representáramos como un conjunto de variables sin relación aparente y trabajando directamente sobre ella y no mediante métodos.
- Funciones de librería: C++ tiene una serie de librerías útiles para facilitar el trabajo, como sería vector o algorithm, y un conjunto de métodos sobre ellas que nos son muy útiles.
- Memoria dinámica y punteros: la memoria dinámica nos permite optimizar la asignación de la memoria. Pero la principal razón por la que usamos C++ es por el uso de punteros, ya que estos nos permiten crear los elementos del proyecto, asignándoles una posición de memoria; y luego podremos acceder a ellos mediante punteros y modificar sus parámetros a través de estos punteros, sin necesidad de trasladar los cambios que pueden hacer los algoritmos o métodos sobre los objetos, porque trabajamos directamente sobre ellos de forma óptima.

4.2 Estructuras de Datos utilizadas

4.3 Recurso

```
class Resource {  
  
    std::string name;  
    float units_max;           //unidades disponibles del recurso  
  
public:  
  
    Resource(string name);  
    Resource(string name, float units_max);  
    ~Resource();  
  
    void setUnitsMax(int units){ units_max=units;};  
    int getUnitsMax(){ return units_max;};  
    string getName(){ return name;};  
};
```

La clase Resource representa los recursos de los que disponemos. Para representar varios recursos del mismo tipo, tenemos un parámetro que indica el número de recursos del mismo tipo disponibles.

4.3.1 Actividad

```

class Activity {

    string name;

    /*Instantes*/
    int t_normal; // tiempo normal
    int t_tope; // tiempo maximo al que se puede reducir el tiempo normal
    int t_min; // instante de inicio minimo
    int t_max; // instante de inicio maximo
    int dif_t_normal_tope; // tiempo normal - tiempo tope , se usa en el algoritmo ackoff
    int t_secuenciacion; // parametro auxiliar , instante de secuenciacion en la limitacion de recursos

    /*Costes*/
    float cost_normal; //coste normal
    float cost_oportunity; //coste si la actividad se realizase en el tiempo tope

    /*Holguras*/
    int h_total; // holgura total
    int h_free; // holgura libre y holgura libre de retraso

public:

    /*Recurso*/
    typedef struct resource // estructura que representa la asignacion de un recurso
    {
        Resource* resource_asig // recurso asignado
        int units_asig; // unidades del recurso asignadas a la actividad
        resource(Resource *resource_asig, int units);
    };
    std::vector<resource*> resources; // lista de los recursos asignados

    /*Relaciones*/
    typedef struct relation //estructura que representa la relacion de la actividad con otras actividades
    {
        Activity * activity; // actividad con la que se establece la relacion
        relation(Activity * activity);
    };
    vector<relation*> List_Activities_pred; // lista de actividades predecesoras
    vector<relation*> List_Activities_suc; // lista de actividades sucesoras

    /*Metodos*/
    Activity(string name, int t_normal, int t_tope, float cost_normal, float cost_oportunity);
    Activity(string name, int t_normal, float cost_normal);
    Activity();
    ~Activity();

    void clear(); // resetea la actividad
    void modify(int t_normal, int t_tope, float cost_normal, float cost_oportunity); //modifica la actividad
    bool isContained(vector<Activity*> list); //comprueba si la actividad esta incluida dentro del vector

    string getName(){ return name;}
    void setTNormal(int time){ t_normal=time;}
    int getTNormal(){ return t_normal;}
    int getTTope(){ return t_tope; }

    /*Instantes*/
    void setTMin(int t_min){ this->t_min=t_min;}
    int getTMin(){ return t_min;}
    void setTMax(int t_max){ this->t_max=t_max;}
    int getTMax(){ return t_max;}
    int getEarlyStart(){ return t_min;} // Inicio temprano
    int getLateStart(){ return t_max;} // Inicio tardio
    int getEarlyEnd(){ return t_min+t_normal;} // Fin temprano
    int getLateEnd(){ return t_max+t_normal;} // Fin tardio
    int getTSecuenciacion(){ return t_secuenciacion;}
    void setTSecuenciacion(int t){ this->t_secuenciacion=t;}
    void setDiffTNormalTope(){ this->dif_t_normal_tope=t_normal-t_tope;} //Calcula el tiempo que se puede reducir
    void setDiffTNormalTope( int dif){ this->dif_t_normal_tope=dif;}
    int getDiffTNormalTope(){ return dif_t_normal_tope;};

    /*Holguras*/
    void setHTotal(int h_total){ this->h_total=h_total;}
    int getHTotal(){ return h_total;}
    void setHFree(int h_free){ this->h_free=h_free;}
    int getHFree(){ return h_free;}
    void calculHolguraFree(int time); // calcula la holgura libre
    int getHLR(){ return h_free;} // holgura libre de retraso
    int getHLA(); // holgura libre de adelanto
}

```

```

/*Costes*/
float getCostNormal(){ return cost_normal;}
float getOportunityCost(){ return this->cost_oportunity;};

/*Predecesora*/
vector<relation *> *getList_Activities_pred(){return &List_Activities_pred;};
void addPredecesor(Activity * pred);                                     // crea una relacion desde una actividad a esta
int getPositionRelationPredecesor(string name);                         // devuelve la posicion en el vector predecesoras donde esta
//la relacion con la actividad name
int calculMinTime( );                                                 // calcula el tiempo minimo de la actividad
bool predecesorsInitial();                                              // comprueba si es una actividad de inicio

/*Sucesora*/
vector<relation *> *getList_Activities_suc(){return &List_Activities_suc;};
void addSuccesor(Activity * succ);                                         // crea una relacion hace una actividad desde esta
int getPositionRelationSuccesor(string name);                           // devuelve la posicion en el vector sucesoras donde esta la
//relacion con la actividad de nombre
int calculMaxTime( );                                                 // calcula el tiempo maximo de la actividad
bool sucessorsEnd();                                                    // comprueba si es una actividad de fin
string sucessorsToString();                                             // obtiene la lista de nombre de las sucesoras
bool validateDependences();                                              // comprueba si se cumplen las relaciones de precedencias a la hora de calcular los instantes

/*Recursos*/
void incrementResource(int value, string resource);                      // incrementa las unidades asignadas del recurso si este ha sido asignado
//previamente
void decrementResource(int value, string resource);                     // decremente las unidades asignadas del recurso si este ha sido asignado
//previamente
void allocateResource(Resource * resource, int value);                  // asigna a la actividad unidades de un recurso, reseteando las unidades si
//ya estaba asignado
void deleteResource(Resource *resource);                                 // elimina la asignacion que tiene la actividad del recurso resource
int getPositionResource(string resource);                                // obtiene la posicion en el vector resources donde esta el recurso
Activity::resource* getResource(string name_resource);                  // obtiene la asignacion del recurso a la actividad, si no esta asignado
//devuelve null
vector<Activity::resource*> getResources(){return resources;}
bool isEnoughResource(set<disp_instant, CompareDisponibilidades>::iterator disponibility); //comprueba si hay suficientes recursos para
//que pueda empezar la actividad
bool isEnoughResource(disp_instant disponibility);

};

```

Activity es una clase que representa las actividades del proyecto. En ella tenemos los parámetros representativos de las actividades, como la duración, el coste... que son parámetros fijos. Hay otros parámetros, como la holgura libre de retraso y la de adelanto, que están representados, no como parámetros, sino como métodos que devuelven esa información. Estos últimos se calculan cada vez y información que devuelven será actual y evitamos actualizarlos cada vez que se modifican parámetros de los que dependen, como serían los tiempos mínimos.

Para representar las relaciones de precedencia hemos optado por usar una estructura que la represente. En ella guardamos cual es la actividad con la que se relaciona, mediante un puntero que apunta a esa actividad. La razón por la que usamos una estructura con un puntero, y no un puntero directamente, es por la posibilidad en un futuro de añadir nueva información sobre esa relación como parámetros dentro de esa estructura, por ejemplo un coste o indicar solape...

Para facilitar el recorrido por las actividades del proyecto, hemos representado tanto las relaciones de precedencia como las de suceso, de forma que cuando se añade una relación, se añadirá en las relaciones de precedencia de una actividad y en las sucesoras de la otra actividad; así podríamos movernos de la actividad de inicio a la de fin y de la actividad fin a la de inicio sin problema.

Para representar las asignaciones de recursos, tenemos un vector de estructuras que representan cada asignación. Esta estructura guarda un puntero hacia el recurso que se asigna, así tenemos información del recurso asignado fácilmente, además esta estructura tendrá las unidades asignadas, con posibilidad en el futuro de añadir más información sobre la asignación, en forma de parámetro dentro de la estructura.

Por último hemos añadido un parámetro auxiliar que es el instante de secuenciación. La razón de usar este parámetro es porque cuando realizamos la limitación con multipasada, no debemos modificar el instante de inicio antes de saber qué resultado es el mejor, sino, si usásemos el t_{min} como referencia, los resultados siempre se calcularían sobre resultados de una iteración anterior de la multipasada.

Asimismo tenemos un parámetro que representa el número de días que se puede reducir una actividad, calculado como la diferencia entre la duración normal y la duración tope, para que nos sea más sencillo aplicar el algoritmo de mínima duración mínimo coste.

4.3.2 Camino

```

class Path{

    int duration;                                // duracion del camino
    std::vector<Activity*>* path;                // actividades que forman parte del camino

public:

    Path(std::vector<Activity*> *path);
    Path(Path *path);
    Path(Activity* act);
    Path();
    ~Path();
    void copy(Path* path);                        //copia la informacion del camino que se le pasa
    void clear(){ this->path->clear();}          // resetea el camino
    std::vector<Activity*>* getPath(){ return path;}; // calcula la duracion del camino segun las duraciones de las actividades que forman parte
    int getDuration(){ return duration;};
    void setDuration(int duration){ this->duration=duration;};
    bool includesActivity(Activity * act);          // comprueba si la actividad forma parte del camino
    std::string toString();                         // devuelve en string el camino con los nombres de las actividades que forman
};


```

La clase Path la creamos para facilitar nuestro trabajo, ya que representa los caminos del proyecto en forma de vector de punteros que apuntan a las actividades que forman su recorrido.

Nos ha parecido útil representar el concepto de camino, porque así hemos podido crear algunos métodos útiles, por ejemplo para calcular la duración del camino. Este cálculo se podría hacer con un bucle sobre las actividades que lo forman, pero con el método no hará falta repetir este código en varias partes del proyecto, y le damos mayor abstracción sobre como hace el cálculo.

Además, representando este concepto en forma de clase, damos mayor expresividad al proyecto

4.3.3 Proyecto

```

class Project {

    /*ACTIVIDADES*/
    vector<Activity*> activities;                                // actividades del proyecto

    /*CAMINOS CRITICOS*/
    vector<Path*> *critical_paths;                                // caminos criticos del proyecto

    /*RECURSOS*/
    vector<Resource *> * resources;                             // recursos del proyecto

    /*COSTES*/
    float overrun;                                                 // sobrecoste

    /*ACTIVIDADES AUXILIARES*/
    Activity * begin;                                            // actividad auxiliar de inicio
    Activity * end;                                               // actividad auxiliar de fin

    /*DEBUG*/
    string report;                                                 // nombre del fichero donde guardar la informacion de debug
    string sub_report;                                            // nombre del subfichero segun el algoritmo que se debuguee
    ofstream output;                                              // puntero fichero debug
    bool enabled;                                                 // modo debug habilitado o no

private:
    int getPositionActivity(string name);                         // devuelve la posicion de la actividad dentro del vector de actividades
    int getPositionResource(string name);                         // devuelve la posicion del recurso dentro del vector de recursos
    bool validateDependences();                                  // valida que los instantes de tiempo entre relaciones sean correcto,
                                                               //todas las sucesoras deben empezar mas tarde que las predecesoras
    bool validateLimitationMaxUnitsAllocated();                  // valida que la asignacion de cada uno de los recursos, por dia, no supera
                                                               //el maximo de unidades cada dia
    void clear();                                                 // borra los datos calculados del proyecto

    /*CAMINO CRITICO*/
    void extractCriticalPaths( Path* path, Activity * partida ); // extrae los caminos criticos que forman parte del proyecto

    /*NIVELACION RECURSOS*/
    bool levelResources(vector<Resource*> *resources);        // nivela los recursos de forma que para mejorar debe mejorar la suma de
                                                               //los CV de los recursos
    double calculCostResource(string name_resource);            // calcula el coste de la asignacion de ese recurso a lo largo de los dias que
                                                               //dura el proyecto

    /*CALCULO MIN COSTE MIN DURACION*/
    void decrement_days(vector<Path*> *paths,int max_decr_dias,Path* act_to_modif); // actualiza la duracion de las actividades
    float get_min_cost(vector<Path*> paths_a_modif, int n_act,Path * activities_to_modif_pruebas,Path* activities_to_modif_sol); // selecciona
                                                               //la mejor combinacion de actividades que reducen los caminos criticos que no superen el sobrecoste permitido

    Path* select_activities_min_cost(vector<Path*> *paths, float& cost, int &max_decr_days, int cost_searched); // selecciona la mejor
                                                               //combinacion de actividades que reducen los caminos criticos
    Path* select_activities_min_cost(vector<Path*> *paths, float& cost, int &max_decr_days);
    void extractPaths( vector<Path*> *paths,Path * path, Activity* partida ); // obtiene todos los caminos que forman parte del proyecto

    /*LIMITACION RECURSOS*/
    float limitationResourcesSerie(int rule);                   // limitar recursos con el esquema serie y regla de prioridad rule
    float limitationResourcesParalel(int rule);                 // limitar recursos con el esquema paralelo y regla de prioridad rule
    float HBRPMultiPasada();                                    // limitar recursos con multipasada
    float limitationResourcesSerie(deque<Activity*> elegir, string step); // proceso de adelanto/traso (step) de las actividades despues de la
                                                               //limitacion de recursos con esquema serie
    float limitationResourcesParalel(deque<Activity*> elegir, string step); // proceso de adelanto/traso (step) de las actividades despues de la
                                                               //limitacion de recursos con esquema paralelo

    /*DEBUG*/
    void mode_debug(string text);                               // metodos para mostrar los pasos de los algoritmos si esta habilitado el modo debug
    void mode_debug(string before,vector<Activity*> *path, string after);
    void mode_debug(string before,deque<Activity*> path, string after);
    void mode_debug(string before,vector<disp_resource> disp, int t, string after);
    void mode_debug(string before,set<disp_instant, CompareDisponibilidades> *disponibility, string after);

public:
    Project();
    ~Project(void);
    vector<Path*>* getCriticalPaths(){ return critical_paths;}; // devuelve los caminos criticos
    int sizeCriticalPaths(){ return critical_paths->size();} // numero de caminos criticos
    vector<Activity*> getActivities(){ return activities;}; // devuelve las actividades
    int sizeActivities(){ return activities.size();} // numero de actividades
    vector<Resource *>* getResources(){ return resources;}; // devuelve los recursos
    int sizeResources(){ return resources->size();} // numero de recursos
}

```

```

Activity * getEnd(){ return end; }; // actividad de fin
Activity * getBegin(){ return begin; }; // actividad de inicio
float getTotalCost(); // devuelve el coste total de proyecto, incluyendo el sobrecoste
float getOverrun(){ return overrun; }; // devuelve el sobrecoste del proyecto

/*ACTIVIDADES*/
void addActivity(string name, int tn, int tp, float cnormal, float cuoport); // añade una actividad incluyendo el tiempo tope y coste de oportunidad
void addActivity(string name, int tn, float cnormal); // añade una actividad
void modifyActivity(string name, int tn, int tp, float cnormal, float cuoport); // modifica la actividad nombre
Activity * getActivity(string name); // devuelve la actividad nombre
void deleteActivity(string name); // borra la actividad nombre
void updateTMax(); // actualiza el tiempo maximo de las actividades segun el tiempo minimo que tengan

/*RELACIONES*/
void addRelation(string orig, string dest); // añade una relacion de orig a dest
void deleteRelation(string orig, string dest); // borra la relacio de orig a dest

/*RECURSOS*/
Resource * getResource(string name); // devuelve el recurso nombre
void addResource(string name, int units_max); // añade un recurso
void modifyResource(string name, int units_max); // modifica el recurso
void deleteResource(string name); // borra el recurso nombre
void allocateResourceActivity(string resource, string activity, int units); // asigna una cantidad de recurso a una actividad
void deleteResourceActivity(string resource, string Activity); // quita la asignacion del recurso a la actividad
void incrementUnitsResourceActivity(string resource, string activity, int units); //incrementa el numero de unidades del recurso asignadas a la actividad
void decrementUnitsResourceActivity(string resource, string activity, int units); //decremenata el numero de unidades del recurso asignadas a la actividad
vector<double> getAllocationResourcePerDay(string name); // devuelve en el vector las unidades asignadas del recurso cada dia

/*ALGORITMO CAMINO CRITICO*/
void calculCriticalPath(); // algoritmo de calculo de camino critico

/*ALGORITMO DE NIVELACION DE RECURSOS*/
bool levelAllResources(); // nivela todos los recursos permitiendo, de forma que para mejorar debe mejorar la suma de los CV de los recursos
bool levelResources(vector<string> names_rec); // nivela los recursos names_Rec independientemente unos de otros

/*ALGORITMO ACKOFF-SASIENI: MINIMO COSTE MINIMA DURACION*/
void calculPathMinTimeMinCost(); // algoritmo minimo coste minima duracion
void calculPathMinTimeMinCost(int cost_searched); // algoritmo min coste min duracion sin que el sobrecoste diario sea mayor a cost_searched

/*ALGORITMO limitacion DE RECURSOS*/
bool limitarResources(int rule, int eschema); // limitacion de recursos con una regla de prioridad y esquema
bool retrasoAdelantoActivities(int eschema_retraso, int eschema_adelanto); // proceso de adelanto/traso despues de la limitacion

/*ANALISIS*/
float getProbabDurationLessThan(int x, double var); // devuelve la probabilidad de que el proyecto finalize antes de x dias
float getProbabDurationBetween(int x,int y, double var); // probabilidad de que en fin de proyecto este entre x e y
int getDurationByProbability(double x, double var); // instante fin del proyecto con una probabilidad maxima x
string analizeDelayTMin(int days, string name_activity); // devuelve el informe de realizar el analisis de flexibilidad si se retrasa una actividad
string analizeAnticipateTMin(int days, string name_activity); // devuelve el informe de realizar el analisis de flexibilidad si se adelanta una actividad

/*DEBUG*/
void setReportDebug(string path); // asigna un directorio donde guardar los resultados
bool startDebug(std::string sub_report_chosed); // empieza a depurar
void finishDebug(); // termina de depurar
void enableDebug(); // habilita el modo debug
void disableDebug(); // deshabilita el modo debug
};

}

```

La clase Project representa toda la información que puede tener un proyecto de gestión.

Esta clase es quien asigna la memoria a las actividades y recursos, que los guarda en un vector de punteros que apunta a la actividad o recurso. Además, tiene un vector de caminos que representan los caminos críticos, que es apuntado por un puntero a la dirección de memoria donde se guarda el vector; así tenemos siempre disponible la información de los caminos críticos.

Al igual que Activity, hay parámetros del proyecto directamente representados y otros que están representados por medio de métodos, como sería el coste del proyecto, ya que es dependiente del coste de las actividades y si se modifica el coste de alguna actividad, se

tendría que actualizar, y como no es un parámetro muy usado, es más eficiente calcularlo cuando lo necesitamos.

Con dirección a simular la representación de proyectos dada en clase, hemos añadido dos actividades ficticias: INICIO y FIN. Con esto todo el proyecto empieza por el INICIO y termina por FIN. La utilidad de incorporar estas actividades, además de representar con similitud los conceptos de clases, es que, por ejemplo, siempre tendríamos en FIN la duración del proyecto. Si no lo tuviésemos, tendriáramos que recorrer todas las actividades que no tienen sucesoras y entre ellas elegir la de mayor finalización, y teniendo en cuenta que la duración del proyecto es un parámetro que usamos mucho, con el uso de FIN ganamos en eficiencia. También cabe decir que para el método del camino crítico es muy útil esta tener INICIO y FIN, porque evitaríamos tener que encontrar cuales son las actividades inicio, ya que sabemos que empezamos por INICIO. A cambio tendremos que tener en cuenta una serie de pasos que se deberán hacer para trabajar con estos nodos: cuando se crea una actividad automáticamente se añade esta actividad como sucesora de inicio y como predecesora de fin, y a medida que añadimos relaciones de esta actividad a otra, las relaciones con INICIO y FIN desaparecen, al igual que cuando quitamos relaciones entre actividades, si la actividad se queda sin sucesoras o predecesora, se añade una relación entre esta y INICIO o FIN.

También tenemos, a efectos de comprobar resultados y para exámenes, un modo de depuración que guardarlos los resultados de pasos al realizar los algoritmos, que podemos habilitar y guardar en un fichero .debug.

4.3.4 Otros

```
/* ACTIVIDAD PROCESANDOSE EN LIMITACION DE RECURSOS*/

typedef struct           // estructura para representar las actividades que se estan procesando en la limitacion de
    recursos
{
    int instant;          // instante que termina de procesarse la actividad
    Activity * act;       // actividad procesandose

}procesing_activity;

class CompareActProcesandose{ // clase auxiliar para ordenar de las actividades que estan procesandose por instante
                           //de finde proceso
public:
    bool operator()(procesing_activity d1, procesing_activity d2)
    {
        if (d1.instant < d2.instant) return true;
        return false;
    }
};
```

Estructura que usamos en el algoritmo de limitación de recursos con esquema paralelo, para representar que actividades están realizándose y aun no han terminado, de forma que tiene un puntero que apunta a la actividad que esta procesándose y el instante en que esta termina de procesarse. Debajo tenemos una clase que sirve para ordenar estas estructuras según su instante de fin, de menor a mayor.

```
/*DISPONIBILIDAD DE RECURSO EN UN INSTANTE EN LIMITACION DE RECURSOS*/

typedef struct {           // estructura que representa las unidades disponibles de un recurso
    Resource * resource;   // recurso
    int units_disp;        // unidades disponibles para asignar del recurso
}disp_resource;

typedef struct {           // estructura que representa las unidades disponibles de los recursos en un instante
{
    int instant;           //instante de tiempo
    vector<disp_resource> resources; // unidades de cada recurso disponible en ese instante
}disp_instant;

class CompareDisponibilites // clase auxiliar para permitir ordenar las disponibilidades por el instante
{
public:
    bool operator()(disp_instant d1, disp_instant d2)
    {
        if (d1.instant < d2.instant) return true;
        return false;
    }
};
```

La estructura disp_instant representa las disponibilidades que hay de recurso, para representar estas disponibilidades, guardaremos un vector de unidades se podrán asignar de cada recurso y el instante en que se tiene esa disponibilidad.

Para representar la disponibilidad de cada recurso tenemos la estructura disp_resource, que apunta al recurso del cual indicara su disponibilidad, y el número de unidades que aun se pueden asignar del recurso en ese instante.

La clase de abajo la usamos para ordenar las disponibilidades según el instante que se dan, en orden creciente.

4.4 Desarrollo de Algoritmos

4.4.1 Camino Crítico

```
oid Project::calculCriticalPath(){
    Activity *act;

    /*Limpiar*/
    clear();

    /*Calcular tiempos minimos*/
    end->setTMin(end->calculMinTime());

    /*Calcular tiempos maximos*/
    begin->setTMax(begin->calculMaxTime());

    /*depurar*/
    mode_debug("\n\n----- CAMINO CRITICO-----\n\n");
    for(int i=0; i<this->activities.size();i++)
    {
        act=activities.at(i);
        mode_debug("Actividad:" +act->getName() + " Tmin:" +num_to_str(act->getTMin())+"
                                         Tmax:" +
                                         num_to_str(act->getTMax())+"\n");
    }

    /*Calcular Hogura Libre*/
    begin->calculHolguraFree(end->getTMax());

    /*Calcular caminos criticos*/
    Path * path= new Path();
    extractCriticalPaths(path, this->begin);
}
```

La clase proyecto realiza los siguientes pasos:

- 1) Limpia los cálculos anteriores de tiempo mínimo y tiempo máximo para que no influyan en este cálculo.
- 2) Calcula el tiempo mínimo de todas las actividades, para ello llama al método de la actividad para calcularlo. Como el recorrido que hace el cálculo de tmin es de la actividad de inicio a la de fin, hacemos que la actividad fin llame primero al método de cálculo de tmin, que es recursivo hacia sus sucesoras; así nos aseguramos que todas las actividades tendrán su tmin calculado.
- 3) Calcula el tiempo máximo de todas las actividades llamando al método de la actividad para calcularlo. Como tmax se calcula de la actividad de fin a la de inicio, sería inicio quien llama al método, que es recursivo hacia sus predecesoras; así se calcularán todos los tmax de las actividades. Además a medida que se calcula tmax, se calcula también la holgura total.
- 4) Calcula la holgura libre de todas las actividades.
- 5) Calcula y guarda los caminos críticos que hay.

4.4.1.1 PASO2: calculo tmin

```

int Activity::calculMinTime()
{
    int max=-1;
    Activity * act;

    if(List_Activities_pred.empty())
        return 0;

    /*Examinar lista de predecesoras*/
    for(int i=0; i<List_Activities_pred.size();i++)
    {
        act=List_Activities_pred.at(i)->activity;

        /*Si la sucesora no tiene tiempo minimo, calcularlo*/
        if(act->t_min== -1)
            act->t_min=act->calculMinTime();

        /*Maximo instante de finalizacion temprano entre las predecesoras*/
        if(max<act->t_min+act->t_normal)
            max=act->t_min+act->t_normal;
    }

    return max;
}

```

Para una actividad se examinan todas sus predecesoras, si alguna aun no tiene calculado el tiempo minimo mediante la recursividad lo calcula. Por lo que se realiza una exploracion de actividades siguiendo un orden DFS.

Entre todas sus predecesoras se guarda cual es el instante de finalizacion más temprano (t_{min} predecesora + tiempo normal predecesora).

PARAMETROS DE SALIDA

- Instante de inicio más temprano

COSTE TEMPORAL: $O(n)$

4.4.1.2 PASO 3: calculo tmax

```

int Activity::calculMaxTime()
{
    int min=10000000000000; // infinito
    Activity * act;

    if(List_Activities_suc.empty())
        return t_min;

    /*Examina las sucesoras*/
    for(int i=0; i<List_Activities_suc.size();i++)
    {
        act=List_Activities_suc.at(i)->activity;

        /*Si la sucesora no tiene calculado el TMAX, calcularlo*/
        if(act->t_max== -1)
        {
            act->t_max=act->calculMaxTime();

            /*Holgura total*/
            act->h_total=act->t_max-act->t_min;
        }

        /*Minimo instante de comienzo mas tardio de las sucesoras*/
        if(min>act->t_max-t_normal)
            min=act->t_max-t_normal;
    }

    return min;
}

```

Para una actividad, recorre sus actividades sucesoras, calculando su tmax mediante recursividad, en caso de que no lo tenga, realizando una exploración recursiva DFS.

Entre las sucesoras se asigna a tmax el instante de comienzo más tardío entre las sucesoras menos la duración de la actividad para la que se calcular el tmax. Asimismo como ya se tiene calculado el tmin de la actividad, calculamos la holgura total como la diferencia entre tmax y tmin.

PARAMETROS DE SALIDA

- Instante de inicio máximo

COSTE TEMPORAL: O(n)

4.4.1.3 PASO 4: calculo holgura libre

```

void Activity::calculHolguraFree( int time)
{
    h_free=time;           // como maximo usamos la propia duracion del proyecto

    /*Es la actividad de fin*/
    if(List_Activities_suc.empty())
        h_free=time-(t_normal+t_min);

    else
    {
        /*Explora sus sucesoras*/
        for(int i=0; i<this->List_Activities_suc.size();i++)
        {
            /*Minima diferencia entre el TMIN de una sucesora y el instante de finalizacion mas temprano de la
            actividad*/
            if(h_free>List_Activities_suc.at(i)->activity->t_min-(t_normal+t_min))
                h_free=List_Activities_suc.at(i)->activity->t_min-(t_normal+t_min);

            /*Calculamos la holgura libre para la sucesora*/
            List_Activities_suc.at(i)->activity->calculHolguraFree(time);
        }
    }
}

```

Para una actividad dada, recorre sus sucesoras, buscando la mínima diferencia entre el instante de comienzo más temprano de la sucesora y el instante de fin temprano de la actividad. Como máximo usamos la duración del proyecto que, como sabemos, la holgura libre no puede ser mayor.

Luego calculamos la holgura libre para cada sucesora aplicando recursividad, así recorremos todas las actividades.

PARAMETROS DE ENTRADA

- Duración del proyecto

COSTE TEMPORAL: O(n)

4.4.1.4 PASO 5: obtener los caminos críticos

```

void Project ::extractCriticalPaths( Path * path, Activity* partida )
{
    Activity * act;

    /*Ultima actividad, se añade el camino critico calculado*/
    if(partida->sucesorsIsEnd())
    {
        Path * finished_path=new Path(path);
        critical_paths->push_back(finished_path);
        path->clear(); // lo reseteamos para poder usarlo
    }
    else
    {

        for(int i=0; i< partida->getList_Activities_suc()->size(); i++)
        {
            act=partida->getList_Activities_suc()->at(i)->activity;

            /*La actividad es critica?*/
            if(act->getHTotal()==0)
            {
                /*Guardamos el camino antes de añadirle la actividad*/
                Path * path2=new Path(path);
                path->getPath()->push_back(act);

                /*Se añade y seguimos examinando su sucesora*/
                extractCriticalPaths( path, act);

                /*Cuando terminamos de examinar el anterior camino, habiendo añadido la sucesora,
                restauramos el camino que teníamos hasta entonces (sin la sucesora)
                y seguimos buscando entre las otras sucesoras*/
                path->copy(path2);
                delete(path2);
            }
        }
        path->clear();
    }
}

```

PARAMETROS DE ENTRADA:

- Camino donde se añadirán las actividades
- Actividad a partir de la cual se calcula los caminos críticos

PARAMETROS DE SALIDA

- Vector de caminos críticos, que se guarda dentro del proyecto

DESCRIPCION:

Si la actividad para la que se calcula los caminos es una actividad conectada a fin, habremos encontrado una camino crítico, lo guardamos en el vector de caminos críticos, y limpiamos el vector donde guardar el camino para otra iteración.

Si tiene sucesoras diferentes a la actividad de fin, examinamos sus sucesoras, si alguna tiene holgura total 0, es una actividad crítica, por lo que guardamos en otro sitio el camino obtenido hasta ese momento, para buscar caminos que contengan las otras predecesoras. Añadiremos la predecesora al camino y seguiremos buscando a partir de esta; cuando termine, restauraremos el camino sin ella, y comprobaremos si alguna otra sucesora es crítica.

COSTE TEMPORAL PEOR CASO: $O(n^2)$.

4.4.2 Mínimo coste mínima duración

4.4.2.1 OPCIÓN 1: FUNCIONAMIENTO NORMAL

```

void Project::calculPathMinTimeMinCost(){

    std::vector<Path*> *paths= new vector<Path*>;
    Path * path=new Path();
    float cost;
    int max_decr_days;
    Path *act_to_modif;
    Activity * act;

    /* INICIALIZACION */
    mode_debug("*****INICIALIZACION*****\n\nCaminos:\n");

    this->overrun=0; //sobrecoste

    /*obtiene en paths todos los posibles caminos del proyecto*/
    extractPaths(paths, path,this->begin);

    /*calcula la duración en día de cada uno de los caminos*/
    for(int i=0; i<paths->size();i++)
    {
        paths->at(i)->calculDuration();
        mode_debug("      "+num_to_str(i)+"o: ",paths->at(i)->getPath()," "+num_to_str(paths->at(i)->getDuration())+"\n");
    }

    /*calcula el número de días que se puede reducir cada actividad , como Tiempo normal - Tiempo tope*/
    for(int i=0; i<activities.size();i++)
    {
        act=activities.at(i);
        act->setDiffTNormalTope();
        mode_debug("\nT.Normal-T.tope "+act->getName()+" = "+num_to_str(act->getDiffTNormalTope())+"\n");
    }

    /*          EJECUCION          */
    mode_debug("\n***** EJECUCION *****\n");

    /*Elegimos las actividades a modificar que tengan min coste, el coste mínimo y cuantos days se reducirá cada actividad elegida*/
    act_to_modif=select_activities_min_cost(paths,cost,max_decr_days);

    /*Se puede reducir la duración del proyecto?*/
    while(cost!=INF)
    {
        mode_debug("Sobrecoste Minimo = "+num_to_str(cost)+"\n");
        mode_debug("Actividades a modificar: ",act_to_modif->getPath(),"\n");

        /*Sobrecoste*/
        this->overrun+=cost;

        /*Se actualizan los caminos, reduciendo los días que duran sus actividades*/
        decrement_days(paths,max_decr_days,act_to_modif);

        /*Elegimos las actividades a modificar a min cost y cuantos days se reducen*/
        act_to_modif=select_activities_min_cost(paths,cost,max_decr_days);
    }

    /*Actualiza los parámetros del proyecto según los resultados*/
    for(int i=0; i<activities.size();i++)
    {
        act=activities.at(i);
        act->setTNormal(act->getTTope()+act->getDiffTNormalTope());
    }
    this->calculCriticalPath();
    delete(paths);
    delete(path);
    delete(act_to_modif);
}

```

Este método aplica el algoritmo reduce la duración de proyecto con mínimo coste.

PASOS:

- 1) Inicialización de los parámetros que vamos a usar.
 - a. Obtener todos los caminos posibles que tiene el proyecto y calcular la duración de cada uno.

- b. Calcular el número de días que se podrá reducir cada actividad.
- 2) Ejecución: Mientras que se puedan reducir todos los caminos críticos del proyecto, se podrá reducir la duración del proyecto.
- Buscamos la combinación de actividades a reducir, de forma que todos los caminos críticos se puedan reducir, y esta combinación sea la de menor coste. Comprobamos que se puedan reducir los caminos críticos. Si no es posible, el coste de la reducción será infinito y terminaríamos, si no, además obtendremos el número máximo de días que se podrán reducir las actividades seleccionadas.
 - Con el número de días que se reducen las actividades, aplicaremos esta reducción a la duración de las actividades, y actualizaremos la duración de los caminos que las contienen.

ESTRUCTURAS USADAS:

- Vector: en este algoritmo usamos un vector de caminos, como hemos explicado anteriormente. La razón de usar un vector es porque estas estructuras son más eficientes en el acceso a sus elementos, que otras estructuras, y la mayor parte de operaciones que hacemos son de acceso a sus elementos, y muy pocas operaciones de inserción.

4.4.2.1.1 PASO 1.A: extraer caminos

```
void Project :: extractPaths( vector<Path*> *paths, Path * path, Activity* begin ){  
  
    Activity * act;  
  
    /*No hay mas actividades, se añade el camino a la lista*/  
    if(begin->successorsEnd())  
    {  
        Path * finished_path=new Path(path); //guarda el camino y lo añade  
        paths->push_back(finished_path);  
        path->clear(); //limpia el camino para posteriores usos  
    }  
    else  
    {  
        /*Tiene sucesoras y las examinamos*/  
        for(int i=0; i< begin->getList_Activities_suc()->size(); i++)  
        {  
            act=begin->getList_Activities_suc()->at(i)->activity;  
            Path * path2=new Path(path); // guardamos el camino antes de modificarlo  
  
            /*Se añade una de las sucesoras y seguimos examinando su sucesores*/  
            path->getPath()->push_back(act);  
            extractPaths(paths, path, act);  
  
            /*Cuando terminamos de examinar el anterior camino,  
            restauramos el camino que teníamos hasta entonces y seguimos buscando*/  
            path->copy(path2);  
            delete(path2);  
        }  
        path->clear();  
    }  
}
```

Este método extrae todo los caminos de proyecto. Para cada actividad examina sus sucesoras y crea un camino para cada una de ellas, de forma que hace un recorrido DFS. Cuando se ha examinado una de las sucesoras, restaura el camino que había hasta la actividad de partida y examina los caminos en que se bifurca su predecesora.

PARÁMETROS ENTRADA:

- Camino donde se ha guardado la lista de actividades que componen el camino hasta la actividad
- Actividad a partir de la cual partimos para encontrar los caminos que parten de ella

PARAMETROS DE SALIDA:

- Vector de caminos donde devuelve todos los caminos que se encuentran

COSTE TEMPORAL PEOR CASO: $O(n^2)$

4.4.2.1.2 PASO 2.A: seleccionar actividades a mínimo coste

```

Path * Project::select_activities_min_cost(std::vector<Path*> *paths, float &cost, int & max_decr_days)
{
    int Time=max(paths); // cual es la duracion del camino que tarda mas
    int next_instant=0;
    Path* activities_to_modif_pruebas=new Path();
    Path* activities_to_modif_solution=new Path();
    std::vector<Path*> paths_to_modif;

    mode_debug("Maxima duracion a reducir = "+num_to_str(Time)+"\n");

    /*Guardar todos los caminos que se quieren reducir porque son los que mas tardan, que seran los caminos criticos*/
    for(int i=0; i< paths->size();i++)
    {
        if(paths->at(i)->getDuration()==Time)
            paths_to_modif.push_back(paths->at(i));
    }

    /*calcula el coste de modificar los caminos criticos*/
    cost=get_min_cost(paths_to_modif, 0, activities_to_modif_pruebas, activities_to_modif_solution);

    /*No es posible reducir todos los caminos criticos*/
    if(cost==INF)      return NULL;

    /*Se elige, de los caminos que no contienen actividades a reducir, el de mayor duracion*/
    next_instant=calcul_next_instant(paths,activities_to_modif_solution);

    /*obtener el minimo numero de dias que se pueden decrementar las actividades que forman parte de los caminos a reducir*/
    max_decr_days=min_dif_between_tnormal_ttope(/*paths,*/activities_to_modif_solution->getPath());

    /*Si hay algun camino que no contiene actividades a reducir*/
    if(next_instant!=-1)
    {
        /*calcular el minimo entre: combertir el camino no critico, de maxima duracion , en critico o
        los dias maximo que se pueden reducir todas las actividades segun su t.normal-t.tope */
        if(max_decr_days>Time-next_instant)
            max_decr_days=Time-next_instant;
    }

    /*numero de dias que se reduce * coste*/
    cost*=max_decr_days;

    delete(activities_to_modif_pruebas);
    return activities_to_modif_solution;
}

```

Selecciona las actividades que se deben reducir para que se reduzcan todos los caminos críticos y por tanto el proyecto, con mínimo coste; además devuelve este coste y el numero de días que se puede reducir estas actividades seleccionadas.

PARÁMETROS ENTRADA:

- Todos los caminos de los que se compone el proyecto

PARÁMETROS SALIDA

- Coste de reducir las actividades que forman parte de los caminos críticos
- Máxima Reducción de días que se reducirá el proyecto
- Actividades que se reducirán para disminuir la duración del proyecto. Cabe decir que aunque la función retorna un camino, este no se interpreta como tal, sino que se interpreta como una lista de actividades. La razón por la que hemos usado un objeto path y no un vector de actividades es porque la clase path tiene algunos métodos, como su constructor y copy, que no son muy útiles por este algoritmo.

PASOS:

- 1) Guardar cuales son los caminos críticos.
- 2) Seleccionar las actividades que se modificaran a mínimo coste y el coste.
 - a. Si el coste es infinito es que no se pueden reducir todos los caminos críticos más.
- 3) Buscar entre los caminos no críticos, cual es el siguiente instante que hará un camino no crítico , crítico.
- 4) Buscar el máximo número de días que se pueden reducir las actividades que coincide con el mínimo número de días que se puede reducir entre todas las actividades.
- 5) Calcular que es más pequeño, el resultado del punto 3 o el del punto 4.
- 6) Calcula el coste como el número de días que se reduce el proyecto por el coste.

ESTRUCTURAS USADAS:

- Vector: en este algoritmo usamos un vector de caminos, como hemos explicado anteriormente. La razón de usar un vector es porque estas estructuras son más eficientes en el acceso a sus elementos, que otras estructuras, y la mayor parte de operaciones que hacemos son de acceso a sus elementos, y muy pocas operaciones de inserción.

4.4.2.1.2.1 PASO 2.A.2: calcular actividades a reducir

```

float calcul_overrun(std:: vector <Path*> paths, Path * activities)
{
    int cost=0;
    int j;

    /*No hay actividades seleccionadas*/
    if(activities->getPath()->size()==0)
        return INF;

    /*Comprobar si todos los caminos almenos tienen una actividad de las elegidas para reducir su t.normal */
    for(int i=0; i<paths.size();i++)
    {
        for(j=0; j<activities->getPath()->size();j++)
            if(paths.at(i)->includesActivity(activities->getPath()->at(j)))
                break;

        /*Hay almenos un camino que no tiene ninguna actividad a reducir*/
        if(j==activities->getPath()->size())      return INF; // no es una solucion factible
    }

    /*calcula el coste*/
    for(int i=0; i<activities->getPath()->size();i++)
    {
        for(int j=0;j<paths.size();j++)
        {
            if(paths.at(j)->includesActivity(activities->getPath()->at(i)))
            {
                // solo contamos la actividad una vez, da igual en que camino este
                cost+=activities->getPath()->at(i)->getOpportunityCost();
                break;
            }
        }
    }
    return cost;
}

```

```

float Project:: get_min_cost(std::vector<Path*> paths_to_modif, int n_act, Path * activities_to_modif, Path * solution)
{
    float con_act=INF;
    Activity * act;
    float t_calculated;
    Path * aux=new Path();
    Path * solution_with_act=new Path();
    Path * solution_without_act=new Path();
    float sin_act;

    /*Probar todas las posibles combinaciones de actividades a modificar y elegir la de minimo coste*/

    /*Todas la actividades se han tenido en cuenta*/
    if(n_act==activities.size())
    {
        /*Calcula el sobrecoste de esa elección de actividades*/
        t_calculated=calcul_overrun(paths_to_modif,activities_to_modif); // es factible?
        solution->copy(activities_to_modif);
        return t_calculated;
    }
    else /*BACKTRACKING: solución sin esta actividad, y con esta actividad*/
    {
        aux=new Path(activities_to_modif); // guardamos las actividades seleccionadas hasta el momento

        /*Solución sin haber cogido esta actividad*/
        sin_act=get_min_cost(paths_to_modif, n_act+1,activities_to_modif, solution_without_act);

        activities_to_modif=new Path(aux); // restaurar la lista a como estaba antes de calcular la solución sin la actividad
        act=activities.at(n_act);

        /*Se puede reducir los días de esta actividad?*/
        if(act->getDifTNormalTope()>0)
        {
            /*Solución habiendo añadido la actividad a la solución*/
            activities_to_modif->getPath()->push_back(act);
            con_act=get_min_cost(paths_to_modif, n_act+1,activities_to_modif,solution_with_act);
        }
        activities_to_modif=new Path(aux); // restaurar la lista a como estaba antes de calcular la solución con la actividad

        /*Es mejor solución, sin la actividad o con la actividad?*/
        if(con_act<sin_act)
        {
            solution->copy(solution_with_act);
            return con_act;
        }
        else
        {
            solution->copy(solution_without_act);
            return sin_act;
        }
    }
}

```

Esta función, por medio de backtracking , hace todas las combinaciones de actividades posibles, para ello primero supone que esa actividad no forma parte de la solución y sigue buscando combinaciones, cuando termina decide tener en cuenta la actividad y vuelve a hacer todas las combinaciones con las otras actividades. Así cuando ya se han examinado todas las actividades, calcula cual sería el sobrecoste, y la combinación que da menor sobrecoste esa será la mejor.

PARÁMETROS DE ENTRADA:

- Vector de caminos que se debe reducir su duración.
- Posición, en el vector de actividades del proyecto, de la actividad que estamos teniendo en cuenta en ese momento.
- Vector de actividades sobre el que hacer pruebas de coger la actividad esa y no cogerla. Este no se interpreta como un camino, sino como una lista de actividades. La

razón por la que hemos usado un objeto path y no un vector de actividades es porque la clase path tiene algunos métodos, como su constructor y copy, que nos son muy útiles por este algoritmo.

-

PARÁMETROS DE SALIDA:

- Conjunto de actividades que reducen el proyecto a mínimo coste. Este no se interpreta como un camino, sino como una lista de actividades.
- Sobrecoste.

PASOS:

- 1) Si ya se han examinado todas las actividades, calcular el sobrecoste de la solución y devolverla.
- 2) Suponer que la actividad que se está teniendo en cuenta en ese momento , no forma parte de la solución y calcular el coste de la solución sin ella
- 3) Suponer que la actividad forma parte de la solución y añadirla, para calcular el coste de esa solución.
- 4) Comparar cual de las 2 opciones es la mejor, y quedarse con la mejor.

COSTE TEMPORAL: $O(n^2)$

4.4.2.1.2.2 PASO 2.A.3: calcular duración máxima de los caminos no críticos

```
int max(std::vector<Path*> *paths) //calcula cual camino dura mas
{
    int max=-1;
    for(int i=0; i<paths->size();i++)
        if(max<paths->at(i)->getDuration())
            max=paths->at(i)->getDuration();
    return max;
}
```

```
int calcul_next_instant(std::vector<Path*> *paths, Path* activities)
{
    int j;
    int next_instant=-1;
    std::vector<Path*> *paths_without_activities= new std::vector<Path*>;
    /*Guardar los caminos que no incluyen actividades que se reducirán*/
    for(int i=0; i< paths->size();i++)
    {
        for(j=0; j<activities->getPath()->size();j++)
            if(paths->at(i)->includesActivity(activities->getPath()->at(j)))
                break;
        /*El camino no tiene ninguna de las actividades*/
        if(j==activities->getPath()->size())
            paths_without_activities->push_back(paths->at(i));
    }

    /*De los caminos sin actividades a reducir, entre estos obtener el que dura mas tiempo */
    next_instant=max(paths_without_activities);

    delete(paths_without_activities);
    return next_instant;
}
```

Calcula cuanto se podría reducir la duración del proyecto, calculando hasta cuanto se puede reducir el proyecto, que se podría reducir hasta que coincida con la máxima duración de los caminos que no contienen actividades a reducir.

PARÁMETROS ENTRADA

- Vector de caminos que se compone un proyecto.
- Conjunto de actividades elegidas para reducir la duración del proyecto. Este no se interpreta como un camino, sino como una lista de actividades.
-

PARÁMETROS SALIDA

- Duración máxima de los caminos que no son caminos críticos y que no contienen actividades que se modificarán. Si devuelve -1 es que todos los caminos contienen actividades que se modificarán.

ESTRUCTURAS USADAS:

- Vector: en este algoritmo usamos un vector de caminos, como hemos explicado anteriormente. La razón de usar un vector es porque estas estructuras son más eficientes en el acceso a sus elementos, que otras estructuras, y la mayor parte de operaciones que hacemos son de acceso a sus elementos, y muy pocas operaciones de inserción.

4.4.2.1.2.3 PASO 2.A.4: mínima diferencia entre t.normal y t.tope

```

int min_dif_between_tnormal_ttope(std::vector<Activity*> *activities){

    float min=INF;

    /*entre las actividades ,encontrar la mínima diferencia entre el t. normal y el tope*/
    for(int i=0; i<activities->size();i++)
    {
        if(min>activities->at(i)->getDifTNormalTope())
            min=activities->at(i)->getDifTNormalTope();
    }

    return min;
}

```

Busca entre todas la actividades que forman parte de algún camino , la actividad que permite una menor reducción de tiempo, ya que es este tiempo el que se reducirán todas las actividades, y será el máximo tiempo que se reducirá el proyecto.

PARÁMETROS ENTRADA

- Conjunto de actividades que se van a reducir

PARÁMETROS SALIDA

- Número de días que se puede reducir el proyecto

COSTE TEMPORAL O(n)

4.4.2.1.3 PASO 2.B: decrementar duración de las actividades

```
void Project::decrement_days(std::vector<Path*> *paths,int max_decr_days,Path*act_to_modif){

    int max_decr_days_allowed=max_decr_days;
    Activity * act;
    Path* path;

    mode_debug("\nDecremento de días de los caminos = "+num_to_str(max_decr_days_allowed)+"\n-----\n");

    /*Reducir el t.normal de las actividades a modificar, lo maximo que se pueda*/
    for(int i=0; i< act_to_modif->getPath()->size();i++)
    {
        act=act_to_modif->getPath()->at(i);
        act->setDiffTNormalTope(act->getDiffTNormalTope()-max_decr_days_allowed);

        /*Reducir la duracion de los caminos que incluyen la actividad*/
        for(int j=0; j<paths->size();j++)
        {
            path=paths->at(j);
            if(path->includesActivity(act))
                path->setDuration(path->getDuration()-max_decr_days_allowed);
        }
    }
}
```

Esta función actualiza las duraciones de las actividades y de los caminos según los días de reducción calculados. Para ello, obtiene el mínimo de días que le está permitido reducir la actividad entre todas las actividades, y comprueba si este es menor al número máximo de días que le habíamos pasado, porque si en el paso anterior, el máximo decremento de días era obtenido como la diferencia entre la duración actual y la duración del siguiente camino que se convertirá en camino crítico, puede que este valor sea más pequeño, en cuyo caso se seguirá reduciendo este, sino se reducirá el mínimo número de días que las actividades se pueden reducir.

PARÁMETROS ENTRADA

- Vector de los caminos que se compone un proyecto
- Máximo de días que se va a decrementar el proyecto
- Actividades elegidas para reducirse. Este path no se interpreta como un camino, sino como una lista de actividades.

PASOS:

- 1) Reducir la duración de las actividades
- 2) Reducir la duración de los caminos

4.4.2.2 OPCIÓN 2: CON UN COSTE MÁXIMO PERMITIDO POR DÍA

```

void Project::calculPathMinTimeMinCost(int cost_searched){

    std::vector<Path*> *paths= new vector<Path*>;
    Path * path=new Path();
    float cost;
    int max_decr_days;
    Path *act_to_modif;
    Activity * act;

    /* INICIALIZACION */
    mode_debug("*****INICIALIZACION*****\n\nCaminos:\n");
    this->overrun=0; //sobrecoste

    /*obtiene en paths todos los posibles caminos del proyecto*/
    extractPaths(paths, path,this->begin);

    /*calcula la duración en día de cada uno de los caminos*/
    for(int i=0; i<paths->size();i++)
    {
        paths->at(i)->calculDuration();
        mode_debug("      "+num_to_str(i)+"@"+" ,paths->at(i)->getPath()," "+num_to_str(paths->at(i)->getDuration())+"\n");
    }

    /*calcula el número de días que se puede reducir cada actividad, como Tiempo normal - Tiempo tope*/
    for(int i=0; i<activities.size();i++)
    {
        act=activities.at(i);
        act->setDiffTNormalTope();
        mode_debug("T.Normal-T.tope "+act->getName()+" = "+num_to_str(act->getDiffTNormalTope())+"\n");
    }

    /*
     *          EJECUCION
     */
    mode_debug("\n***** EJECUCION *****\n\n");

    /*Elegimos las actividades a modificar que tengan min coste, el coste mínimo
     * y cuantos days se reducirá cada actividad elegida*/
    act_to_modif=select_activities_min_cost(paths,cost,max_decr_days, cost_searched);

    /*Se puede reducir la duración del proyecto?*/
    while(cost!=INF)
    {
        mode_debug("Sobrecoste Minimo = "+num_to_str(cost)+"\n");
        mode_debug("Actividades a modificar",act_to_modif->getPath(),"\n");

        /*Sobrecoste*/
        this->overrun+=cost;

        /*Se actualizan los caminos, reduciendo los días que duran sus actividades*/
        decrement_days(paths,max_decr_days,act_to_modif);

        /*Si el coste es mayor al permitido por día, terminamos*/
        if(cost>cost_searched)
            break;

        /*Elegimos las actividades a modificar a min cost y cuantos days se reducen*/
        act_to_modif=select_activities_min_cost(paths,cost,max_decr_days, cost_searched);
    }

    /*Actualiza los parámetros del proyecto según los resultados*/
    for(int i=0; i<activities.size())
    {
        act=activities.at(i);
        act->setTNormal(act->getTTope()+act->getDiffTNormalTope());
    }
    this->calculCriticalPath();
    delete(paths);
    delete(path);
    delete(act_to_modif);
}

```

Este método reduce la duración de proyecto con mínimo coste, permitiendo una reducción de la duración del proyecto siempre que el coste diario sea menor o igual al coste permitido. Las funciones usadas son las mismas que en el algoritmo anterior, la única que cambia es la función `select_activities_min_cost()`

PARÁMETROS ENTRADA

- Coste que se está dispuesto a pagar por día al reducir el proyecto.

PASOS

- 1) Inicialización de los parámetros que vamos a usar.
 - a. Obtener todos los caminos posibles que tiene el proyecto y calcular la duración de cada uno.
 - b. Calcular el número de días que se podrá reducir cada actividad.
- 2) Ejecución: Mientras que se puedan reducir todos los caminos críticos del proyecto se podrá reducir la duración del proyecto.
 - a. Buscamos la combinación de actividades a reducir, de forma que todos los caminos críticos se puedan reducir, y esta combinación sea la de menor coste. Comprobamos que se puedan reducir los caminos críticos, si no es posible el coste de la reducción sería infinito y terminaríamos, sino obtendremos el número máximo de días que se podrán reducir las actividades seleccionadas.
 - b. Si el coste es mayor al permitido, terminar.
 - c. Con el número de días que se reducen las actividades, aplicaremos esta reducción a la duración de las actividades y actualizaremos la duración de los caminos que las contienen.

ESTRUCTURAS USADAS:

- Vector: en este algoritmo usamos un vector de caminos, como hemos explicado anteriormente. La razón de usar un vector es porque estas estructuras son más eficientes en el acceso a sus elementos, que otras estructuras, y la mayor parte de operaciones que hacemos son de acceso a sus elementos, y muy pocas operaciones de inserción.

4.4.2.2.1 PASO 2.A: seleccionar actividades a reducir con un coste máximo permitido

```

Path * Project::select_activities_min_cost(std::vector<Path*> *paths, float &cost, int& max_decr_days, int cost_searched)
{
    int Time=max(paths); // cual es la duracion del camino que tarda mas
    int next_instant=0;
    Path* activities_to_modif_pruebas=new Path();
    Path* activities_to_modif_solution=new Path();
    std::vector<Path*> paths_to_modif;
    stringstream cost_searched_s;
    int days=1;

    mode_debug("Maxima duracion a reducir = "+num_to_str(Time)+"\n");

    /*Guardar todos los caminos que se quieren reducir porque son los que mas tardan, que seran los caminos criticos*/
    for(int i=0; i< paths->size();i++)
    {
        if(paths->at(i)->getDuration()==Time)
            paths_to_modif.push_back(paths->at(i));
    }

    /*calcula el coste de modificar los caminos criticos*/
    cost=get_min_cost(paths_to_modif, 0, activities_to_modif_pruebas, activities_to_modif_solution);

    /*No es posible reducir todos los caminos criticos*/
    if(cost==INF)    return NULL;

    /*Se elige, de los caminos que no contienen actividades a reducir, el de mayor duracion*/
    next_instant=calcul_next_instant(paths,activities_to_modif_solution);

    /*obtener el minimo numero de dias que se pueden decrementar las actividades que forman parte de los caminos a reducir*/
    max_decr_days=min_dif_between_tnormal_ttope(/*paths,*/activities_to_modif_solution->getPath());

    /*Si hay algun camino que no contiene actividades a reducir*/
    if(next_instant!=-1)
    {
        /*calcular el minimo entre: combertir el camino no critico, de maxima duracion , en critico o
        los dias maximo que se pueden reducir todas las actividades segun su t.normal-t.tope */
        if(max_decr_days>Time-next_instant)
            max_decr_days=Time-next_instant;
    }

    /*Calculo de coste*/
    if(cost>cost_searched) // superamos el coste por dia establecido, ya no podremos reducir mas la duracion del proyecto
    {
        cost=INF;
        return NULL;
    }

    delete(activities_to_modif_pruebas);
    return activities_to_modif_solution;
}

```

Selecciona las actividades que se deben reducir para que se reduzcan todos los caminos críticos y por tanto el proyecto, con mínimo coste sin superar el coste máximo, además devuelve este coste y el número de días que se pueden reducir estas actividades seleccionadas.

PARÁMETROS ENTRADA:

- Todos los caminos de los que se compone el proyecto
- Coste máximo permitido por día

PARÁMETROS SALIDA

- Coste de reducir las actividades que forman parte de los caminos críticos
- Máxima Reducción de días que se reducirá el proyecto
- Actividades que se reducirán para disminuir la duración del proyecto. Este path no se interpreta como un camino, sino como una lista de actividades.

PASOS:

- 1) Guardar cuales son los caminos críticos.
- 2) Seleccionar las actividades que se modificaran a mínimo coste y el coste.
 - a. Si el coste es infinito es que no se pueden reducir todos los caminos críticos más.
- 3) Buscar entre los caminos no críticos, cual es el siguiente instante que hará un camino no crítico, crítico.
- 4) Buscar el máximo número de días que se pueden reducir las actividades, que coincide con el mínimo número de días que se puede reducir entre todas las actividades.
- 5) Calcular que es más pequeño, el resultado del punto 3 o el del punto 4.
- 6) Calcula el coste diario, si es mayor al permitido, devolverá coste infinito con lo que terminaría el algoritmo.

4.4.3 Limitación de recursos

```

bool Project::limitarResources(int rule, int eschema)
{
    float t;

    /*Resolver segun el esquema elegido y la regla de prioridad*/
    if(eschema==_SERIE)
        t=limitationResourcesSerie(rule);
    else if(eschema==_PARALEL)
        t=limitationResourcesParalel(rule);
    else
        t=HBRPMultiPasada();

    if(t==INF)
        return false; //no se ha podido aplicar el algoritmo

    /*actualizar los resultados*/
    for(int i=0; i<activities.size();i++)
        activities.at(i)->setTMin(activities.at(i)->getTSecuenciacion());
    end->setTMin(t);

    /*Actualizar t.max y las holguras*/
    this->updateTMax();

    /*comprobar que el resultado es valido*/
    if(!validateDependences())           //cumple las dependencias entre sucesor y predecesor
        exit(-1);
    if(!validateLimitationMaxUnitsAllocated()) //cumple con la limitacion diaria de recursos
        exit(-1);

    return true;
}

```

Método que llama a la limitación de recursos según el esquema y la regla de prioridad indicada. Una vez calculada la solución en los tsecuenciacion de las actividades, se traslada a los tmin, esto es así porque la limitación mutipasada llama a varios esquemas y reglas, cuyos resultados son independientes a resultados anteriores y por ello solo se deben modificar los tiempos mínimos una vez calculado que esquema da la menor duración.

PARÁMETROS ENTRADA

- Esquema de secuenciación , que puede ser: _SERIE, _PARALEL_ MULTIPASADA
- Regla de prioridad para ordenar las actividades a elegir, puede ser: MIN_LFT_FIFO, MIN_EFT_FIFO, MIN_LST_FIFO, MIN_EST_FIFO, MIN_HT_FIFO, MIN_HL_FIFO, MAX_DEM_FIFO, MIN_DUR_FIFO, MAX_NSUC_FIFO

PARAMETROS SALIDA

- Devuelve si se ha podido aplicar el algoritmo, ya que no se podrá aplicar si no existen recursos

PASOS:

- 1) Selecciona un esquema de secuenciación, una regla de prioridad y lo aplica.
- 2) Traslada los resultados guardados en tsecuenciacion a tmin.
- 3) Actualizar según los tmin calculados , el valor de los tmax y las holguras.
- 4) Comprueba que se cumplen las relaciones de precedencia, comprobando que ninguna actividad se adelante a una predecesora.
- 5) Comprueba que se cumple la limitación diaria de un recurso, de forma que la asignación de ese recurso a actividades no supere el máximo de unidades del recurso.

4.4.3.1 ESQUEMA SERIE

```

float Project::limitationResourcesSerie(int rule)
{
    int n_act_to_process=activities.size();
    std::set<disp_instant, CompareDisponibilities> *disponibility=new set<disp_instant, CompareDisponibilities>;
    std::deque<Activity*> selected;
    std::vector<Activity*> examined;
    int t=0;
    float t_max=0;
    disp_resource resource_in_t;
    disp_instant resources_in_t;
    Activity * act;

    if(this->resources->size()==0)
    {
        showErrorInfo(E_ALGORITHM_NOT_EXIST_RESOURCES, " limitacion de recursos en serie");
        return INF;
    }

    /*INICIALIZACION*/
    mode_debug("\n\n*****SERIE*****\n\n");

    /*Disponibilidad inicial, t=0*/
    for(int i=0; i<this->resources->size();i++)
    {
        resource_in_t.resource=resources->at(i);
        resource_in_t.units_disp=resources->at(i)->getUnitsMax();
        resources_in_t.resources.push_back(resource_in_t);
    }
    resources_in_t.instant=0;
    disponibility->insert(resources_in_t);

    /*Actividades elegibles son las actividades de inicio*/
    for(int i=0; i<begin->getList_Activities_suc()->size();i++)
        selected=insert(selected, begin->getList_Activities_suc()->at(i)->activity, rule);
    mode_debug("",disponibility, "\n");

    /*Se examinan todas las actividades*/
    while(n_act_to_process>0)
    {
        mode_debug("\n\nITERACION _____ \n");
        mode_debug("          elegir={" ,selected, "}\n");

        /*Actividad a secuenciar*/
        act=selected.front();
        selected.pop_front();

        /*calcular instante en que se secuenciará*/
        t= get_sequencing_instant(disponibility, act);

        /*Secuenciar*/
        act->setTSecuenciac(t);
        examined.push_back(act);
        mode_debug("          La actividad " +act->getName()+" se secuencia en "+num_to_str(t)+"\n");

        /*Guardamos cual es el instante de fin del proyecto, que será cuando finalice la actividad que termina mas tarde*/
        if(t+act->getTNormal()>t_max)      t_max=t+act->getTNormal();

        /*actualizar elegibles con las sucesoras de la actividad secuenciada, solo si todas las predecesoras de cada sucesora han sido vistas*/
        for(int i=0; i<act->getList_Activities_suc()->size();i++)
            if(all_predecesors_examined(act->getList_Activities_suc()->at(i)->activity, examined)&&!act->succesorsEnd())
                selected=insert(selected,act->getList_Activities_suc()->at(i)->activity, rule);

        /*actualizar disponibilidad*/
        update_disponibility(disponibility, act, t);
        mode_debug("          ",disponibility, "\n");
        n_act_to_process--;
    }
    delete(disponibility);
    return t_max;
}

```

Aplica el esquema serie para realizar la secuenciación de actividades.

PARÁMETROS DE ENTRADA

- Regla de prioridad para establecer el ordenamiento de la lista de actividades elegibles.

PARÁMETROS DE SALIDA

- Nueva duración del proyecto.

PASOS

1. Inicialización
 - a. Inicializar la lista de disponibilidades con las unidades de cada recurso que se tienen.
 - b. Añadir en la lista de actividades elegibles, las actividades que son de inicio. Cuando se insertan en esta lista, se ordenan según la regla de prioridad usando la función insertar, que realizan la ordenación mediante un algoritmo de divide y vencerás con coste $O(\log n)$.
2. Para cada actividad se realizará una iteración
 - a. Seleccionar la actividad más prioritaria, que al ser un vector ordenado por prioridades, se encuentra en la cabeza del vector.
 - b. Calcular el instante en que se secuencia esa actividad.
 - c. secuenciarla y añadirla a la lista de vistas. Guardamos cual será el instante de fin de proyecto, que será el máximo instante que terminaría la actividad al secuenciarse en ese momento.
 - d. Actualizar la lista de elegibles de forma que se añaden las predecesoras de la actividad secuenciada, solo si todas las predecesoras de cada sucesora han sido vistas.
 - e. Actualizar disponibilidad.

ESTRUCTURAS USADAS:

- Vector: en este algoritmo usamos un vector de caminos, como hemos explicado anteriormente. La razón de usar un vector es porque estas estructuras son más eficientes en el acceso a sus elementos, que otras estructuras, y la mayor parte de operaciones que hacemos son de acceso a sus elementos, y muy pocas operaciones de inserción.
- Deque: usamos esta estructura porque nos interesa recrear el comportamiento de una pila ordenada, donde la cabeza contendrá siempre el máximo. Asimismo cuando eliminemos o insertemos un elemento, al poder acceder tanto a los elementos del principio como del final, podremos eliminar el que está en la cabeza, al igual que las pilas. La razón por la que hemos usado esta estructura y no una pila en sí, es porque con esta estructura podemos además de insertar elementos en cualquier posición, no solo insertarlos en la cabeza, y así podremos ordenar su contenido como queramos. En verdad, usamos esta estructura por sus métodos de desapilar, que nos resultan muy útiles, y su gran eficiencia la hora de insertar y eliminar elementos, que al tratarse de la lista de actividades seleccionadas se producen muchas operaciones de inserción

y borrado. Otro aspecto importante es que esta estructura nos permite tener duplicados, no como set.

- Set: utilizamos esta estructura, ya que nos permite asignar funciones para ordenar como queramos los elementos que la componen, cosa que la dota de una gran eficiencia, aunque es muy ineficiente recorrer sus elementos. Aun así , debido al hecho de que no pueden haber instantes repetidos, podremos usar esta estructura que no los permite.

COSTE TEMPORAL: $O(n)$

4.4.3.1.1 PASO 1.A Y 2.D: inserción según regla de prioridad

```
deque<Activity*> insert(deque<Activity*> activities, Activity * act, int rule_prio){

    /*Inserta la actividad act de forma ordenada*/
    if(activities.size()==0)
    {
        activities.push_back(act);
        return activities;
    }else
        /*busca la posicion donde insertar la actividad*/
        return binsearch(activities, act, 0, activities.size()-1, rule_prio);
}
```

Función que ordena las actividades, según una regla de prioridad elegida, mediante la llamada a un algoritmo divide y vencerás. Para más información consultar el apartado “Librería de reglas” de este mismo documento.

PARAMETROS DE ENTRADA

- Lista de actividades donde insertar en elemento
- Actividad a insertar
- Regla para determinar donde se insertara la actividad, según el ordenamiento que establece esa regla

PARAMETROS DE SALIDA

- Lista de actividades ordenadas

COSTE TEMPORAL: O(log n)

4.4.3.1.2 PASO 2.B: obtener instante secuenciación

```
bool Activity::isEnoughResource(std::set<disp_instant, CompareDisponibilities>::iterator disponibility)
{
    /*Comprobar que para cada disponibilidad de un recurso, no se consume más alla de esa disponibilidad*/
    for(int i=0; i<disponibility->resources.size();i++)
    {
        resource * res=getResource(disponibility->resources.at(i).resource->getName());
        if(res!=NULL&&res->units_asig>disponibility->resources.at(i).units_disp)  return false;
    }
    return true;
}
```

```
int get_sequencing_instant(std::set<disp_instant, CompareDisponibilities> *disponibility, Activity * act)
{
    int t_earliest=0;
    set<disp_instant, CompareDisponibilities>::iterator it=disponibility->begin();
    Activity * act_pred;

    /*Encontrar el instant mas pronto que puede empezar*/
    if(!act->predecesorInitial()) // si no es una actividad de comienzo
    {
        /*Calcular cual de sus predecesoras termina mas tarde, que seria el isntante mas pronto que podria empezar la actividad*/
        for(int pred=0;pred<act->getList_Activities_pred()->size();pred++)
        {
            act_pred=act->getList_Activities_pred()->at(pred)->activity;
            if(act_pred->getTSecuenciacion()+act_pred->getTNormal()>t_earliest)
                t_earliest=act_pred->getTSecuenciacion()+act_pred->getTNormal();
        }
    }

    /*Situarse en el instante que mas pronto que puede empezar donde haya disponibilidad de recursos*/
    while(disponibility->end()!=it&& it->instant<t_earliest) it++; // asi tinc dubte de si shauria de fer it-- si no es igual al isntat

    /*Comprobar si todo el tiempo que dura la actividad desde que se inicia hay suficientes recursos*/
    while(disponibility->end()!=it&&it->instant< t_earliest+act->getTNormal() )
    {
        /*En la lista no hay mas disponibilidades, por lo que no hay impedimento para que comience en el t_earliest*/
        if(disponibility->end()==it)          return t_earliest; // t mas pronto

        /*Si no hay suficiente recurso en ese instante, la actividad comenzara el siguiente instante,
        siempre que este instante lo permita, cosa que se comprueba durante la siguiente interacion*/
        if(!act->isEnoughResource(it))
        {
            it++;
            t_earliest=it->instant;
        }else
            it++;
    }
    return t_earliest;
}
```

Función que se encarga de buscar en que instante se secuencia la actividad , dependiendo del instante en que han finalizado sus predecesoras y las unidades de recursos que quedan disponibles para los días siguientes, de forma que si no hay suficientes recursos para que la actividad pueda finalizar, se tendrá que retrasar su fecha de inicio.

PARÁMETROS DE ENTRADA

- Lista de las disponibilidades de recursos
- Actividad a secuenciar

PARÁMETROS DE SALIDA

- Instante en que se secuenciará la actividad

PASOS

- 1) Entre las actividades predecesoras obtener cual termina más tarde.
- 2) Comprobar que desde el instante obtenido en el paso 1 hasta su instante de inicio + la duración, hay suficientes recursos para realizar la actividad. Si hay algún día que no puede, se actualiza el instante en que comenzaría y se vuelve a comprobar si hay suficientes recursos

ESTRUCTURAS USADAS:

- Iterator: debido a que la estructura set no tiene métodos para acceder a sus elementos internos, a diferencia de los vectores, hemos de usar un objeto iterator para recorrer sus elementos de forma eficiente, que podrá recorrer sus elementos.

4.4.3.1.3 PASO 2.D: comprobar si las predecesoras han sido ya secuenciadas

```

bool all_predecesors_examined(Activity* act, std::vector<Activity*> examined)
{
    int j;

    /*Si es una actividad de inicio no hay problema*/
    if(act->predecesorsInitial())
        return true;

    /*Examinar todas las predecesoras y comprobar que estan vistas*/
    for(int i=0; i<act->getList_Activities_pred()->size();i++)
    {
        /*Esta ya examinada?*/
        for(j=0; j<examined.size();j++)
        {
            if(act->getList_Activities_pred()->at(i)->activity->getName()==examined.at(j)->getName())
                break;
        }
        /*Alguna no se ha examinado aun*/
        if(j==examined.size())
            return false;
    }
    return true;
}

```

Función que comprueba que todas las predecesoras de una actividad han sido secuenciadas, para ello comprueba cada una de las predecesoras, si están en examinadas. Si alguna no está, termina e informa del problema.

PARÁMETROS DE ENTRADA

- Actividad.
- Lista de actividades que ya han sido secuenciadas.

PARAEMTROS DE SALIDA

- True, todas las predecesoras están contenidas en examined.
- False, hay alguna predecesora que no ha sido secuenciada aun.

4.4.3.1.4 PASO 2.E: actualizar disponibilidad

```
disp_instant createDisponibility(int t, Activity * act, set<disp_instant, CompareDisponibilities>::iterator disponibility )
{
    disp_resource resource_in_t;
    disp_instant resources_in_t;
    Activity::resource * rec;

    /*Para todos los recursos, añadirle a las unidades del instante anterior, las unidades que devolvería esta actividad*/
    for(int i=0; i<disponibility->resources.size();i++)
    {
        rec=act->getResource(disponibility->resources.at(i).resource->getName());
        resource_in_t.resource=disponibility->resources.at(i).resource;

        if(rec!=NULL)           // la actividad consumía este recurso
            resource_in_t.units_disp=rec->units_asig+ disponibility->resources.at(i).units_disp;
        else                     //la actividad no consumía el recurso
            resource_in_t.units_disp=disponibility->resources.at(i).units_disp;
        resources_in_t.resources.push_back(resource_in_t);
    }
    resources_in_t.instant=t;
    return resources_in_t;
}
```

```
void update_disponibility(std::set<disp_instant, CompareDisponibilities> *disponibility, Activity *act, int t)
{
    int i=0;
    Activity::resource *rec;
    set<disp_instant, CompareDisponibilities>::iterator it=disponibility->begin();

    /*Dentro de la lista de disponibilidades, situarse en el momento que empezaría a consumir recursos*/
    while(disponibility->end() != it && it->instant < t) it++;

    /*Consumir los recursos de los diferentes instantes de disponibilidad, hasta que termine la actividad*/
    while(disponibility->end() != it && it->instant < t+act->getTNormal())
    {
        /*Actualizar disponibilidades de todos los recursos*/
        for(int k=0; k<it->resources.size();k++)
        {
            rec=act->getResource(it->resources.at(k).resource->getName());
            if(rec!=NULL)           //la actividad no tiene asignado ese recurso
                it->resources.at(k).units_disp-=rec->units_asig;
        }
        it++;
    }
    it--;

    /*Si el tiempo que dura la actividad va más allá de los instantes de disponibilidad guardados, añadir una nueva disponibilidad, que será
     el resultado de devolver a la disponibilidad anterior los recursos consumidos*/
    if(it->instant != t+act->getTNormal()) disponibility->insert(createDisponibility(t+act->getTNormal(), act, it));
}
```

Función que actualiza la lista de disponibilidades, decrementando el número de recursos disponibles desde el comienzo de la actividad hasta su fin. En caso de que la actividad dure más que el último instante de disponibilidad guardado, se añadirá una nueva disponibilidad, resultado de incrementarle a la última disponibilidad los recursos que se devolverían, cosa que hace la función createDisponibility, y la añade a la lista de disponibilidades, que se ordenaría crecientemente según el instante de disponibilidad.

PARÁMETROS DE ENTRADA

- Lista de disponibilidades.
- Actividad.
- Instante en que se secuencia la actividad.

ESTRUCTURAS USADAS:

- Iterator: debido a que la estructura set no tiene métodos para acceder a sus elementos internos, a diferencia de los vectores, hemos de usar un objeto iterator para recorrer sus elementos de forma eficiente, que podrá recorrer sus elementos.

4.4.3.2 ESQUEMA PARALELO

```
vector<processing_activity> insert_proces(vector<procesing_activity> procesing, procesing_activity act ){
    int i;

    /*Ordena las actividades procesandose crecientemente segun su instante de finalizacion de proceso */
    if(procesing.size()==0){
        procesing.push_back(act);
        return procesing;
    }

    for(i=0; i<procesing.size();i++){
        if(act.instant <procesing.at(i).instant){
            procesing.insert(procesing.begin()+i,act); //inserta el proceso en la posicion i
            return procesing;
        }
    }
    procesing.push_back(act);
    return procesing;
}
```

```
float Project:: limitationResourcesParalel(int rule)
{
    std:: vector<procesing_activity> procesing;
    disp_instant disponibility;
    std:: deque<Activity*> selected;
    std::vector<Activity*> examined;
    int t=0;
    int j=0;
    disp_resource resource_in_t;
    Activity * act;
    Activity::resource * rec;
    procesing_activity procesing_act, act_proces_fin;

    if(this->resources->size()==0)
    {
        showErrorInfo(E_ALGORITHM_NOT_EXIST_RESOURCES, " limitacion de recursos en paralelo");
        return INF;
    }

    /*Inicializacion*/
    mode_debug("\n\n*****PARALELO*****\n\n");

    /*Disponibilidad inicial, t=0*/
    for(int i=0; i<this->resources->size();i++)
    {
        resource_in_t.resource=resources->at(i);
        resource_in_t.units_disp=resources->at(i)->getUnitsMax();
        disponibility.resources.push_back(resource_in_t);
    }
    disponibility.instant=0;

    /*Actividades elegibles son las actividades de inicio*/
    for(int i=0; i< begin->getList_Activities_suc()->size();i++) selected=insert(selected, begin->getList_Activities_suc()->at(i)->activity, rule);

    /*Secuenciar actividades*/
    while(procesing.size()!=0 | t==0)
    {
        /*Hay alguna actividad procesandose?*/
        if(procesing.size() !=0)
        {
            act_proces_fin=procesing.front(); // obtiene la actividad que termina antes de procesarse
            t=act_proces_fin.instant;
        }
        mode_debug("\n\nITERACION ____T=" +num_to_str(t)+ " _____\n");

        /*actualizar el procesamiento, todas borrar las actividades que terminan en t y añadirlas a vistas*/
        while(procesing.size()>0&&act_proces_fin.instant==t)
        {
            mode_debug("La actividad "+act_proces_fin.act->getName()+" termina de procesarse\n");

            /*Añadirlas a vistas*/
            examined.push_back(act_proces_fin.act);
            procesing.erase(procesing.begin());

            /*Restaurar la disponibilidad*/
            for(int k=0; k<disponibility.resources.size();k++)
            {
                rec=act_proces_fin.act->getResource(disponibility.resources.at(k).resource->getName());
            }
        }
    }
}
```

```

        if(rec!=NULL) //no tiene asignado ese recurso
            disponibilidad.resources.at(k).units_disp+=rec->units_asig;
    }

    /*actualizar elegibles con las sucesoras de la actividad secuenciada, solo si todas las predecesoras de cada
    sucesora han sido vistas*/
    for(int i=0; < act_proces_fin.act->getList_Activities_suc()->size();i++)
        if(all_predecesors_examined(act_proces_fin.act->getList_Activities_suc()->at(i)->activity, examined)&&
           !act_proces_fin.act->sucesorIsEnd())
            selected=insert(selected,act_proces_fin.act->getList_Activities_suc()->at(i)->activity, rule);

    /*Siguiente procesandose*/
    if(procesing.size()>0)    act_proces_fin=procesing.front();
}

mode_debug("      ",disponibilidad.resources,t,"\\n");
mode_debug("      elegir={",selected,"}\\n");

/*Comprobar que actividades elegibles se pueden secuenciar*/
j=0;
while(!selected.empty()&&j<selected.size())
{
    /*Hay suficientes recursos para secuenciar esa actividad?*/
    if(selected.at(j)->isEnoughResource(disponibilidad))
    {
        /*Pasa a procesarse*/
        act=selected.at(j);
        procesing_act.act=act;
        procesing_act.instant=t+act->getTNormal();
        procesing=insert_proces(procesing,procesing_act);
        act->setTSecuenciacion(t);
        selected.erase(selected.begin()+j);

        mode_debug("      Se secuencia "+act->getName()+" en "+num_to_str(t)+" y termina en "
                  "+num_to_str(procesing_act.instant)+"\\n");

        /*Consumo los recursos disponibles*/
        for(int k=0; k<disponibilidad.resources.size();k++)
        {
            rec=act->getResource(disponibilidad.resources.at(k).resource->getName());
            if(rec!=NULL) //no tiene asignacion de ese recurso
                disponibilidad.resources.at(k).units_disp-=rec->units_asig;
        }
        j++;
    }
    mode_debug("      ",disponibilidad.resources, t,"\\n");
}
return t;
}

```

Aplica el esquema paralelo para realizar la secuenciación de actividades.

PARÁMETROS D E ENTRADA

- Regla de prioridad para establecer el ordenamiento de la lista de actividades elegibles.

PARÁMETROS DE SALIDA

- Nuevo duración del proyecto

PASOS

- 1) Inicialización
 - f. Inicializar la disponibilidad de recursos al máximo permitido por cada recurso.
 - g. Añadir en la lista de actividades elegibles, las actividades que son de inicio. Cuando se insertan en esta lista, se ordenan según la regla de prioridad usando la función insertar, que realizan la ordenación mediante un algoritmo de divide y vencerás con coste O(log n).
- 2) Mientras haya actividades procesándose o sea el primer instante

- h. Seleccionar la actividad más temprana que termina de procesarse, que al ser un vector ordenador crecientemente por instante de finalización, se encuentra en la cabeza de la lista. Actualizaríamos el instante de tiempo de finalización de esa actividad.
- i. Para esta actividad y todas las que terminen en ese momento, quitarlas de procesándose, añadirlas a vistas, y restauramos los unidades que consumía de disponibilidad.
- j. Actualizar la lista de elegibles de forma que se añaden las predecesoras de la actividad secuenciada, solo si todas las predecesoras de cada sucesora han terminado de procesarse. Esta lista se ordenaría según la regla de prioridad , usando la función insertar que ordinaria la lista con el método divide y vencerás, con un coste de $O(\log n)$.
- k. Seleccionaríamos de la lista de actividades elegibles, cuales tienes suficientes recursos para secuenciarse, las secuenciamos y consumimos los recursos de disponibilidad.

ESTRUCTURAS USADAS

- Vector: en este algoritmo usamos un vector de caminos, como hemos explicado anteriormente. La razón de usar un vector es porque estas estructuras son más eficientes en el acceso a sus elementos, que otras estructuras, y la mayor parte de operaciones que hacemos son de acceso a sus elementos, y muy pocas operaciones de inserción.
- Deque: usamos esta estructura porque nos interesa recrear el comportamiento de una pila ordenada, donde la cabeza contendrá siempre el máximo. Asimismo cuando eliminemos o insertemos un elemento, al poder acceder tanto a los elementos del principio como del final, podremos eliminar el que está en la cabeza, al igual que las pilas. La razón por la que hemos usado esta estructura y no una pila en sí, es porque con esta estructura podemos además de insertar elementos en cualquier posición, no solo insertarlos en la cabeza, y así podremos ordenar su contenido como queramos. En verdad, usamos esta estructura por sus métodos de desapilar, que nos resultan muy útiles, y su gran eficiencia la hora de insertar y eliminar elementos, que al tratarse de la lista de actividades seleccionadas se producen muchas operaciones de inserción y borrado. Otro aspecto importante es que esta estructura nos permite tener duplicados, no como set.

COSTE TEMPORAL $< O(n)$

4.4.3.3 ESQUEMA MULTIPASADA

```

float Project::HBRPMultiPasada()
{
    float best=INF;
    int best_rule=MIN_LFT_FIFO;
    int best_eschema=_SERIE;
    float temporal;

    if(this->resources==NULL)
    {
        showErrorInfo(E_ALGORITHM_NOT_EXIST_RESOURCES, "HBRP multi pasada");
        return INF;
    }

    mode_debug("##### MULTI PASADA #####\n\n");

    /*Para cada regla*/
    for(int rule=MIN_LFT_FIFO; rule<=MAX_NSUC_FIFO;rule++)
    {
        /*Resultado con el esquema paralelo*/
        temporal=this->limitationResourcesParalel(rule);
        if(best>temporal) //es best, update
        {
            best=temporal;
            best_rule=rule;
            best_eschema=_PARALLEL;
        }

        /*Resultado con el esquema secuencial*/
        temporal=this->limitationResourcesSerie(rule);
        if(best>temporal)// es mejor, actualizar
        {
            best=temporal;
            best_rule=rule;
            best_eschema=_SERIE;
        }
    }
    mode_debug("Mejor esquema y mejor regla con coste "+num_to_str(best)+" es la siguiente\n");

    if(best==INF)  return INF;           //no se ha podido aplicar el algoritmo

    /*Una vez se sepa cual es el mejor resultado, aplicarlo*/
    if(best_eschema==_SERIE)
        return limitationResourcesSerie(best_rule);
    else
        return limitationResourcesParalel(best_rule);
}

```

Método que aplica varias reglas de prioridad para ordenar las actividades elegibles, y para cada regla aplica el esquema serie y el esquema paralelo, de forma que se queda con el mejor esquema y reglas, y luego los aplica para fijar los resultados

PARÁMETROS DE SALIDA

- Duración nueva del proyecto

COSTE TEMPORAL: O(número de reglas)

4.4.4 Atraso/Adelanto Asignación de recursos

```

bool Project::retrasoAdelantoActivities(int eschema_retraso, int eschema_adelanto){

    float max1, max2;
    deque<Activity*> activities_ordered;

    do{
        mode_debug("\n\n##### ADELANTO/ATRASO#####\n\n");

        /*Ordenar actividades en orden decreciente de finalizacion*/
        for(int i=0; i<this->activities.size();i++)
            activities_ordered=insert(activities_ordered,activities.at(i), MIN_EFT_FIFO);
        reverse(activities_ordered.begin(), activities_ordered.end()); //orden decreciente

        /*RETRASO: aplicar esquema con regla*/
        mode_debug("||||| RETRASO|||||\n");
        mode_debug("elegir=",activities_ordered,")\n");

        if(eschema_retraso==_SERIE)
            max1=limitationResourcesSerie( activities_ordered, "retraso");
        else
            max1=limitationResourcesParalel( activities_ordered,"retraso");

        if(max1==INF)
            return false;//no se ha podido aplicar el algoritmo

        /*Traducir tiempos de secuenciacion*/
        for(int i=0; i<this->activities.size();i++)
            activities.at(i)->setTMin(max1-activities.at(i)->getTSecuenciacion()-activities.at(i)->getTNormal());
        end->setTMin(max1);

        /*Comprobar que el algoritmo lo resuelva bien*/
        if(!validateDependences())
            exit(-1);
        if(!validateLimitationMaxUnitsAllocated())
            exit(-1);

        /*Ordenar actividades en orden creciente de inicio*/
        activities_ordered.clear();
        for(int i=0; i<this->activities.size();i++)
            activities_ordered=insert(activities_ordered,activities.at(i), MIN_EST_FIFO);

        /*ADELANTO: aplicar esquema con regla*/
        mode_debug("||||| ADELANTO|||||\n");
        mode_debug("elegir=",activities_ordered,")\n");

        if(eschema_adelanto==_SERIE)
            max2=limitationResourcesSerie( activities_ordered, "adelanto");
        else
            max2=limitationResourcesParalel( activities_ordered,"adelanto");

        if(max2==INF)
            return false;//no se ha podido aplicar el algoritmo

        /*actualizar t.min*/
        for(int i=0; i<activities.size();i++)
            activities.at(i)->setTMin(activities.at(i)->getTSecuenciacion());
        end->setTMin(max2);

        /*Actualizar t.max y holguras*/
        updateTMax();

        /*Comprobar si el algoritmo lo resuelve bien*/
        if(!validateDependences())
            exit(-1);
        if(!validateLimitationMaxUnitsAllocated())
            exit(-1);

    }
    /*Estos pasos se aplicaran mientras no se converja a una solucion*/
    while(max1-max2!=0);

    return true;
}

```

Método que implementa la compactación de actividades, primero realizando el retraso y luego realizando el adelanto, y se deberán realizar estos 2 pasos mientras las duraciones que devuelvan ambos pasos sean diferentes, y así se converge a una solución.

PARÁMETROS DE ENTRADA

- Esquema para el paso de retraso, que puede ser _SERIE o _PARALEL
- Esquema para el paso de adelanto, que puede ser _SERIE o _PARALEL

PARAMETROS DE SALIDA

- Devuelve si se ha podido aplicar el algoritmo o no.

PASOS

- 1) Crear la lista de elegibles ordenando las actividades por el instante de finalización más tardío obtenido en la limitación. El coste de realizar esta ordenación es $O(\log n)$ con divide y vencerás.
- 2) Realizar el paso de retraso con el esquema escogido.
- 3) Como los instantes obtenidos están en función de que la última actividad a realizarse es en realidad la primera, se traducen estos tiempos.
- 4) Validar que se cumplen las dependencias entre actividades y la limitación de recursos.
- 5) Ordenar las actividades según el instante de inicio más temprano.
- 6) Aplicar el paso de adelanto con el esquema elegido.
- 7) Traspasar los resultados a t_{min} .
- 8) Comprobar que se cumplen las dependencias entre actividades
- 9) Comprobar que se cumple la limitación de recursos.
- 10) Calcular las holguras y t_{max} .
- 11) Comprobar si ya no se puede seguir mejorando la duración. Si la duración obtenida con el adelanto y el retraso es igual, terminamos, sino seguimos aplicando ambos pasos.

ESTRUCTURAS UTILIZADAS

- Deque: usamos esta estructura porque nos interesa recrear el comportamiento de una pila ordenada, donde la cabeza contendrá siempre el máximo. Asimismo cuando eliminemos o insertemos un elemento, al poder acceder tanto a los elementos del principio como del final, podremos eliminar el que está en la cabeza, al igual que las pilas. La razón por la que hemos usado esta estructura y no una pila en sí, es porque con esta estructura podemos además de insertar elementos en cualquier posición, no solo insertarlos en la cabeza, y así podremos ordenar su contenido como queramos. En verdad, usamos esta estructura por sus métodos de desapilar, que nos resultan muy útiles, y su gran eficiencia la hora de insertar y eliminar elementos, ya que se producen muchas operaciones de inserción y borrado. Otro aspecto importante es que esta estructura nos permite tener duplicados, no como set.

4.4.4.1 ESQUEMA SERIE

```

float Project::limitationResourcesSerie( deque<Activity*> selected, string step)
{
    int n_act_to_process=activities.size();
    std::set<disp_instant, CompareDisponibilities> *disponibility=new set<disp_instant, CompareDisponibilities>;
    std::vector<Activity*> examined;
    int t;
    int t_max=0;
    disp_resource resource_in_t;
    disp_instant resources_in_t;
    Activity * act;

    if(this->resources->size()==0)
    {
        showErrorInfo(E_ALGORITHM_NOT_EXIST_RESOURCES, " proceso de "+step+" de actividades en serie");
        return INF;
    }

    /*INICIALIZACION/
    mode_debug("\n\n*****SERIE*****\n\n");

    /*Disponibilidad inicial, t=0*/
    for(int i=0; i<this->resources->size();i++)
    {
        resource_in_t.resource=resources->at(i);
        resource_in_t.units_disp=resources->at(i)->getUnitsMax();
        resources_in_t.resources.push_back(resource_in_t);
    }
    resources_in_t.instant=0;
    disponibility->insert(resources_in_t);

    /*Se examinan todas las actividades*/
    while(n_act_to_process>0)
    {
        mode_debug("\n\nITERACION _____ \n");
        mode_debug("      elegir={"selected,"}\n");

        /*Actividad a secuenciar*/
        act=selected.front();
        selected.pop_front();

        /*calcular instante en que se secuenciará*/
        if(step.compare("retraso")==0)
            t= get_sequencing_instant_suc(disponibility, act); // comprueba las sucesoras
        else
            t= get_sequencing_instant(disponibility, act); //comprueba las predecesoras

        /*Secuenciar*/
        act->setTSecuenciacion(t);
        examined.push_back(act);
        mode_debug("      La actividad "+act->getName()+" se secuencia en "+num_to_str(t)+"\n");

        /*Guardamos cual es el instante de fin del proyecto, que será cuando finalice la actividad que termina más tarde*/
        if(t+act->getTNormal()>t_max)
            t_max=t+act->getTNormal();

        /*actualizar disponibilidad*/
        update_disponibility(disponibility, act, t);
        mode_debug("      ,disponibility,\n");

        n_act_to_process--;
    }

    delete(disponibility);
    return t_max;
}

```

Aplica el esquema serie para realizar la compactación de actividades. Según el paso que reciba, realizará adelanto o retraso; además, a diferencia del otro esquema serie explicado anteriormente, este se ha adaptado al hecho de no tener añadir las actividades elegibles, sino que su orden viene impuesto como parámetro de entrada.

PARÁMETROS DE ENTRADA

- Lista ordenada de las actividades elegibles
- Paso que indicará si es la pasada de retroceso o adelanto

PARÁMETROS DE SALIDA

- Nueva duración del proyecto

PASOS

- 1) Inicializar la lista de disponibilidades con las unidades de cada recurso que se tienen.
- 2) Para cada actividad se realizará una iteración.
 - a. Seleccionar la actividad más prioritaria, que al ser un vector ordenador por prioridades, se encuentra en la cabeza del vector.
 - b. Calcular el instante en que se secuencia esa actividad, su cálculo depende de si es el paso de adelanto o de retraso.
 - i. Si es de adelanto se realizará de la misma forma que habíamos visto.
 - ii. si es de retraso, se tendrá que adaptar a que ahora el recorrido es de sucesoras a predecesoras, y no al contrario como en adelanto.
 - c. Secuenciar la actividad y añadirla a la lista de vistas. Guardando el instante de finalización del proyecto, que será el instante en que termine la actividad que termina mas tarde.
 - d. Actualizar disponibilidad.

ESTRUCTURAS UTILIZADAS

- Vector: en este algoritmo usamos un vector de caminos, como hemos explicado anteriormente. La razón de usar un vector es porque estas estructuras son más eficientes en el acceso a sus elementos, que otras estructuras, y la mayor parte de operaciones que hacemos son de acceso a sus elementos, y muy pocas operaciones de inserción.
- Set: utilizamos esta estructura, ya que nos permite asignar funciones para ordenar como queramos los elementos que la componen, cosa que la dota de una gran eficiencia, aunque es muy ineficiente recorrer sus elementos. Aun así , debido al hecho de que no pueden haber instantes repetidos, podremos usar esta estructura que no los permite.

COSTE TEMPORAL O(n)

4.4.4.1.1 PASO 2.B.II

```

int get_sequencing_instant_suc(std::set<disp_instant, CompareDisponibilities> *disponibility, Activity * act)
{
    int t_earliest=0;
    set<disp_instant, CompareDisponibilities>::iterator it=disponibility->begin();

    /*Encontrar el instante mas pronto que puede empezar, dependiendo de cuando terminen sus sucesoras*/
    if(!act->sucesorsEnd())
    {
        for(int suc=0;suc<act->getList_Activities_suc()->size();suc++)
            if(act->getList_Activities_suc()->at(suc)->activity->getTSecuenciacion()+
               act->getList_Activities_suc()->at(suc)->activity->getTNormal()>t_earliest)
                t_earliest=act->getList_Activities_suc()->at(suc)->activity->getTSecuenciacion()+
                           act->getList_Activities_suc()->at(suc)->activity->getTNormal();

    }

    /*Situarse en ese instante*/
    while(disponibility->end()!=it&& it->instant<t_earliest) it++;

    while(disponibility->end()!=it&&it->instant< t_earliest+act->getTNormal() )
    {
        if(disponibility->end()==it) return t_earliest;

        if(!act->isEnoughResource(it))
        {
            it++;
            t_earliest=it->instant;
        }else
            it++;
    }
    return t_earliest;
}

```

Función que se encarga de buscar en que instante se secuencia la actividad , dependiendo del instante en que han finalizado sus sucesoras y las unidades de recursos que quedan disponibles para los días siguientes, de forma que si no hay suficientes recursos para que la actividad pueda finalizar, se tendrá que retrasar su fecha de inicio.

PARÁMETROS DE ENTRADA

- Lista de las disponibilidades de recursos
- Actividad a secuenciar

PARÁMETROS DE SALIDA

- Instante en que se secuenciará la actividad

PASOS

- 1) Entre las actividades sucesoras obtener cual termina más tarde.
- 2) Comprobar que desde el instante obtenido en el paso 1 hasta su instante de inicio + la duración, hay suficientes recursos para realizar la actividad. Si hay algún día que no puede, se actualiza el instante en que comenzaría y se vuelve a comprobar si hay suficientes recursos

ESTRUCTURAS USADAS:

- Iterator: debido a que la estructura set no tiene métodos para acceder a sus elementos internos, a diferencia de los vectores, hemos de usar un objeto iterator para recorrer sus elementos de forma eficiente, que podrá recorrer sus elementos.

4.4.4.2 ESQUEMA PARALELO

```

float Project::limitationResourcesParallel(deque<Activity*> selected, string step)
{
    std::vector<procesing_activity> procesing;
    disp_instant disponibility;
    std::vector<Activity*> examined;
    int t=0, j=0;
    float t_max=0;
    disp_resource resource_in_t;
    Activity * act;
    Activity::resource * rec;
    procesing_activity procesing_act, act_proces_fin;

    if(this->resources->size()==0)
    {
        showErrorInfo(E_ALGORITHM_NOT_EXIST_RESOURCES, " proceso de "+step+" de actividades en paralelo");
        return INF;
    }

    /*Inicializacion*/
    mode_debug("\n\n*****PARALELO*****\n\n");

    /*Disponibilidad inicial, t=0*/
    for(int i=0; i<this->resources->size();i++)
    {
        resource_in_t.resource=resources->at(i);
        resource_in_t.units_disp=resources->at(i)->getUnitsMax();
        disponibility.resources.push_back(resource_in_t);
    }
    disponibility.instant=0;

    /*Secuenciar actividades*/
    while(procesing.size()!=0 | | t==0)
    {
        mode_debug("\n\nITERACION ____T="+num_to_str(t)+" _____\n");

        /*Hay alguna actividad procesandose?*/
        if(procesing.size()!=0)
        {
            act_proces_fin=procesing.front();
            t=act_proces_fin.instant;
        }

        /*actualizar el procesamiento, todas borrar las actividades que terminan en t y añadirlas a vistas*/
        while(procesing.size()>0&&act_proces_fin.instant==t)
        {
            mode_debug("      La actividad "+act_proces_fin.act->getName()+" termina de procesarse\n");

            /*Añadirlas a vistas*/
            examined.push_back(act_proces_fin.act);
            procesing.erase(procesing.begin());

            /*Restaurar la disponibilidad*/
            for(int k=0; k<disponibility.resources.size();k++)
            {
                rec=act_proces_fin.act->getResource(disponibility.resources.at(k).resource->getName());
                if(rec!=NULL) //consume de ese recurso
                    disponibility.resources.at(k).units_disp+=rec->units_asig;
            }
            if(procesing.size()>0) act_proces_fin=procesing.front();
        }
        mode_debug("      ,disponibility.resources,t,""\n");
        mode_debug("      elegir={,selected,""}\n");

        /*Comprobar que actividades elegibles se pueden secuenciar*/
        j=0;
        while(!selected.empty()&&j<selected.size())
        {
            /*Hay suficientes recursos para secuenciar esa actividad?*/
            if(selected.at(j)->isEnoughResource(disponibility)&&dependenciesFinished(step,selected.at(j),examined))
            {
                /*Pasa a procesarse*/
                act=selected.at(j);
                procesing_act.act=act;
                procesing_act.instant=t+act->getTNormal();
                procesing=insert_proces(procesing,procesing_act);
                act->setTSecuenciacion(t);
                selected.erase(selected.begin()+j);

                mode_debug("      Se secuencia "+act->getName()+" en "+num_to_str(t)+" y termina en "+
            }
        }
    }
}

```

```

        num_to_str(procesing_act.instant)+"\n");

    /*Consumo los recursos disponibles*/
    for(int k=0; k<disponibility.resources.size();k++)
    {
        rec=act->getResource(disponibility.resources.at(k).resource->getName());
        if(rec!=NULL)// no esta asignado ese recurso
            disponibility.resources.at(k).units_disp-=rec->units_asig;
    }
    mode_debug("","disponibility.resources, t, "\n");
}
return t;
}

```

Aplica el esquema paralelo para realizar la compactación de actividades. Según el paso que reciba, realizará adelanto o retraso; además, a diferencia del otro esquema paralelo explicado anteriormente, este se ha adaptado al hecho de no tener que añadir las actividades elegibles, sino que su orden viene impuesto como parámetro de entrada.

PARÁMETROS D E ENTRADA

- Lista ordenada de las actividades elegibles
- Paso que indicará si es la pasada de retroceso o adelanto

PARÁMETROS DE SALIDA

- Nueva duración del proyecto

PASOS:

- 1) Inicializar la disponibilidad de recursos al máximo permitido por cada recurso
- 2) Mientras haya actividades procesándose o sea el primer instante
 - a. Seleccionar la actividad más temprana que termina de procesarse, al ser un vector ordenador crecientemente por el instante de finalización, se encuentra en la cabeza de la lista. Actualizaremos el instante de tiempo actual al de finalización de esa actividad.
 - b. Para esta actividad y todas las que terminen en ese momento, quitarlas de procesándose, añadirlas a vistas, y restauramos las unidades que consumía de disponibilidad.
 - c. Seleccionaríamos de la lista de actividades elegibles, cuales tienes suficientes recursos para secuenciarse, y cuyas actividades dependientes hayan sido examinadas. Esas dependencias dependiendo de si el paso es retraso o adelanto, se comprueba las sucesoras o las predecesoras.
 - d. Las secuenciamos y consumimos los recursos de disponibilidad.

ESTRUCTURAS UTILIZADAS

- Vector: en este algoritmo usamos un vector de caminos, como hemos explicado anteriormente. La razón de usar un vector es porque estas estructuras son más eficientes en el acceso a sus elementos, que otras estructuras, y la mayor parte de

operaciones que hacemos son de acceso a sus elementos, y muy pocas operaciones de inserción.

COSTE TEMPORAL $< O(n)$

4.4.4.2.1 PASO 2.C: comprobar dependencias

```

bool dependenciesFinished(string step, Activity * act, std::vector<Activity *> examined)
{
    /*Comprobar si ya estan examinadas las dependencias*/
    if(step.compare("retraso")==0)
    {
        /*En retraso tenemos que ver el proyecto del revés por lo que intenresa comprobar las sucesoras*/
        for(int i=0; i< act->getList_Activities_suc()->size();i++)
            if(!act->getList_Activities_suc()->at(i)->activity->isContained(examined)&&!act->sucesorIsEnd())
                return false;
    }
    else
    {
        /*En adelanto atenderemos a lasp rededesoras*/
        for(int i=0; i< act->getList_Activities_pred()->size();i++)
            if(act->getList_Activities_pred()->at(i)->activity->isContained(examined)&&!act->predecesorIsInitial())
                return false;
    }
    return true;
}

```

Función que comprueba que las dependencias están resueltas, para ello si el paso es retraso, el proyecto se debe de ver al revés, de forma que las sucesoras serian las predecesoras y se valida si las sucesoras han terminado o no, comprobando que todas se encuentran en la lista de examinadas. Si el paso es de adelanto, el proyecto se ve tal cual, y comprobamos que todas las predecesoras han sido examinadas.

PARÁMETROS DE ENTRADA

- Paso, que puede ser retraso o adelanto
- Actividad
- Lista de actividades examinadas o terminadas

PARÁMETROS DE SALIDA

- True, todas las dependencias han sido completadas.
- False, alguna dependencia aun no se ha completado.

4.4.4.3 Validar Dependencias

```
bool Activity::validateDependences()
{
    bool result=true;

    /*Comprueba si las dependencias se cumplen*/
    for(int i=0; i< this->List_Activities_suc.size();i++)

        /*La sucesoras no pueden empezar antes de que terminen las predecesoras*/
        if(t_min+t_normal>List_Activities_suc.at(i)->activity->t_min)
        {
            showErrorInfo(E_DEPENDENCES, name+" y "+List_Activities_suc.at(i)->activity->name);
            result=false;
        }
    return result;
}
```

```
bool Project::validateDependences()
{
    bool result=true;

    /*Comprobar relaciones de precedencia entre actividades, solo en que una
    relacion no se cumpla, todo el proyecto no cumplira con esta validacion */
    for(int i=0; i<activities.size();i++)
        result*=activities.at(i)->validateDependences(); // AND logica
    return result;
}
```

Los métodos de arriba se encargan de comprobar que las relaciones de precedencia entre actividades se cumplan. Para ello comprueba para todas las actividades cumplen que su instante de inicio no sea menor al instante de fin de su predecesora, es decir, que no empiece una sucesora antes de que cualquiera de sus predecesoras. Si alguna actividad un cumple esta restricción, la validación devolverá error. Para ello utilizamos una AND lógica, según la cual si alguna actividad no lo cumple, el resultado será falso, da igual lo que las otras actividades devuelven de su análisis.

4.4.4.4 Validar Limitación de recursos

```

bool Project::validateLimitationMaxUnitsAllocated()
{
    int tiempo=end->getTMax();
    int n_resources=0;
    Activity *act;
    Activity::resource * rec_asig;

    /*Comprueba que se cumple la limitacion para cada recurso*/
    for(int rec=0; rec<this->resources->size(); rec++)
    {

        /*Recorremos cada dia*/
        for(int dia=0; dia<tiempo;dia++)
        {
            n_resources=0;

            /*Selecciona las actividades que se estan realizando ese dia*/
            for(int i=0; i<activities.size();i++)
            {
                act=activities.at(i);
                if(act->getTMin()<=dia&&act->getTMin()+act->getTNormal()-1>=dia)
                {
                    rec_asig=act->getResource( resources->at(rec)->getName());
                    if(rec_asig!=NULL)
                        n_resources+=rec_asig->units_asig;
                }
            }
            /*Ese dia se supera el maximo de recursos permitidos?*/
            if(n_resources>resources->at(rec)->getUnitsMax())
                return false;
        }
    }
    return true;
}

```

Este método comprueba que las asignaciones de cada recurso al día, no superen las unidades máximas del recurso por día. Para ello comprueba para cada recurso del sistema, las actividades que se usan ese recursos cada uno de los días que dura el proyecto y si el total de recursos que usan esas actividades al día, es mayor al máximo que tiene, es que el algoritmo no ha funcionado bien, i devolverá false.

4.4.5 Nivelación de recursos

Existen dos opciones de funcionamiento para este algoritmo.

El primero modo nivela todos los recursos, para ello llama a la función levelResources, que es la función que aplica propiamente el algoritmo de nivelación. Esta función debe recibir los recursos a nivelar, que en este caso son todos.

```
bool Project::levelAllResources()
{
    /*Para nivelar todos los recursos le pasamos a la funcion de nivelacion todos los recursos del proyecto.*/
    return levelResources( resources);
}
```

La otra forma de funcionamiento es indicar un subconjunto de recursos, que son los que se quiere nivelar, para ello esta función recibe un conjunto de nombres de recursos a nivelar, que serán los que recibirá el algoritmo principal.

```
bool Project::levelResources(std::vector<std::string> names_rec)
{
    std::vector<Resource *> *resources= new std::vector<Resource *>;
    Resource * resource;

    /*Transformamos los nombres de los recursos en recursos y los guardamos*/
    for(int i=0; i<names_rec.size();i++)
    {
        resource=this->getResource(names_rec.at(i));

        if(resource==NULL)
        {
            showErrorInfo(E_RESOURCE_NOT_EXIST, names_rec.at(i));
            return false;
        }
        else
            resources->push_back(resource);
    }

    /*Pasamos a la funcion los recursos que queremos nivelar */
    return levelResources(resources);
    delete(resources);
}
```

El siguiente método es el que implementa el algoritmo de nivelación de recursos en si.

```

bool Project::levelResources(std::vector<Resource*> *resources_level)
{
    Activity*act;
    double charges_origin=0;
    double charges_calculated=0;
    std::deque<Activity*> ordered_activities;
    int max_instant;
    int t_best_begin;
    int instant;

    if(resources_level->size()==0)
    {
        showErrorInfo(E_ALGORITHM_NOT_EXIST_RESOURCES, " nivelacion de recursos");
        return false;
    }

    /*Ordenar las actividades en orden decreciente segun el instante de finalizacion mas temprano*/
    for(int i=0; i<activities.size();i++)
        ordered_activities=insert(ordered_activities, activities.at(i),MIN_EFT_FIFO);
    reverse(ordered_activities.begin(),ordered_activities.end());

    mode_debug("***** NIVELACION RECURSOS *****\n\n");
    mode_debug("Orden de las actividades a mover: ",ordered_activities," \n");

    /*Carga disposicion original*/
    for(int i=0; i<resources_level->size();i++)
        charges_origin+=calculCostResource(resources_level->at(i)->getName());
    mode_debug("\nCVoriginal = "+num_to_str(charges_origin)+"\n\n");

    /*Calcula el instante de inicio de las actividades con menor carga de recursos*/
    for(int i=0; i<ordered_activities.size();i++)
    {
        act=ordered_activities.at(i);
        t_best_begin=act->getTMin();           //parte de intante de inicio mas temprano

        /*Modificar el dia de comienzo de las activities no criticas, consumiendo la holgura libre*/
        instant=act->getTMin()+1;
        max_instant=act->getTMin()+act->getHFree();
        mode_debug(act->getName()+" TMin Original="+num_to_str(instant)+" H.Libre="+num_to_str(act->getHFree())+":\n\n");

        /*Mientras haya holgura libre*/
        while(instant<=max_instant)
        {
            /*Inicializa*/
            act->setTMin(instant);
            charges_calculated=0;

            /*Calculamos la carga para cada recurso con el CV*/
            for(int r=0; r<resources_level->size();r++)
                charges_calculated+=calculCostResource(resources_level->at(r)->getName());
            mode_debug("      Movemos la actividad "+act->getName()+" al instante "+num_to_str(instant)+"
                      con un CV="+num_to_str(charges_calculated)+"\n");

            /*La suma total de las cargas de los recursos mejora respecto a la original?*/
            if(charges_calculated<=charges_origin)
            {
                charges_origin=charges_calculated;
                t_best_begin=instant;
                mode_debug("      La nueva disposicion mejora la anterior\n");
            }
            instant++;
        }
        /*Asignamos la mejora*/
        act->setTMin(t_best_begin);
        mode_debug("      "+act->getName()+" se realiza en "+num_to_str(t_best_begin)+"\n-----\n");
    }
    /*Actualizar t.max y holguras*/
    updateTMax();

    /*Validar si despues de aplicar el algoritmo, se siguen cumpliendo las dependencias y la limitacion de recursos*/
    if(!validateDependences())
        exit(-1);
    if(!validateLimitationMaxUnitsAllocated())
        exit(-1);
    return true;
}

```

Esta es la función principal que nivela los recursos indicados, si se han seleccionado recursos ilimitados se hará una nivelación de recursos ilimitada, pero si los recursos tienen unas unidades máximas, se hará una nivelación de recurso limitada. Se realizará una nivelación en que solo se mejora si se reduce la suma de CV de los recursos, respecto a la original o si es igual, porque nos interesa tener la actividad lo más a la derecha posible.

PARÁMETROS DE ENTRADA

- Vector de recursos a nivelar

PARÁMETROS DE SALIDA

- Si el algoritmo es posible aplicar

PASOS

- 1) Ordenar las actividades decrecientemente según su instante de fin más temprano.
- 2) Calcular la carga (sumatoria de cv) para la planificación original.
- 3) Para todas las actividades no críticas modificaremos su día de comienzo hasta consumir toda su holgura.
 - a. Modificar su instante de comienzo y calcular las cargas para cada recurso (coeficiente variación) y sumarlas.
 - b. Si la carga calculada es menor o igual que la carga original, entonces esta solución es mejor y guardamos la nueva carga como la original y el instante en que se da.
- 4) Agregar a las actividades cuál ha sido el mejor instante.
- 5) Actualizar t_{max} y holguras según los nuevos t_{min} calculados.
- 6) Comprobar que las dependencias y la limitación de recursos se siguen cumpliendo.

ESTRUCTURAS UTILIZADAS

- Deque: usamos esta estructura porque nos interesa recrear el comportamiento de una pila ordenada, donde la cabeza contendrá siempre el máximo. Asimismo cuando eliminemos o insertemos un elemento, al poder acceder tanto a los elementos del principio como del final, podremos eliminar el que está en la cabeza, al igual que las pilas. La razón por la que hemos usado esta estructura y no una pila en sí, es porque con esta estructura podemos además de insertar elementos en cualquier posición, no solo insertarlos en la cabeza, y así podremos ordenar su contenido como queramos. En verdad, usamos esta estructura por sus métodos de desapilar, que nos resultan muy útiles, y su gran eficiencia la hora de insertar y eliminar elementos, ya que se producen muchas operaciones de inserción y borrado. Otro aspecto importante es que esta estructura nos permite tener duplicados, no como set.

4.4.5.1 PASO 2 Y 3.A: calcular coeficiente de variación

```

double Project :: calculCostResource( string name_resource)
{
    int time=end->getTMax();
    double charge=0;
    double cost_day=0;
    Activity::resource * rec_asig;
    double average=0;
    double s=0;
    double t_min;
    int units_max=this->getResource(name_resource)->getUnitsMax();

    /*CALCULAR LA MEDIA PARA ESE RECURSO*/

    /*Recorremos cada dia del proyecto*/
    for(int day=0; day<time;day++)
    {
        charge=0;

        /* Selecciona las actividades que se estan realizando ese dia*/
        for(int i=0; i<activities.size();i++)
        {
            t_min=activities.at(i)->getTMin();
            if(t_min<=day&&t_min+activities.at(i)->getTNormal()-1>=day)
            {
                rec_asig=activities.at(i)->getResource(name_resource);
                if(rec_asig!=NULL)
                {
                    average+=rec_asig->units_asig;
                    charge+=rec_asig->units_asig;
                }
            }
        }
        /* Prueba factibilidad, se cumpliria la limitacion de recursos?*/
        if(charge>units_max)
        {
            mode_debug("      Se supera la limitacion de recursos\n");
            return INF; // como no es factible, coste infinito
        }
    }

    /* Media */
    average=average/time;

    /*CALCULO COEFICIENTE VARIACION */

    /*Recorremos cada dia del proyecto*/
    for(int day=0; day<time;day++)
    {
        /* S */
        cost_day=0;

        /* Selecciona las actividades que se estan realizando ese dia*/
        for(int i=0; i<activities.size();i++)
        {
            t_min=activities.at(i)->getTMin();
            if(t_min<=day&&t_min+activities.at(i)->getTNormal()-1>=day)
            {
                rec_asig=activities.at(i)->getResource(name_resource);
                if(rec_asig!=NULL) cost_day+=rec_asig->units_asig;
            }
        }
        s+=(cost_day-average)*(cost_day-average);
    }
    s=s/time;
    s=sqrt(s);
    return (s/average);
}

```

Función que calcula el coeficiente de variación de un recurso.

PARÁMETROS DE ENTRADA

- Recurso sobre el cual calcular el coeficiente de variación

PARÁMETROS DE SALIDA

- Coeficiente de variación

PASOS

- 1) Calcular la media de uso de ese recurso durante todos los días del proyecto
 - a. Si el uso del recurso es mayor al número de unidades máximas, es una solución no factible y por tanto no es buena, por lo que devolverá un coste muy alto.
- 2) Calcular el uso de los recursos para cada día.
- 3) Obtener el sumatorio de la diferencia entre el coste por día y el coste medio, y lo elevamos al cuadrado.
- 4) Dividirlo por la duración del proyecto y sacar la raíz cuadrada, sobre esta, dividirla por la media.

4.5 ANALISIS

4.5.1 Flexibilidad

En este análisis estudiamos como afectaría el hecho de retrasar o adelante la actividad respecto a su fecha de comienzo más tardía. Para el caso de retrasar la actividad de su instante de inicio, tenemos el siguiente código que analizaría su resultado

```

string Project::anализDelayTMin(int days, string name_activity)
{
    int hlr;
    int n_resources=0;
    Activity * activity= getActivity( name_activity);
    Activity * act;
    Resource * rec;
    Activity::resource * rec_asig;
    string inform="";
    bool exceeded_limitation=false;

    /*ANALISIS FLEXIBILIDAD*/
    if(activity==NULL)
    {
        showErrorInfo(E_ACTIVITY_NOT_EXIST, name_activity);
        return "";
    }

    hlr=activity->getHLR();

    /*Valido?*/
    if(days<0)      return "Valor de dias no valido";

    /*Adelanta la actividad mas de instante de inicio del proyecto?*/
    if(days>activity->getTMin())  return "No se puede adelantar la actividad mas alla del instante de inicio de proyecto";

    /*Factible?*/
    if(days>hlr)     return "Retraso no factible, aumentaria la duracion del proyecto";

    /*Como se ve afectada la limitacion?*/

    /*Para los nuevos instantes que se realizaria la actividad*/
    for(int d=activity->getTMin()+days; d<=activity->getTMin()+days+activity->getTNormal();d++)
    {
        /*Comprobar todos los recursos*/
        for(int r=0; r<activity->getResources().size();r++)
        {
            rec=activity->getResources().at(r)->resource_asig;
            n_resources=0;

            /*Selecciona las actividades que se estan realizando ese dia, sin incluir la actividad a analizar*/
            for(int i=0; i<activities.size();i++)
            {
                act=activities.at(i);
                if(act->getTMin()<=d&&act->getTMin()>=d && act->getName()!=activity->getName())
                {
                    rec_asig=act->getResource( rec->getName());
                    if(rec_asig!=NULL)      n_resources+=rec_asig->units_asig;
                }
            }

            /*El consumo de recursos mas el consumo que haria esa actividad si se realizase en ese instante, supera el limite de recursos?*/
            if(n_resources+activity->getResource(rec->getName())->units_asig>rec->getUnitsMax())
            {
                /*Guardamos todos los análisis para cada recurso*/
                if(exceeded_limitation)  inform.append("La actividad se podria retrasar sin repercutir en la duracion del proyecto, pero:\n");
                inform.append("Excederia el maximo de recurso " + rec->getName() + " en " + num_to_str(n_resources+
activity->getResource(rec->getName())->units_asig-rec->getUnitsMax())+" unidades en el instante "+
num_to_str(d)+"\n");
                exceeded_limitation=true;
            }
        }
    }

    if(!exceeded_limitation)      return "Retraso factible, no se modificaria la duracion del proyecto y se seguiria cumpliendo la limitacion de recursos";
    else                          return inform;
}

```

PARÁMETROS DE ENTRADA

- Actividad a analizar
- Días en que se retrasará su comienzo

PARÁMETROS DE SALIDA

- String con el informe de su análisis

PASOS

- 1) Comprobamos si es un valor valido o coherente.
- 2) Si el número de días que se quiere retrasar es mayor al número de días que se podría retrasar la actividad, sin modificar la duración del proyecto (HLR), entonces no sería un cambio factible.
- 3) Si se puede retrasar sin afectar la duración del proyecto, estudiamos como afectaría este cambio a los recursos. De forma que se obtiene cual sería la asignación de recursos si moviésemos la actividad los días pertinentes.
- 4) Estudiamos para cada día que duraría la actividad desde su nuevo tiempo de comienzo, como afectaría a los recursos de los que hace uso. Si supera alguna limitación de algún recurso, informamos de ello junto con la cantidad que se excedería y guardaríamos este resultado.
- 5) Si no se ha excedido ninguna limitación, informaríamos de que se cumpliría con las limitaciones, sino informamos de cada recurso lo que se excede.

Para analizar el efecto de un adelanto, el código sería el siguiente:

```

string Project::analizeAnticipateTMin(int days, string name_activity)
{
    int hla;
    int n_resources=0;
    Activity * activity=getActivity( name_activity);
    Activity * act_pred, *act;
    Resource * rec;
    Activity:resource * rec_asig;
    string pred_problem="";
    string inform="";
    bool exceded_limitation=false;

    /*ANALISIS FLEXIBILIDAD*/
    if(activity==NULL)
    {
        showErrorInfo(E_ACTIVITY_NOT_EXIST, name_activity);
        return "";
    }
    hla=activity->getHLA();

    /*Valido?*/
    if(days<0)      return "Valor de dias no valido";

    /*No se puede adelantar mas alla de el inicio del proyecto*/
    if(days>activity->getTMin())  return "No se puede adelantar la actividad mas alla del instante de inicio del proyecto";

    /*Es factible?*/
    if(days>hla)
    {
        /*Que predecesoras condicionan su inicio temprano*/
        for(int i=0; i< activity->getList_Activities_pred()->size();i++)
        {
            act_pred=activity->getList_Activities_pred()->at(i)->activity;
            if(act_pred->getTMin()+act_pred->getTNormal()==activity->getTMin())
            {
                if(pred_problem.size()>0)  pred_problem.append(" -");
                pred_problem.append(act_pred->getName());
            }
        }
        return "Adelanto no factible, no se cumpliran las dependencias con las predecesoras "+pred_problem;
    }

    /*Como se ve afectada la limitacion?*/

    /*Para los nuevos instantes que se realizaria la actividad*/
    for(int d=activity->getTMin()-days; d<=activity->getTMin()-days+activity->getTNormal();d++)
    {
        /*Comprobar todos los recursos*/
        for(int r=0; activity->getResources().size();r++)
        {
            rec=activity->getResources().at(r)->resource_asig;
            n_resources=0;

            /*Selecciona las actividades que se estan realizando ese dia, sin incluir la actividad a analizar*/
            for(int i=0; i<activities.size();i++)
            {
                act=activities.at(i);
                if(act->getTMin()<=d&&act->getTMin()+act->getTNormal()-1>=d && act->getName()!=activity->getName())
                {
                    rec_asig=act->getResource( rec->getName());
                    if(rec_asig!=NULL) n_resources+=rec_asig->units_asig;
                }
            }

            /*El consumo de recursos mas el consumo que haria esa actividad si se realizase en ese instante, supera el limite de recursos?*/
            if(n_resources+activity->getResource(rec->getName())->units_asig>rec->getUnitsMax())
            {
                /*Guardamos todos los análisis para cada recurso*/
                if(!exceded_limitation) inform.append("La actividad se podria retrasar sin repercutir en la duración del proyecto, pero:\n");
                inform.append("      Excederia el maximo de recurso "+rec->getName()+" en "+num_to_str(n_resources+
activity->getResource(rec->getName())->units_asig-rec->getUnitsMax())+" unidades el instante "+
num_to_str(d)+"\n");
                exceded_limitation=true;
            }
        }
    }

    if(!exceded_limitation)      return "Adelanto factible, se cumpliran las dependencias del proyecto y se seguiria cumpliendo la limitacion de recursos";
    else                          return inform;
}

```

PARÁMETROS DE ENTRADA

- Actividad a analizar
- Días en que se adelantará su comienzo

PARÁMETROS DE SALIDA

- String con el informe de su análisis

PASOS

- 1) Comprobar si es un día valido o coherente.
- 2) Si el número de días que se quiere adelanta es mayor al número de días que se podría adelantar la actividad sin modificar la duración del proyecto, entonces s no sería un cambio factible, y estudiaríamos qué actividades predecesoras limitarían este adelantamiento.
- 3) Si se puede retrasar sin afectar la duración del proyecto, estudiamos como afectaría este cambio a los recursos. De forma que se obtiene cual sería la asignación de recursos si moviésemos la actividad los días pertinentes.
- 4) Estudiamos para cada día que duraría la actividad desde su nuevo tiempo de comienzo, como afecta a los recursos. Si supera alguna limitación de algún recurso, informamos de ello junto con la cantidad que se excedería y guardaríamos este resultado.
- 5) Si no se ha excedido ninguna limitación, informaríamos de que se cumpliría con las limitaciones, sino informamos de cada recurso lo que se excede.

4.5.2 Probabilidad

El proyecto realiza análisis de probabilidades, que pueden trabajar de 3 formas:

- Obtenemos cual será la probabilidad de terminar el proyecto antes de una duración dada y con la una variación recibida. Primero calculamos la media, que es la duración del camino crítico, de forma que la duración máxima que queremos se resta por la media y se divide por la raíz de la varianza, y por último aplicarle una función para convertirla a una normal 0 1. Cabe decir que si el parámetro que la pasamos a GetCumulativeDensityNormal es negativo, el lo transforma para poder trabajar con el.

```
float Project::getProbabDurationLessThan(int x, double var)
{
    double media=end->getTMax();
    double var_max=0;
    double var_temp=0;

    /*Valido?*/
    if(x<0) return 0;
    if(var<0) return 0;

    /*Probabilidad de que el proyecto termine antes de x, segun la varianza*/
    return GetCumulativeDensityNormal( x,media,sqrt(var))*100;
}
```

- Obtenemos la probabilidad de que el proyecto tarde en finalizarse entre 2 duraciones de tiempo, con una varianza del proyecto dada. Primero calculamos la media que es la duración del camino crítico. Al ser un intervalo se tiene que restar la probabilidad de un límite por la del otro.

```
float Project::getProbabDurationBetween(int x,int y, double var)
{
    double media=end->getTMax();

    /*Valido?*/
    if(x<0 || y>0) return 0;
    if(var<0) return 0;

    /*Probabilidad de que el proyecto termine entre x e y segun la varianza*/
    float norm_x=GetCumulativeDensityNormal( x,media,sqrt(var));
    float norm_y=GetCumulativeDensityNormal( y,media,sqrt(var));
    return (norm_y-norm_x)*100;
}
```

- Obtenemos cual será la duración del proyecto con una probabilidad y varianza dadas. Primero calculamos la media que es la duración del camino crítico. De forma que a esa probabilidad normal le calculamos su inversa y multiplicándola por la raíz de la varianza más la media, tendríamos la duración.

```
int Project::getDurationByProbability(double x, double var)
{
    double media=end->getTMax();

    /*Valido?*/
    if(x<0 || x>1)
        return 0;
    if(var<0)
        return 0;

    /*Duración del proyecto con una probabilidad x y una varianza dada*/
    return GetCumulativeDensityNormalInv( x, media,sqrt(var));
}
```

4.5.3 LIBRERÍA DISTRIBUCION NORMAL

Para implementar el cálculo de probabilidades que siguen una normal, hemos hecho uso de librerías de libre distribución, disponibles en la siguiente referencia, ya que la implementación del algoritmo de cálculo de probabilidades según una normal (0,1) queda fuera del ámbito de este proyecto.

<http://www.richelbilderbeek.nl/CppGetCumulativeDensityNormal.htm>

La siguiente función se encarga de obtener la probabilidad acumulada para un valor x, siguiendo una distribución normal con media 0 y desviación 1.

```
double GetCumulativeDensityNormal(const double x)
{
    const double c0 = 0.2316419;
    const double c1 = 1.330274429;
    const double c2 = 1.821255978;
    const double c3 = 1.781477937;
    const double c4 = 0.356563782;
    const double c5 = 0.319381530;
    const double c6 = 0.398942280401;
    const double negative = (x < 0 ? 1.0 : 0.0);
    const double xPos = (x < 0.0 ? -x : x);
    const double k = 1.0 / (1.0 + (c0 * xPos));
    const double y1 = (((((c1*k-c2)*k)+c3)*k)-c4)*k+c5)*k;
    const double y2 = 1.0 - (c6*std::exp(-0.5*xPos*xPos)*y1);
    return ((1.0-negative)*y2) + (negative*(1.0-y2));
}
```

La siguiente función obtiene la probabilidad acumulada para cualquier valor x, con media mean y desviacion stddev. Para ello aplica de la fórmula $\frac{x-mean}{stddev}$, que transforma el valor x, con media mean y desviacion tipica stddev, a su equivalente con media 0 y desviacion 1. Luego obtiene la probabilidad acumulada para ese valor siguiendo una distribucion normal(0,1)

```
double GetCumulativeDensityNormal( const double x, const double mean,const double stddev)
{
    return GetCumulativeDensityNormal(
        (x - mean) / stddev);
}
```

La siguiente función realiza el proceso contrario a `GetCumulativeDensityNormal(const double x)`, calcula el valor que tendría una densidad de probabilidad acumulada x, siguiendo una normal (0,1). El código que calcula este valor es de libre distribución, disponible en:

http://code.google.com/p/py-fcm/source/browse/src/statistics/cdp_ext/ltnorm.cpp?spec=svn4a5f99e72107f981b52215a0b10a1a548854e6bd&r=4a5f99e72107f981b52215a0b10a1a548854e6bd

```

double ltqnorm(double p)
{
    double q, r;
    errno = 0;

    if (p < 0 || p > 1)
    {
        errno = EDOM;
        return 0.0;
    }
    else if (p == 0)
    {
        errno = ERANGE;
        return -HUGE_VAL /* minus "infinity" */;
    }
    else if (p == 1)
    {
        errno = ERANGE;
        return HUGE_VAL /* "infinity" */;
    }
    else if (p < LOW)
    {
        /* Rational approximation for lower region */
        q = sqrt(-2*log(p));
        return (((((c[0]*q+c[1])*q+c[2])*q+c[3])*q+c[4])*q+c[5]) /
               (((d[0]*q+d[1])*q+d[2])*q+d[3])*q+1);
    }
    else if (p > HIGH)
    {
        /* Rational approximation for upper region */
        q = sqrt(-2*log(1-p));
        return -((((c[0]*q+c[1])*q+c[2])*q+c[3])*q+c[4])*q+c[5]) /
               (((d[0]*q+d[1])*q+d[2])*q+d[3])*q+1;
    }
    else
    {
        /* Rational approximation for central region */
        q = p - 0.5;
        r = q*q;
        return (((((a[0]*r+a[1])*r+a[2])*r+a[3])*r+a[4])*r+a[5])*q /
               (((((b[0]*r+b[1])*r+b[2])*r+b[3])*r+b[4])*r+1);
    }
}

```

Funcion que obtiene el valor que corresponderia a la probabilidad x, con media mean y desviacion stdvar. Para ello obtenemos el valor para la probabilidad x que digue una dsitribución normal (0,1), el cual lo transformariamos en un valor con media mean y desviacion stddev, mediante la siguiente fórmula $x * stddev + mean$

```

double GetCumulativeDensityNormalInv( const double x, const double mean,const double stddev)
{
    double normal=ltqnorm(x);
    double result=normal*stddev+mean;
    return result;
}

```

4.5.4 Librería de reglas

Función que realiza un ordenamiento mediante un algoritmo de Divide y Vencerás; en donde nos aprovechamos de que la lista esta ordenada, la dividimos por la mitad, y solo inspeccionamos la mitad donde deberá insertarse, según el resultado que devuelva esa regla de prioridad. El ordenamiento se hace por comparación entre dos actividades, así cuando la lista ya solo contenga una actividad insertará la actividad dada , según la regla de prioridad, que le indicará que se inserte delante o detrás de la actividad de la lista.

```
deque<Activity*> binsearch(deque<Activity*> activities, Activity * act, int left, int right, int rule)
{
    /*Esta funcion hace un ordenamiento de actividades segun una regla de prioridad,
    siguiendo una estrategia de divide y venceras, con coste temporal de O(log n)*/

    if (left > right) return activities;

    /*Solo hay un elemento*/
    if(left==right)
    {
        /*Ordenar segun la regla de prioridad*/
        if(choiceRule(act,activities.at(left), rule)) activities.insert(activities.begin()+left, act);
        else activities.insert(activities.begin()+left+1, act);
        return activities;
    }
    int middle = (left+right)/2;

    /*Como la lista esta ordenada, si la partimos en 2, en que parte estara la posicion donde insertarla*/
    if (choiceRule(act,activities.at(middle), rule)) return binsearch(activities,act,left,middle,rule);
    else return binsearch(activities,act,middle+1,right,rule);
}
```

Función que elige la regla a aplicar según la opción que se le indique. Seleccionará la regla y después de aplicarla, devolverá su resultado, que será usado en el ordenamiento.

```
bool choiceRule(Activity * act1,Activity * act2, int rule)
{
    switch(rule){
        case MIN_LFT_FIFO: return MinLFT_FIFO(act1,act2);
        break;

        case MIN_EFT_FIFO: return MinEFT_FIFO(act1,act2);
        break;

        case MIN_LST_FIFO: return MinLST_FIFO(act1,act2);
        break;

        case MIN_EST_FIFO: return MinEST_FIFO(act1,act2);
        break;

        case MIN_HT_FIFO: return MinHT_FIFO(act1,act2);
        break;

        case MIN_HL_FIFO: return MinHL_FIFO(act1,act2);
        break;

        case MAX_DEM_FIFO: return MaxDEM_FIFO(act1,act2);
        break;

        case MIN_DUR_FIFO: return MinDUR_FIFO(act1,act2);
        break;

        case MAX_NSUC_FIFO: return MaxNSUC_FIFO(act1,act2);
        break;

        default: return false; break;
    }
}
```

A continuación tenemos implementadas una serie de funciones que realizan comparaciones entre actividades y determinaran como se ordenan.

Orden creciente, según el instante de fin tardío

```
bool MinLFT_FIFO(Activity * act1,Activity * act2)
{
    /*Minimo instante de finalizacion tardio*/
    return(act1->getTMax()+act1->getTNormal()<act2->getTMax()+act2->getTNormal());
}
```

Orden creciente, según el instante de fin temprano

```
bool MinEFT_FIFO(Activity * act1,Activity * act2)
{
    /*Minimo instante de finalizacion mas temprano*/
    return(act1->getTMin()+act1->getTNormal()<act2->getTMin()+act2->getTNormal());
}
```

Orden creciente, según la holgura total

```
bool MinHT_FIFO(Activity * act1,Activity * act2)
{
    /*Minima holgura total*/
    return(act1->getHTotal()<act2->getHTotal());
}
```

Orden creciente, según la holgura libre

```
bool MinHL_FIFO(Activity * act1,Activity * act2)
{
    /*Minima holgura libre*/
    return(act1->getHFree()<act2->getHFree());
}
```

Orden creciente, según el instante de inicio temprano

```
bool MinEST_FIFO(Activity * act1,Activity * act2)
{
    /*Minimo instante de comienzo mas temprano*/
    return(act1->getTMin()<act2->getTMin());
}
```

Orden creciente, según el instante de inicio tardío

```
bool MinLST_FIFO(Activity * act1,Activity * act2)
{
    /*Minimo instante de comienzo mas tardio*/
    return(act1->getTMax()<act2->getTMax());
}
```

Orden creciente, según la duración de las actividades

```
bool MinDUR_FIFO(Activity * act1,Activity * act2)
{
    /*Minima duracion de la actividad*/
    return(act1->getTNormal()<act2->getTNormal());
}
```

Orden decreciente, según la cantidad de actividades sucesoras

```
bool MaxNSUC_FIFO(Activity * act1,Activity * act2)
{
    /*Maximo numero de sucesoras*/
    return(act1->getList_Activities_suc()->size()>act2->getList_Activities_suc()->size());
}
```

Orden decreciente, según la demanda total de todos los recursos que tiene una actividad

```
bool MaxDEM_FIFO(Activity * act1,Activity * act2)
{
    int prior1=0, prior2=0;

    /*Maxima demanda de recursos*/
    for(int i=0; i<act1->getResources().size();i++)
        prior1+=act1->getResources().at(i)->units_asig;
    for(int i=0; i<act2->getResources().size();i++)
        prior2+=act2->getResources().at(i)->units_asig;

    prior1*=act1->getTNormal();
    prior2*=act2->getTNormal();
    return(prior1>prior2);
}
```

5 Ejemplos

5.1 Ejemplo 1

Max R1=5

Max R2=6

NOMBRE	DURACION	SUCESORAS	RECURSOS (R1,R2)	COSTE NORMAL	COSTE OPORTUNIDAD	TIEMPO TOPE	COSTE OPORTUNIDAD
A	5		(3,2)	0	0	5	0
B	5		(2,4)	0	0	5	0
C	4	A	(3,1)	0	0	4	0
D	2	A	(4,3)	0	0	2	0
E	3	D	(2,0)	0	0	3	0
F	3	D	(1,1)	0	0	3	0
G	4	B	(3,1)	0	0	4	0
H	3	F-G	(2,2)	0	0	3	0
I	3	C	(3,2)	0	0	3	0
J	2	E-I	(4,1)	0	0	2	0
K	3	J	(5,4)	0	0	3	0

5.1.1 Camino Crítico

5.1.1.1 Informe

Informe Actividades

Camino Crítico: A-C-I-J-K
Fecha Inicio: 17/12/2012 Fecha fin: 10/01/2013
Duración: 17 días laborables

	Nombre	F. Inicio Temprano	F. Inicio Tardío	F. Fin Temprano	F. Fin Tardío	HT	HL	Sucesoras
1	A	17/12/2012	17/12/2012	21/12/2012	21/12/2012	0	0	C-D
2	B	17/12/2012	24/12/2012	21/12/2012	31/12/2012	5	0	G
3	C	24/12/2012	24/12/2012	28/12/2012	28/12/2012	0	0	I
4	D	24/12/2012	27/12/2012	26/12/2012	28/12/2012	2	0	E-F
5	E	27/12/2012	31/12/2012	31/12/2012	03/01/2013	2	2	J
6	F	27/12/2012	03/01/2013	31/12/2012	07/01/2013	4	0	H
7	G	24/12/2012	02/01/2013	28/12/2012	07/01/2013	5	1	H
8	H	02/01/2013	08/01/2013	04/01/2013	10/01/2013	4	4	

Calendario Proyecto

	diciembre 2012						
	lun	mar	mié	jue	vie	sáb	dom
48	26	27	28	29	30	1	2
49	3	4	5	6	7	8	9
50	10	11	12	13	14	15	16
51	17	18	19	20	21	22	23
52	24	25	26	27	28	29	30
1	31	1	2	3	4	5	6

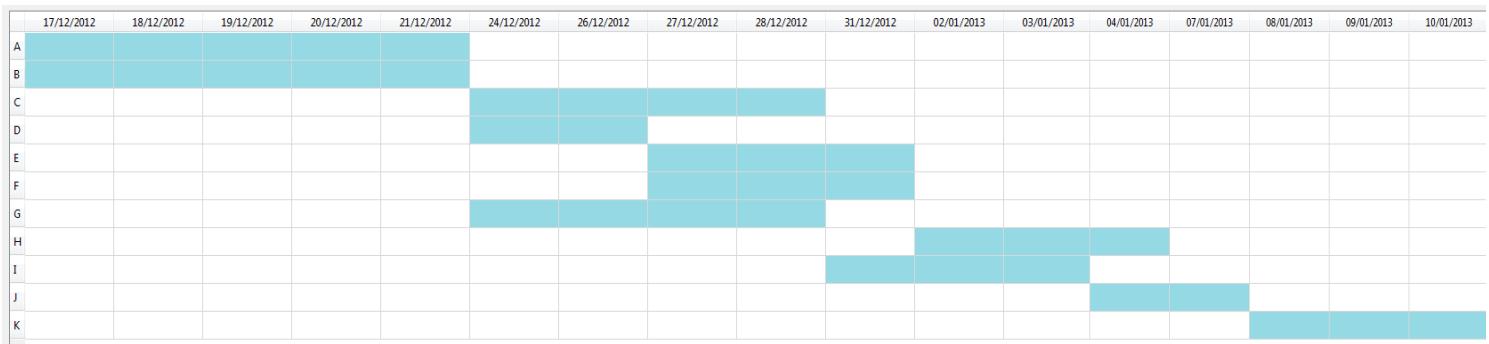
Duración: 17 días laborables

	Nombre	F. Inicio Temprano	F. Inicio Tardío	F. Fin Temprano	F. Fin Tardío	HT	HL	Sucesoras
6	F	27/12/2012	03/01/2013	31/12/2012	07/01/2013	4	0	H
7	G	24/12/2012	02/01/2013	28/12/2012	07/01/2013	5	1	H
8	H	02/01/2013	08/01/2013	04/01/2013	10/01/2013	4	4	
9	I	31/12/2012	31/12/2012	03/01/2013	03/01/2013	0	0	J
10	J	04/01/2013	04/01/2013	07/01/2013	07/01/2013	0	0	K
11	K	08/01/2013	08/01/2013	10/01/2013	10/01/2013	0	0	
12								
13								

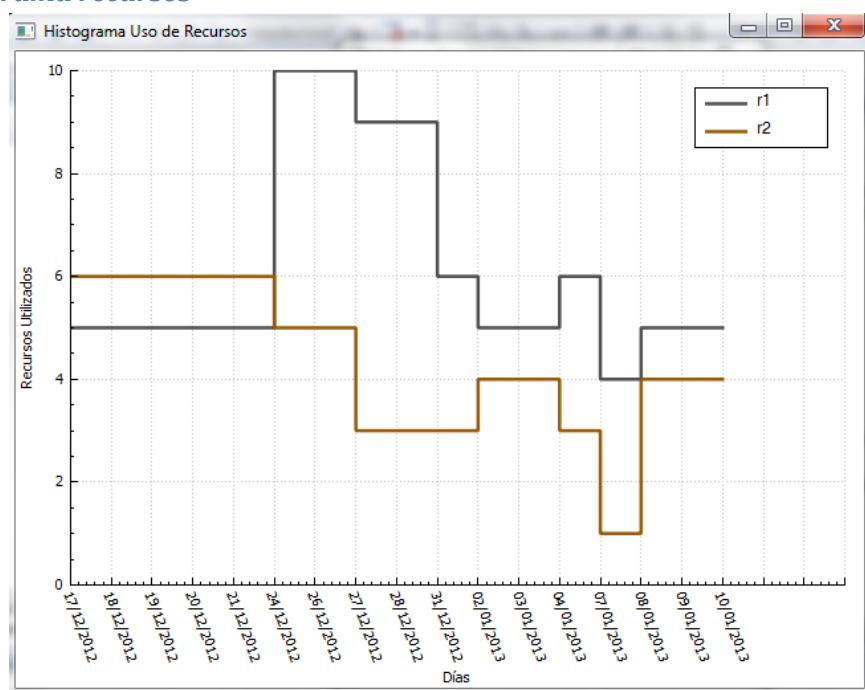
Calendario Proyecto

	enero 2013						
	lun	mar	mié	jue	vie	sáb	dom
1	31	1	2	3	4	5	6
2	7	8	9	10	11	12	13
3	14	15	16	17	18	19	20
4	21	22	23	24	25	26	27
5	28	29	30	31	1	2	3
6	4	5	6	7	8	9	10

5.1.1.2 Gantt



5.1.1.3 Histograma recursos



5.1.1.4 Informe debug:

----- CAMINO CRITICO -----		
Actividad:A	Tmin:0	Tmax:0
Actividad:B	Tmin:0	Tmax:5
Actividad:C	Tmin:5	Tmax:5
Actividad:D	Tmin:5	Tmax:7
Actividad:E	Tmin:7	Tmax:9
Actividad:F	Tmin:7	Tmax:11
Actividad:G	Tmin:5	Tmax:10
Actividad:H	Tmin:10	Tmax:14
Actividad:I	Tmin:9	Tmax:9
Actividad:J	Tmin:12	Tmax:12
Actividad:K	Tmin:14	Tmax:14

5.1.2 Limitación de recursos con regla LFT y esquema serie

5.1.2.1 Informe

Informe Actividades

Camino Crítico: A-C-I-J-K
Fecha Inicio: 17/12/2012 Fecha fin: 23/01/2013
Duración: 26 días laborables

Nombre	F. Inicio Temprano	F. Inicio Tardío	F. Fin Temprano	F. Fin Tardío	HT	HL	Sucesoras
1 A	17/12/2012	31/12/2012	21/12/2012	07/01/2013	9	0	C-D
2 B	03/01/2013	08/01/2013	09/01/2013	14/01/2013	3	0	G
3 C	24/12/2012	08/01/2013	28/12/2012	11/01/2013	9	7	I
4 D	31/12/2012	10/01/2013	02/01/2013	11/01/2013	7	0	E-F
5 E	03/01/2013	14/01/2013	07/01/2013	16/01/2013	7	6	J
6 F	03/01/2013	16/01/2013	07/01/2013	18/01/2013	9	6	H
7 G	10/01/2013	15/01/2013	15/01/2013	18/01/2013	3	0	H
8 H	16/01/2013	21/01/2013	18/01/2013	23/01/2013	3	3	

Calendario Proyecto

	lun	mar	mié	jue	vie	sáb	dom
48	26	27	28	29	30	1	2
49	3	4	5	6	7	8	9
50	10	11	12	13	14	15	16
51	17	18	19	20	21	22	23
52	24	25	26	27	28	29	30
1	31	1	2	3	4	5	6

Duración: 26 días laborables

Nombre	F. Inicio Temprano	F. Inicio Tardío	F. Fin Temprano	F. Fin Tardío	HT	HL	Sucesoras
6 F	03/01/2013	16/01/2013	07/01/2013	18/01/2013	9	6	H
7 G	10/01/2013	15/01/2013	15/01/2013	18/01/2013	3	0	H
8 H	16/01/2013	21/01/2013	18/01/2013	23/01/2013	3	3	
9 I	10/01/2013	14/01/2013	14/01/2013	16/01/2013	2	1	J
10 J	16/01/2013	17/01/2013	17/01/2013	18/01/2013	1	1	K
11 K	21/01/2013	21/01/2013	23/01/2013	23/01/2013	0	0	
12							
13							

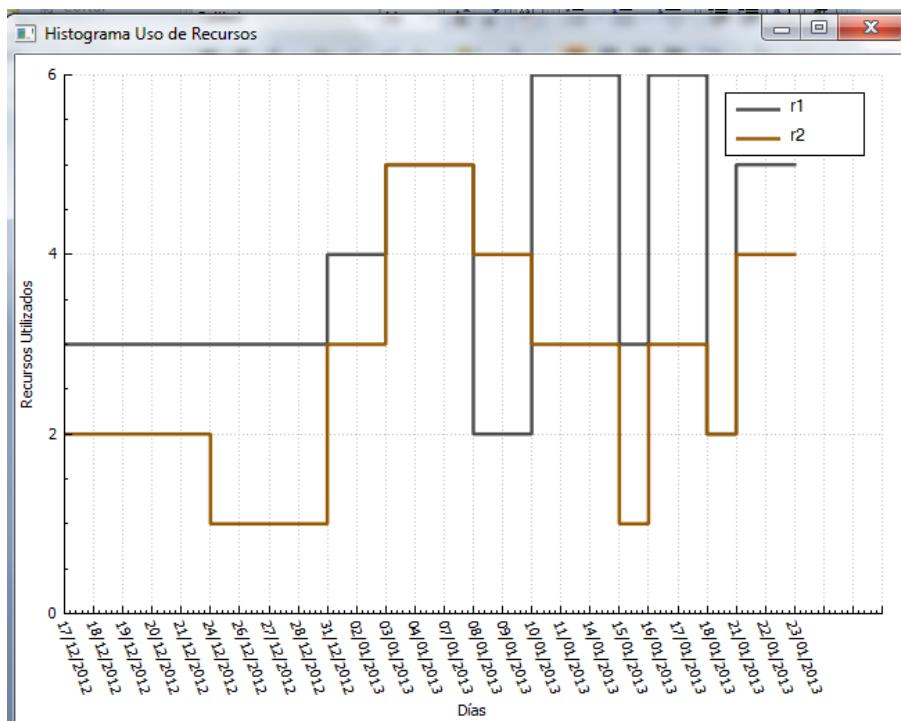
Calendario Proyecto

	lun	mar	mié	jue	vie	sáb	dom
1	31	1	2	3	4	5	6
2	7	8	9	10	11	12	13
3	14	15	16	17	18	19	20
4	21	22	23	24	25	26	27
5	28	29	30	31	1	2	3
6	4	5	6	7	8	9	10

5.1.2.2 Gantt



5.1.2.3 Histograma de recursos



5.1.2.4 Informe debug:

```
*****SERIE*****
disp0{ r1=6 r2=5}

ITERACION _____
elegir={A B }
La actividad A se secuencia en 0
disp0{ r1=3 r2=3}
disp5{ r1=6 r2=5}

ITERACION _____
elegir={C D B }
La actividad C se secuencia en 5
disp0{ r1=3 r2=3}
disp5{ r1=3 r2=4}
disp9{ r1=6 r2=5}

ITERACION _____
elegir={D B I }
La actividad D se secuencia en 9
disp0{ r1=3 r2=3}
disp5{ r1=3 r2=4}
disp9{ r1=2 r2=2}
disp11{ r1=6 r2=5}

ITERACION _____
elegir={B I E F }
La actividad B se secuencia en 11
disp0{ r1=3 r2=3}
disp5{ r1=3 r2=4}
disp9{ r1=2 r2=2}
disp11{ r1=4 r2=1}
disp16{ r1=6 r2=5}

ITERACION _____
elegir={I E F G }
La actividad I se secuencia en 16
disp0{ r1=3 r2=3}
disp5{ r1=3 r2=4}
disp9{ r1=2 r2=2}
disp11{ r1=4 r2=1}
disp16{ r1=3 r2=3}
disp19{ r1=6 r2=5}

ITERACION _____
elegir={E F G }
La actividad E se secuencia en 11
disp0{ r1=3 r2=3}
disp5{ r1=3 r2=4}
disp9{ r1=2 r2=2}
disp11{ r1=2 r2=1}
disp14{ r1=4 r2=1}
disp16{ r1=3 r2=3}
disp19{ r1=6 r2=5}

ITERACION _____
elegir={F G J }
La actividad F se secuencia en 11
disp0{ r1=3 r2=3}
disp5{ r1=3 r2=4}
disp9{ r1=2 r2=2}
disp11{ r1=1 r2=0}
disp14{ r1=4 r2=1}
disp16{ r1=3 r2=3}
disp19{ r1=6 r2=5}

ITERACION _____
elegir={G J }
La actividad G se secuencia en 16
disp0{ r1=3 r2=3}
disp5{ r1=3 r2=4}
disp9{ r1=2 r2=2}
disp11{ r1=1 r2=0}
disp14{ r1=4 r2=1}
disp16{ r1=0 r2=2}
disp19{ r1=3 r2=4}
disp20{ r1=6 r2=5}

ITERACION _____
elegir={J H }
La actividad J se secuencia en 20
disp0{ r1=3 r2=3}
disp5{ r1=3 r2=4}
```

```
disp9{ r1=2 r2=2}
disp11{ r1=1 r2=0}
disp14{ r1=4 r2=1}
disp16{ r1=0 r2=2}
disp19{ r1=3 r2=4}
disp20{ r1=2 r2=4}
disp22{ r1=6 r2=5}

ITERACION _____
elegir={H K }
La actividad H se secuencia en 20
disp0{ r1=3 r2=3}
disp5{ r1=3 r2=4}
disp9{ r1=2 r2=2}
disp11{ r1=1 r2=0}
disp14{ r1=4 r2=1}
disp16{ r1=0 r2=2}
disp19{ r1=3 r2=4}
disp20{ r1=0 r2=2}
disp22{ r1=4 r2=3}
disp23{ r1=6 r2=5}

ITERACION _____
elegir={K }
La actividad K se secuencia en 23
disp0{ r1=3 r2=3}
disp5{ r1=3 r2=4}
disp9{ r1=2 r2=2}
disp11{ r1=1 r2=0}
disp14{ r1=4 r2=1}
disp16{ r1=0 r2=2}
disp19{ r1=3 r2=4}
disp20{ r1=0 r2=2}
disp22{ r1=4 r2=3}
disp23{ r1=1 r2=1}
disp26{ r1=6 r2=5}
```

5.1.3 Retraso adelanto

5.1.3.1 Informe

Informe Actividades

Camino Crítico: A-C-I-J-K

Fecha Inicio: 17/12/2012 Fecha fin: 17/01/2013

Duración: 22 días laborables

	Nombre	F. Inicio Temprano	F. Inicio Tardío	F. Fin Temprano	F. Fin Tardío	HT	HL	Sucesoras	
1	A	17/12/2012	24/12/2012	21/12/2012	31/12/2012	5	0	C-D	
2	B	27/12/2012	02/01/2013	03/01/2013	08/01/2013	3	0	G	
3	C	27/12/2012	02/01/2013	02/01/2013	07/01/2013	3	3	I	
4	D	24/12/2012	04/01/2013	26/12/2012	07/01/2013	7	4	E-F	
5	E	03/01/2013	08/01/2013	07/01/2013	10/01/2013	3	3	J	
6	F	03/01/2013	10/01/2013	07/01/2013	14/01/2013	5	2	H	
7	G	04/01/2013	09/01/2013	09/01/2013	14/01/2013	3	0	H	
8	H	10/01/2013	15/01/2013	14/01/2013	17/01/2013	3	3		

Calendario Proyecto

	diciembre, 2012						
	lun	mar	mié	jue	vie	sáb	dom
48	26	27	28	29	30	1	2
49	3	4	5	6	7	8	9
50	10	11	12	13	14	15	16
51	17	18	19	20	21	22	23
52	24	25	26	27	28	29	30
1	31	1	2	3	4	5	6

Calendario Proyecto

	enero, 2013							
	lun	mar	mié	jue	vie	sáb	dom	
6	F	03/01/2013	10/01/2013	07/01/2013	14/01/2013	5	2	H
7	G	04/01/2013	09/01/2013	09/01/2013	14/01/2013	3	0	H
8	H	10/01/2013	15/01/2013	14/01/2013	17/01/2013	3	3	
9	I	08/01/2013	08/01/2013	10/01/2013	10/01/2013	0	0	J
10	J	11/01/2013	11/01/2013	14/01/2013	14/01/2013	0	0	K
11	K	15/01/2013	15/01/2013	17/01/2013	17/01/2013	0	0	
12								
13								

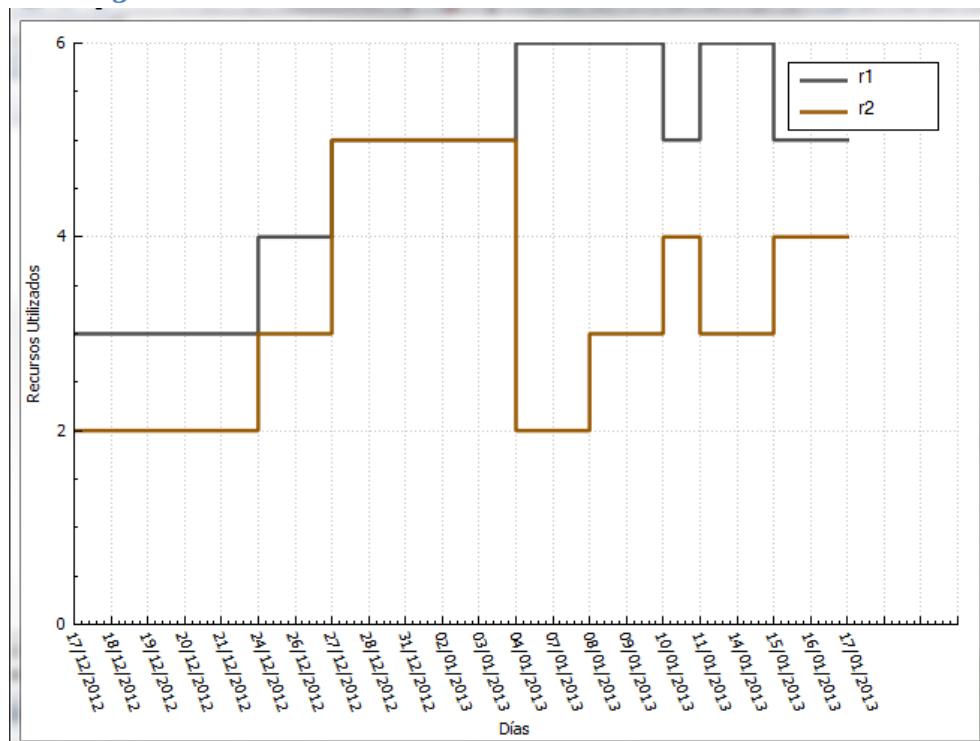
Calendario Proyecto

	enero, 2013						
	lun	mar	mié	jue	vie	sáb	dom
1	31	1	2	3	4	5	6
2	7	8	9	10	11	12	13
3	14	15	16	17	18	19	20
4	21	22	23	24	25	26	27
5	28	29	30	31	1	2	3
6	4	5	6	7	8	9	10

5.1.3.2 Gantt

	17/12/2012	3/12/2012	9/12/2012	15/12/2012	1/12/2012	5/12/2012	7/12/2012	3/12/2012	11/12/2012	2/1/2013	3/1/2013	4/1/2013	7/1/2013	3/1/2013	10/1/2013	13/1/2013	1/1/2013	1/1/2013	4/1/2013	1/1/2013	5/1/2013	17/1/2013
A																						
B																						
C																						
D																						
E																						
F																						
G																						
H																						
I																						
J																						
K																						

5.1.3.3 Histograma de recursos



5.1.3.4 Informe debug

```
#####
# ADELANTO/ATRASO#####
||||||||||||||||| RETRASO|||||||||||||||||
elegir={K H J G I B F E D C A }

*****SERIE*****
ITERACION
elegir={K H J G I B F E D C A }
La actividad K se secuencia en 0
disp0{ r1=1 r2=1}
disp3{ r1=6 r2=5}

ITERACION
elegir={H J G I B F E D C A }
La actividad H se secuencia en 3
disp0{ r1=1 r2=1}
disp3{ r1=4 r2=3}
disp6{ r1=6 r2=5}

ITERACION
elegir={J G I B F E D C A }
La actividad J se secuencia en 3
disp0{ r1=1 r2=1}
disp3{ r1=0 r2=2}
disp5{ r1=4 r2=3}
disp6{ r1=6 r2=5}

ITERACION
elegir={G I B F E D C A }
La actividad G se secuencia en 6
disp0{ r1=1 r2=1}
disp3{ r1=0 r2=2}
disp5{ r1=4 r2=3}
disp6{ r1=3 r2=4}
disp10{ r1=6 r2=5}

ITERACION
elegir={I B F E D C A }
La actividad I se secuencia en 5
disp0{ r1=1 r2=1}
disp3{ r1=0 r2=2}
disp5{ r1=1 r2=1}
disp6{ r1=0 r2=2}
disp8{ r1=3 r2=4}
disp10{ r1=6 r2=5}

ITERACION
elegir={B F E D C A }
La actividad B se secuencia en 10
disp0{ r1=1 r2=1}
disp3{ r1=0 r2=2}
disp5{ r1=1 r2=1}
disp6{ r1=0 r2=2}
disp8{ r1=3 r2=4}
disp10{ r1=4 r2=1}
disp15{ r1=6 r2=5}

ITERACION
elegir={F E D C A }
La actividad F se secuencia en 8
disp0{ r1=1 r2=1}
disp3{ r1=0 r2=2}
disp5{ r1=1 r2=1}
disp6{ r1=0 r2=2}
disp8{ r1=2 r2=3}
disp10{ r1=3 r2=0}
disp11{ r1=4 r2=1}
disp15{ r1=6 r2=5}

ITERACION
elegir={E D C A }
La actividad E se secuencia en 8
disp0{ r1=1 r2=1}
disp3{ r1=0 r2=2}
disp5{ r1=1 r2=1}
disp6{ r1=0 r2=2}
disp8{ r1=0 r2=3}
disp10{ r1=1 r2=0}
disp11{ r1=4 r2=1}
disp15{ r1=6 r2=5}

ITERACION
```

```

elegir={D C A }
La actividad D se secuencia en 15
disp0{ r1=1 r2=1}
disp3{ r1=0 r2=2}
disp5{ r1=1 r2=1}
disp6{ r1=0 r2=2}
disp8{ r1=0 r2=3}
disp10{ r1=1 r2=0}
disp11{ r1=4 r2=1}
disp15{ r1=2 r2=2}
disp17{ r1=6 r2=5}

ITERACION
elegir={C A }
La actividad C se secuencia en 11
disp0{ r1=1 r2=1}
disp3{ r1=0 r2=2}
disp5{ r1=1 r2=1}
disp6{ r1=0 r2=2}
disp8{ r1=0 r2=3}
disp10{ r1=1 r2=0}
disp11{ r1=1 r2=0}
disp15{ r1=2 r2=2}
disp17{ r1=6 r2=5}

ITERACION
elegir={A }
La actividad A se secuencia en 17
disp0{ r1=1 r2=1}
disp3{ r1=0 r2=2}
disp5{ r1=1 r2=1}
disp6{ r1=0 r2=2}
disp8{ r1=0 r2=3}
disp10{ r1=1 r2=0}
disp11{ r1=1 r2=0}
disp15{ r1=2 r2=2}
disp17{ r1=3 r2=3}
disp22{ r1=6 r2=5}

||||||||||||||||| ADELANTO|||||||||||||||
elegir={A D B C E F G I H J K }

*****SERIE*****
ITERACION
elegir={A D B C E F G I H J K }
La actividad A se secuencia en 0
disp0{ r1=3 r2=3}
disp5{ r1=6 r2=5}

ITERACION
elegir={D B C E F G I H J K }
La actividad D se secuencia en 5
disp0{ r1=3 r2=3}
disp5{ r1=2 r2=2}
disp7{ r1=6 r2=5}

ITERACION
elegir={B C E F G I H J K }
La actividad B se secuencia en 7
disp0{ r1=3 r2=3}
disp5{ r1=2 r2=2}
disp7{ r1=4 r2=1}
disp12{ r1=6 r2=5}

ITERACION
elegir={C E F G I H J K }
La actividad C se secuencia en 7
disp0{ r1=3 r2=3}
disp5{ r1=2 r2=2}
disp7{ r1=1 r2=0}
disp11{ r1=4 r2=1}
disp12{ r1=6 r2=5}

ITERACION
elegir={E F G I H J K }
La actividad E se secuencia en 11
disp0{ r1=3 r2=3}
disp5{ r1=2 r2=2}
disp7{ r1=1 r2=0}
disp11{ r1=2 r2=1}
disp12{ r1=4 r2=5}
disp14{ r1=6 r2=5}

ITERACION
elegir={F G I H J K }
La actividad F se secuencia en 11

```

```

disp0{ r1=3 r2=3}
disp5{ r1=2 r2=2}
disp7{ r1=1 r2=0}
disp11{ r1=1 r2=0}
disp12{ r1=3 r2=4}
disp14{ r1=6 r2=5}

ITERACION _____
elegir={G I H J K }
La actividad G se secuencia en 12
disp0{ r1=3 r2=3}
disp5{ r1=2 r2=2}
disp7{ r1=1 r2=0}
disp11{ r1=1 r2=0}
disp12{ r1=0 r2=3}
disp14{ r1=3 r2=4}
disp16{ r1=6 r2=5}

ITERACION _____
elegir={I H J K }
La actividad I se secuencia en 14
disp0{ r1=3 r2=3}
disp5{ r1=2 r2=2}
disp7{ r1=1 r2=0}
disp11{ r1=1 r2=0}
disp12{ r1=0 r2=3}
disp14{ r1=0 r2=2}
disp16{ r1=3 r2=3}
disp17{ r1=6 r2=5}

ITERACION _____
elegir={H J K }
La actividad H se secuencia en 16
disp0{ r1=3 r2=3}
disp5{ r1=2 r2=2}
disp7{ r1=1 r2=0}
disp11{ r1=1 r2=0}
disp12{ r1=0 r2=3}
disp14{ r1=0 r2=2}
disp16{ r1=1 r2=1}
disp17{ r1=4 r2=3}
disp19{ r1=6 r2=5}

ITERACION _____
elegir={J K }
La actividad J se secuencia en 17
disp0{ r1=3 r2=3}
disp5{ r1=2 r2=2}
disp7{ r1=1 r2=0}
disp11{ r1=1 r2=0}
disp12{ r1=0 r2=3}
disp14{ r1=0 r2=2}
disp16{ r1=1 r2=1}
disp17{ r1=0 r2=2}
disp19{ r1=6 r2=5}

ITERACION _____
elegir={K }
La actividad K se secuencia en 19
disp0{ r1=3 r2=3}
disp5{ r1=2 r2=2}
disp7{ r1=1 r2=0}
disp11{ r1=1 r2=0}
disp12{ r1=0 r2=3}
disp14{ r1=0 r2=2}
disp16{ r1=1 r2=1}
disp17{ r1=0 r2=2}
disp19{ r1=1 r2=1}
disp22{ r1=6 r2=5}

```

5.1.4 Nivelación recursos

5.1.4.1 Informe

Informe Actividades

Camino Crítico: A-C-I-J-K
Fecha Inicio: 17/12/2012 Fecha fin: 17/01/2013
Duración: 22 días laborables

Nombre	F.Inicio Temprano	F.Inicio Tardío	F.Fin Temprano	F.Fin Tardío	HT	HL	Sucesoras
1 A	17/12/2012	24/12/2012	21/12/2012	31/12/2012	5	0	C-D
2 B	27/12/2012	02/01/2013	03/01/2013	08/01/2013	3	0	G
3 C	27/12/2012	02/01/2013	02/01/2013	07/01/2013	3	3	I
4 D	24/12/2012	04/01/2013	26/12/2012	07/01/2013	7	4	E-F
5 E	03/01/2013	08/01/2013	07/01/2013	10/01/2013	3	3	J
6 F	03/01/2013	10/01/2013	07/01/2013	14/01/2013	5	2	H
7 G	04/01/2013	09/01/2013	09/01/2013	14/01/2013	3	0	H
8 H	10/01/2013	15/01/2013	14/01/2013	17/01/2013	3	3	

Calendario Proyecto

diciembre, 2012

	lun	mar	mié	jue	vié	sáb	dom
48	26	27	28	29	30	1	2
49	3	4	5	6	7	8	9
50	10	11	12	13	14	15	16
51	17	18	19	20	21	22	23
52	24	25	26	27	28	29	30
1	31	1	2	3	4	5	6

Duración: 22 días laborables

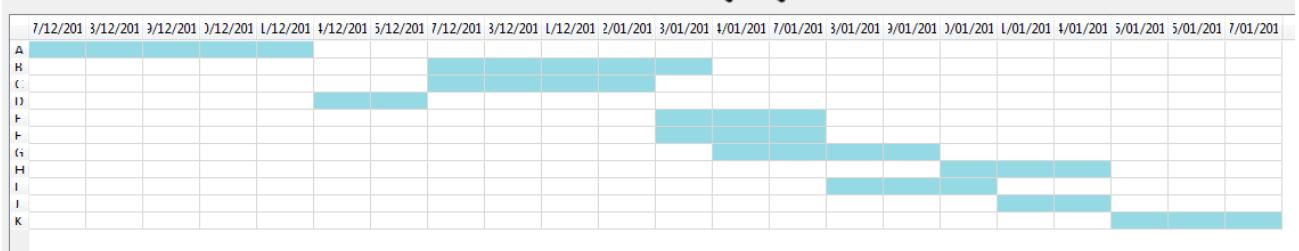
Nombre	F.Inicio Temprano	F.Inicio Tardío	F.Fin Temprano	F.Fin Tardío	HT	HL	Sucesoras
6 F	03/01/2013	10/01/2013	07/01/2013	14/01/2013	5	2	H
7 G	04/01/2013	09/01/2013	09/01/2013	14/01/2013	3	0	H
8 H	10/01/2013	15/01/2013	14/01/2013	17/01/2013	3	3	
9 I	08/01/2013	08/01/2013	10/01/2013	10/01/2013	0	0	J
10 J	11/01/2013	11/01/2013	14/01/2013	14/01/2013	0	0	K
11 K	15/01/2013	15/01/2013	17/01/2013	17/01/2013	0	0	
12							
13							

Calendario Proyecto

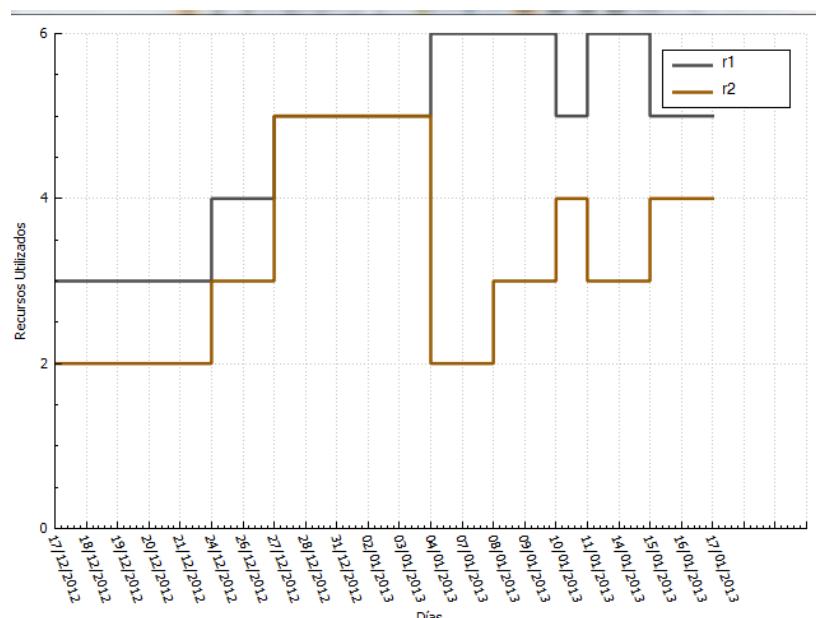
enero, 2013

	lun	mar	mié	jue	vié	sáb	dom
1	31	1	2	3	4	5	6
2	7	8	9	10	11	12	13
3	14	15	16	17	18	19	20
4	21	22	23	24	25	26	27
5	28	29	30	31	1	2	3
6	4	5	6	7	8	9	10

5.1.4.2 Gantt



5.1.4.3 Histograma de recursos



5.1.4.4 Informe debug

```
***** NIVELACION RECURSOS *****

Orden de las actividades a mover: K J H I G F E B C D A
CVoriginal = 0.57622

K   TMin Original=20  H.Libre=0:
    K se realiza en 19
-----
J   TMin Original=18  H.Libre=0:
    J se realiza en 17
-----
H   TMin Original=17  H.Libre=3:
        Se supera la limitacion de recursos
        Se supera la limitacion de recursos
        Movemos la actividad H al instante 17 con un CV=2e+012
        Se supera la limitacion de recursos
        Se supera la limitacion de recursos
        Movemos la actividad H al instante 18 con un CV=2e+012
        Se supera la limitacion de recursos
        Se supera la limitacion de recursos
        Movemos la actividad H al instante 19 con un CV=2e+012
        H se realiza en 16
-----
I   TMin Original=15  H.Libre=0:
    I se realiza en 14
-----
G   TMin Original=13  H.Libre=0:
    G se realiza en 12
```

```

F   TMin Original=12  H.Libre=2:
      Se supera la limitacion de recursos
      Movemos la actividad F al instante 12 con un CV=1e+012
      Se supera la limitacion de recursos
      Movemos la actividad F al instante 13 con un CV=1e+012
      F se realiza en 11
-----
E   TMin Original=12  H.Libre=3:
      Se supera la limitacion de recursos
      Movemos la actividad E al instante 12 con un CV=1e+012
      Se supera la limitacion de recursos
      Movemos la actividad E al instante 13 con un CV=1e+012
      Se supera la limitacion de recursos
      Movemos la actividad E al instante 14 con un CV=1e+012
      E se realiza en 11
-----
B   TMin Original=8   H.Libre=0:
      B se realiza en 7
-----
C   TMin Original=8   H.Libre=3:
      Se supera la limitacion de recursos
      Se supera la limitacion de recursos
      Movemos la actividad C al instante 8 con un CV=2e+012
      Se supera la limitacion de recursos
      Se supera la limitacion de recursos
      Movemos la actividad C al instante 9 con un CV=2e+012
      Se supera la limitacion de recursos
      Se supera la limitacion de recursos
      Movemos la actividad C al instante 10 con un CV=2e+012
      C se realiza en 7
-----
D   TMin Original=6   H.Libre=4:
      Se supera la limitacion de recursos
      Se supera la limitacion de recursos
      Movemos la actividad D al instante 6 con un CV=2e+012
      Se supera la limitacion de recursos
      Se supera la limitacion de recursos
      Movemos la actividad D al instante 7 con un CV=2e+012
      Se supera la limitacion de recursos
      Se supera la limitacion de recursos
      Movemos la actividad D al instante 8 con un CV=2e+012
      Se supera la limitacion de recursos
      Se supera la limitacion de recursos
      Movemos la actividad D al instante 9 con un CV=2e+012
      D se realiza en 5
-----
A   TMin Original=1   H.Libre=0:
      A se realiza en 0
-----
```

5.1.5 Análisis

The screenshots illustrate the analysis of activity E in a project. The first two windows show the results of advancing activity E by 4 units, while the third shows the results of advancing it by 5 units.

Top Left Window (Activity E Advanced by 4 units):

- Flexibilidad Actividad:** Checked.
- Actividad:** E
- HLR:** 3
- HLA:** 4
- Adelantar:** Selected (radio button)
- Atrasar:** Not selected (radio button)
- Value:** 4

Top Right Window (Information Message):

Flexibilidad Actividad E: La actividad se podría retrasar sin repercutir en la duración del proyecto, pero:
 Excedería el maximo de recurso r1 en 1 unidades el instante 7
 Excedería el maximo de recurso r1 en 1 unidades el instante 8
 Excedería el maximo de recurso r1 en 1 unidades el instante 9
 Excedería el maximo de recurso r1 en 1 unidades el instante 10

Bottom Left Window (Activity E Advanced by 5 units):

- Flexibilidad Actividad:** Checked.
- Actividad:** E
- HLR:** 3
- HLA:** 4
- Adelantar:** Selected (radio button)
- Atrasar:** Not selected (radio button)
- Value:** 5

Bottom Right Window (Information Message):

Flexibilidad Actividad E: Adelanto no factible, no se cumplirían las dependencias con las predecesoras D

5.2 Ejemplo 2

Nombre	Duración	Predecesoras	T tope	Coste oportunidad	Recursos (analista, diseñador, programador, tester , operario, auxiliar)
11	1		1	2	(3,0,0,0,0,0)
12	1	11	1	0.9	(5,6,0,0,0,0)
131	6	12	6	0.5	(3,0,0,0,0,0)
132	6	131	4	2.5	(2,0,0,1,0,3)
133	6	131	3	1.5	(1,0,0,0,0,0)
134	6	131	4	3.5	(3,3,0,0,0,0)
135	1	11-131	1	0	(3,0,0,0,0,2)
136	8	132-133-134-135	8	0	(5,0,0,0,0,0)
21	7	136	3	4	(5,8,2,0,0,0)
22	5	21	5	0	(0,4,1,0,0,0)
23	6	22	6	0	(0,3,2,0,0,0)
24	7	136-21	5	3	(0,2,7,0,1,2)
25	5	21	5	0	(0,2,0,0,0,0)
26	4	23-24-25	4	0.5	(0,0,0,0,0,4)
31	6	131-26	4	1.5	(0,2,4,0,0,3)
321	6	25-26	6	2.5	(2,2,1,3,0,1)
322	3	321	3	0	(0,3,2,2,0,0)
34	6	322-352	6	1.5	(2,1,8,2,0,2)
351	3	26	3	0	(2,0,2,0,0,0)
352	3	351	3	0	(1,0,2,0,0,0)
361	6	24-25-26	4	1.5	(2,0,3,0,0,0)
362	2	361	2	0	(1,0,4,0,0,0)
37	3	34	3	1	(2,0,5,0,6,0)
41	3	37-42-43-44-45	2	7.5	(2,2,2,2,2,0)

					Gestor de proyectos
42	40	37	40	0	(0,0,7,1,0,0)
43	10	37	10	0	(0,3,2,0,0,0)
44	7	37	7	0	(0,0,2,0,0,0)
45	5	25	3	0.5	(0,0,1,1,0,1)
46	10	41	10	1.5	(0,0,7,2,0,0)
47	3	46	3	0	(0,2,2,0,3,0)
51	3	47	3	0	(0,0,4,6,2,2)
521	2	51	2	0.5	(0,0,3,6,0,0)
522	1	521	1	0	(0,0,2,6,0,0)

5.2.1 Camino critico

5.2.1.1 Informe

Informe Actividades								
Camino Crítico: 11-12-131-132-136-21-22-23-26-321-322-34-37-41-46-47-51-521-522 11-12-131-132-136-21-22-23-26-321-322-34-37-42-41-46-47-51-521-522 11-12-131-133-136-21-22-23-26-321-322-34-37-41-46-47-51-521-522 11-12-131-133-136-21-22-23-26-321-322-34-37-42-41-46-47-51-521-522 11-12-131-134-136-21-22-23-26-321-322-34-37-41-46-47-51-521-522 11-12-131-134-136-21-22-23-26-321-322-34-37-42-41-46-47-51-521-522								
Fecha Inicio: 17/12/2012 Fecha fin: 12/06/2013 Duración: 124 días laborables								
Nombre	F.Inicio Temprano	F.Inicio Tardío	F.Fin Temprano	F.Fin Tardío	HT	HL	Sucesoras	
1 11	17/12/2012	17/12/2012	17/12/2012	17/12/2012	0	0	12-135	
2 12	18/12/2012	18/12/2012	18/12/2012	18/12/2012	0	0	131	
3 131	19/12/2012	19/12/2012	27/12/2012	27/12/2012	0	0	132-133-134-13...	
4 132	28/12/2012	28/12/2012	07/01/2013	07/01/2013	0	0	136	
5 133	28/12/2012	28/12/2012	07/01/2013	07/01/2013	0	0	136	
6 134	28/12/2012	28/12/2012	07/01/2013	07/01/2013	0	0	136	

Calendario Proyecto								
diciembre, 2012								
	lun	mar	mié	jue	vie	sáb	dom	
48	26	27	28	29	30	1	2	
49	3	4	5	6	7	8	9	
50	10	11	12	13	14	15	16	
51	17	18	19	20	21	22	23	
52	24	25	26	27	28	29	30	
1	31	1	2	3	4	5	6	

Calendario Proyecto								
enero, 2013								
	lun	mar	mié	jue	vie	sáb	dom	
1	31	1	2	3	4	5	6	
2	7	8	9	10	11	12	13	
3	14	15	16	17	18	19	20	
4	21	22	23	24	25	26	27	
5	28	29	30	31	1	2	3	
6	4	5	6	7	8	9	10	

	Nombre	F. Inicio Temprano	F. Inicio Tardío	F. Fin Temprano	F. Fin Tardío	HT	HL	Sucesoras
13	25	29/01/2013	06/02/2013	04/02/2013	12/02/2013	6	0	26-321-45-361
14	26	13/02/2013	13/02/2013	18/02/2013	18/02/2013	0	0	31-321-351-361
15	31	19/02/2013	05/06/2013	26/02/2013	12/06/2013	74	74	
16	321	19/02/2013	19/02/2013	26/02/2013	26/02/2013	0	0	322
17	322	27/02/2013	27/02/2013	01/03/2013	01/03/2013	0	0	34
18	34	04/03/2013	04/03/2013	11/03/2013	11/03/2013	0	0	37

Calendario Proyecto

	febrero_ 2013						
	lun	mar	mié	jue	vie	sáb	dom
5	28	29	30	31	1	2	3
6	4	5	6	7	8	9	10
7	11	12	13	14	15	16	17
8	18	19	20	21	22	23	24
9	25	26	27	28	1	2	3
10	4	5	6	7	8	9	10

Duración: 124 días laborables

	Nombre	F. Inicio Temprano	F. Inicio Tardío	F. Fin Temprano	F. Fin Tardío	HT	HL	Sucesoras
19	351	19/02/2013	22/02/2013	21/02/2013	26/02/2013	3	0	352
20	352	22/02/2013	27/02/2013	26/02/2013	01/03/2013	3	3	34
21	361	19/02/2013	03/06/2013	26/02/2013	10/06/2013	72	0	362
22	362	27/02/2013	11/06/2013	28/02/2013	12/06/2013	72	72	
23	37	12/03/2013	12/03/2013	14/03/2013	14/03/2013	0	0	41-42-43-44
24	41	14/05/2013	14/05/2013	16/05/2013	16/05/2013	0	0	46

Calendario Proyecto

	marzo_ 2013						
	lun	mar	mié	jue	vie	sáb	dom
9	25	26	27	28	1	2	3
10	4	5	6	7	8	9	10
11	11	12	13	14	15	16	17
12	18	19	20	21	22	23	24
13	25	26	27	28	29	30	31
14	1	2	3	4	5	6	7

Duración: 124 días laborables

	Nombre	F. Inicio Temprano	F. Inicio Tardío	F. Fin Temprano	F. Fin Tardío	HT	HL	Sucesoras
25	42	15/03/2013	15/03/2013	13/05/2013	13/05/2013	0	0	41
26	43	15/03/2013	29/04/2013	29/03/2013	13/05/2013	30	30	41
27	44	15/03/2013	03/05/2013	26/03/2013	13/05/2013	33	33	41
28	45	05/02/2013	07/05/2013	11/02/2013	13/05/2013	63	63	41
29	46	17/05/2013	17/05/2013	30/05/2013	30/05/2013	0	0	47
30	47	31/05/2013	31/05/2013	04/06/2013	04/06/2013	0	0	51

Calendario Proyecto

	abril_ 2013						
	lun	mar	mié	jue	vie	sáb	dom
13	25	26	27	28	29	30	31
14	1	2	3	4	5	6	7
15	8	9	10	11	12	13	14
16	15	16	17	18	19	20	21
17	22	23	24	25	26	27	28
18	29	30	1	2	3	4	5

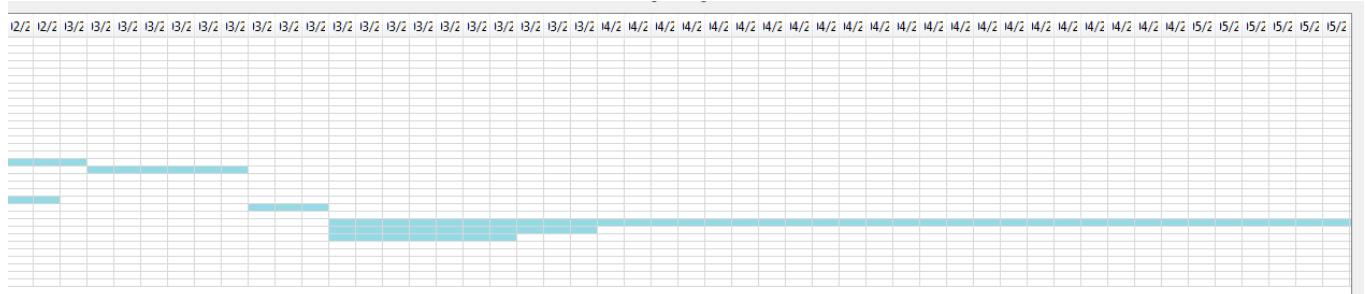
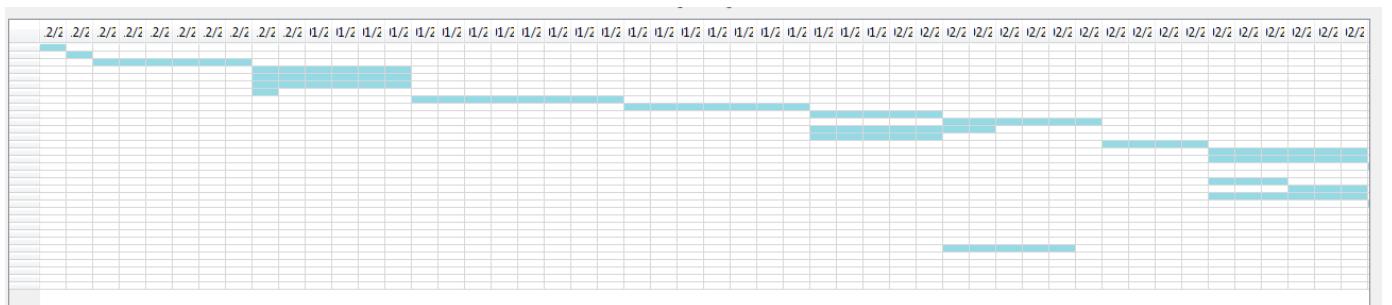
Duración: 124 días laborables

Nombre	F.Inicio Temprano	F.Inicio Tardío	F.Fin Temprano	F.Fin Tardío	HT	HL	Sucesoras
30_47	31/05/2013	31/05/2013	04/06/2013	04/06/2013	0	0	51
31_51	05/06/2013	05/06/2013	07/06/2013	07/06/2013	0	0	521
32_521	10/06/2013	10/06/2013	11/06/2013	11/06/2013	0	0	522
33_522	12/06/2013	12/06/2013	12/06/2013	12/06/2013	0	0	
34							
35							

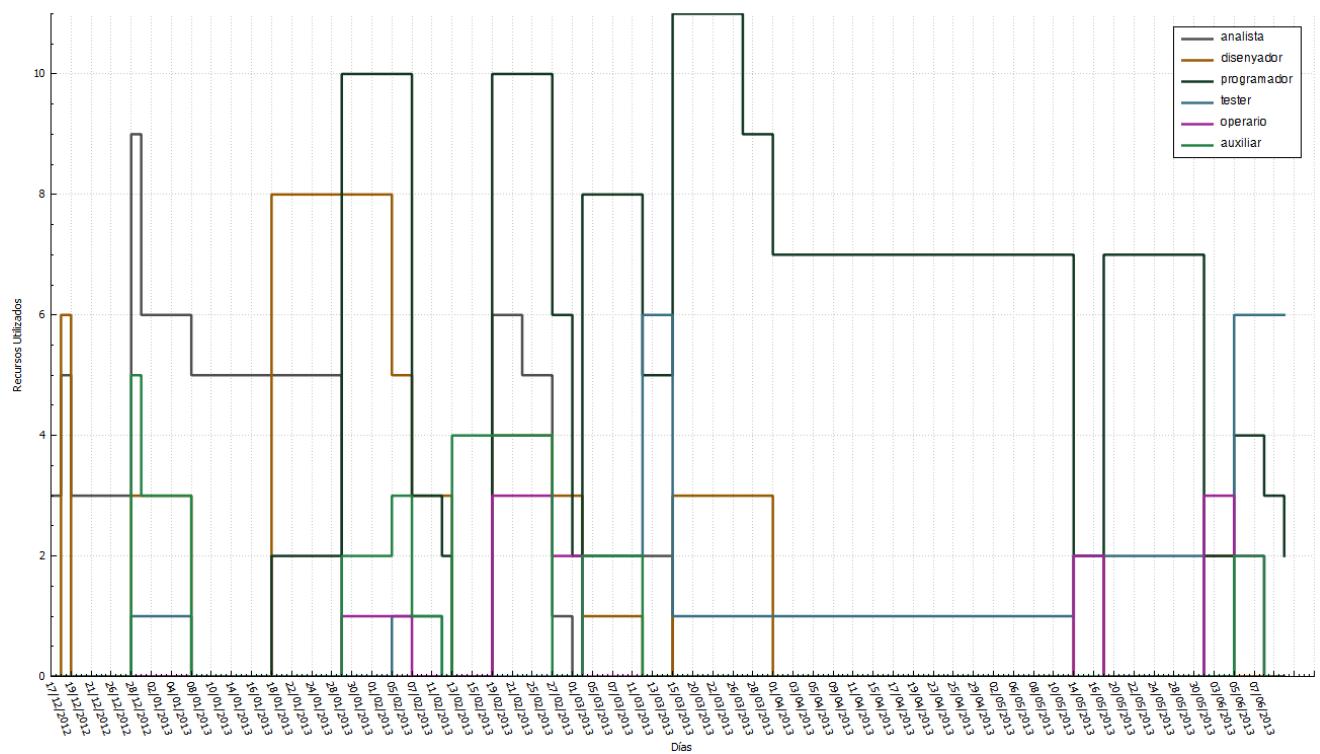
Calendario Proyecto

	lun	mar	mié	jue	vie	sáb	dom
18	29	30	1	2	3	4	5
19	6	7	8	9	10	11	12
20	13	14	15	16	17	18	19
21	20	21	22	23	24	25	26
22	27	28	29	30	31	1	2
23	3	4	5	6	7	8	9

5.2.1.2 Gantt



5.2.1.3 Histograma de recursos



5.2.2 Minima duración minimo coste

Informe Mínimo Coste

Informe Actividades

Camino Crítico: 11-12-131-132-136-21-22-23-26-321-322-34-37-41-46-47-51-521-522
 11-12-131-132-136-21-22-23-26-321-322-34-37-42-46-47-51-521-522
 11-12-131-133-136-21-22-23-26-321-322-34-37-41-46-47-51-521-522
 11-12-131-133-136-21-22-23-26-321-322-34-37-40-41-46-47-51-521-522
 11-12-131-134-136-21-22-23-26-321-322-34-37-41-46-47-51-521-522
 11-12-131-134-136-21-22-23-26-321-322-34-37-42-41-46-47-51-521-522

Fecha Inicio: 17/12/2012 Fecha fn: 03/06/2013

Duración: 117 días laborables

Sobrecosto: 35 euros

Nombre	F. Inicio Temprano	F. Inicio Tardío	F. Fin Temprano	F. Fin Tardío	HT	HL	Sucesoras
1 11	17/12/2012	17/12/2012	17/12/2012	17/12/2012	0	0	12-135
2 12	18/12/2012	18/12/2012	18/12/2012	18/12/2012	0	0	131
3 131	19/12/2012	19/12/2012	27/12/2012	27/12/2012	0	0	132-133-134-13...
4 132	28/12/2012	28/12/2012	03/01/2013	03/01/2013	0	0	136
5 133	28/12/2012	28/12/2012	03/01/2013	03/01/2013	0	0	136
6 134	28/12/2012	28/12/2012	03/01/2013	03/01/2013	0	0	136

Calendario Proyecto

diciembre 2012

	lun	mar	mié	jue	vie	sáb	dom
48	26	27	28	29	30	1	2
49	3	4	5	6	7	8	9
50	10	11	12	13	14	15	16
51	17	18	19	20	21	22	23
52	24	25	26	27	28	29	30
1	31	1	2	3	4	5	6





17/12/2012

Informe Mínimo Coste

Informe Actividades

Camino Crítico: 11-12-131-132-136-21-22-23-26-321-322-34-37-41-46-47-51-521-522
 11-12-131-132-136-21-22-23-26-321-322-34-37-42-46-47-51-521-522
 11-12-131-133-136-21-22-23-26-321-322-34-37-42-46-47-51-521-522
 11-12-131-133-136-21-22-23-26-321-322-34-37-41-46-47-51-521-522
 11-12-131-134-136-21-22-23-26-321-322-34-37-41-46-47-51-521-522

Fecha Inicio: 17/12/2012 Fecha fn: 03/06/2013

Duración: 117 días laborables

Sobrecosto: 35 euros

Nombre	F. Inicio Temprano	F. Inicio Tardío	F. Fin Temprano	F. Fin Tardío	HT	HL	Sucesoras
7 135	28/12/2012	03/01/2013	28/12/2012	03/01/2013	3	3	136
8 136	04/01/2013	04/01/2013	15/01/2013	15/01/2013	0	0	21-24
9 21	16/01/2013	16/01/2013	18/01/2013	18/01/2013	0	0	22-24-25
10 22	21/01/2013	21/01/2013	25/01/2013	25/01/2013	0	0	23
11 23	28/01/2013	02/02/2013	04/02/2013	04/02/2013	0	0	26
12 24	21/01/2013	25/01/2013	29/01/2013	04/02/2013	4	4	26-361

Calendario Proyecto

enero 2013

	lun	mar	mié	jue	vie	sáb	dom
1	31	1	2	3	4	5	6
2	7	8	9	10	11	12	13
3	14	15	16	17	18	19	20
4	21	22	23	24	25	26	27
5	28	29	30	31	1	2	3
6	4	5	6	7	8	9	10





17/12/2012

Informe Mínimo Coste

Informe Actividades

Camino Crítico: 11-12-131-132-136-21-22-23-26-321-322-34-37-41-46-47-51-521-522
 11-12-131-132-136-21-22-23-26-321-322-34-37-41-46-47-51-521-522
 11-12-131-133-136-21-22-23-26-321-322-34-37-41-46-47-51-521-522
 11-12-131-133-136-21-22-23-26-321-322-34-37-41-46-47-51-521-522
 11-12-131-134-136-21-22-23-26-321-322-34-37-41-46-47-51-521-522
 11-12-131-134-136-21-22-23-26-321-322-34-37-41-46-47-51-521-522

Fecha Inicio: 17/12/2012 Fecha Fin: 03/06/2013

Duración: 117 días laborables

Sobrecosto: 35 euros

Nombre	↓ Inicio Temprano	F. Inicio Tardío	F. Fin Temprano	F. Fin Tardío	HT	HL	Sucesoras
13_25	21/01/2013	29/01/2013	25/01/2013	04/02/2013	6	0	26-321-45-361
14_26	05/02/2013	05/02/2013	08/02/2013	08/02/2013	0	0	31-321-351-361
15_31	11/02/2013	27/05/2013	18/02/2013	03/06/2013	73	73	
16_321	11/02/2013	11/02/2013	18/02/2013	18/02/2013	0	0	322
17_322	19/02/2013	19/02/2013	21/02/2013	21/02/2013	0	0	34
18_34	22/02/2013	22/02/2013	01/03/2013	01/03/2013	0	0	37

Calendario Proyecto

febrero, 2013

	lun	mar	mié	jue	vie	sáb	dom
5	28	29	30	31	1	2	3
6	4	5	6	7	8	9	10
7	11	12	13	14	15	16	17
8	18	19	20	21	22	23	24
9	25	26	27	28	1	2	3
10	4	5	6	7	8	9	10

ES 2019 17/12/2012

Informe Mínimo Coste

Informe Actividades

Camino Crítico: 11-12-131-132-136-21-22-23-26-321-322-34-37-41-46-47-51-521-522
 11-12-131-132-136-21-22-23-26-321-322-34-37-41-46-47-51-521-522
 11-12-131-133-136-21-22-23-26-321-322-34-37-41-46-47-51-521-522
 11-12-131-133-136-21-22-23-26-321-322-34-37-41-46-47-51-521-522
 11-12-131-134-136-21-22-23-26-321-322-34-37-41-46-47-51-521-522
 11-12-131-134-136-21-22-23-26-321-322-34-37-41-46-47-51-521-522

Fecha Inicio: 17/12/2012 Fecha Fin: 03/06/2013

Duración: 117 días laborables

Sobrecosto: 35 euros

Nombre	↓ Inicio Temprano	F. Inicio Tardío	F. Fin Temprano	F. Fin Tardío	HT	HL	Sucesoras
19_351	11/02/2013	14/02/2013	13/02/2013	18/02/2013	3	0	352
20_352	14/02/2013	19/02/2013	18/02/2013	21/02/2013	3	3	34
21_361	11/02/2013	23/05/2013	18/02/2013	30/05/2013	71	0	362
22_362	19/02/2013	31/05/2013	20/02/2013	03/06/2013	71	71	
23_37	04/03/2013	04/03/2013	06/03/2013	06/03/2013	0	0	41-42-43-44
24_41	06/05/2013	06/05/2013	07/05/2013	07/05/2013	0	0	46

Calendario Proyecto

marzo, 2013

	lun	mar	mié	jue	vie	sáb	dom
9	25	26	27	28	1	2	3
10	4	5	6	7	8	9	10
11	11	12	13	14	15	16	17
12	18	19	20	21	22	23	24
13	25	26	27	28	29	30	31
14	1	2	3	4	5	6	7

ES 2019 17/12/2012

Informe Mínimo Coste

Informe Actividades

Camino Crítico: 11-12-131-132-136-21-22-23-26-321-322-34-37-41-46-47-51-521-522
 11-12-131-132-136-21-22-23-26-321-322-34-37-42-46-47-51-521-522
 11-12-131-133-136-21-22-23-26-321-322-34-37-41-46-47-51-521-522
 11-12-131-133-136-21-22-23-26-321-322-34-37-40-41-46-47-51-521-522
 11-12-131-134-136-21-22-23-26-321-322-34-37-40-41-46-47-51-521-522
 11-12-131-134-136-21-22-23-26-321-322-34-37-42-46-47-51-521-522

Fecha Inicio: 17/12/2012 Fecha Fin: 03/06/2013

Duración: 117 días laborables

Sobrecosto: 35 euros

Nombre	F. Inicio Temprano	F. Inicio Tardío	F. Fin Temprano	F. Fin Tardío	HT	HL	Sucesoras
25_42	07/03/2013	07/03/2013	03/05/2013	03/05/2013	0	0	41
26_43	07/03/2013	19/04/2013	21/03/2013	03/05/2013	30	30	41
27_44	07/03/2013	24/04/2013	15/03/2013	03/05/2013	33	33	41
28_45	28/01/2013	26/04/2013	01/02/2013	03/05/2013	63	63	41
29_46	08/05/2013	08/05/2013	21/05/2013	21/05/2013	0	0	47
30_47	22/05/2013	24/05/2013	24/05/2013	24/05/2013	0	0	51

Calendario Proyecto

abril 2013

	lun	mar	mié	jue	vie	sáb	dom
13	25	26	27	28	29	30	31
14	1	2	3	4	5	6	7
15	8	9	10	11	12	13	14
16	15	16	17	18	19	20	21
17	22	23	24	25	26	27	28
18	29	30	1	2	3	4	5

17/12/2012

Informe Mínimo Coste

Informe Actividades

Camino Crítico: 11-12-131-132-136-21-22-23-26-321-322-34-37-41-46-47-51-521-522
 11-12-131-132-136-21-22-23-26-321-322-34-37-42-46-47-51-521-522
 11-12-131-133-136-21-22-23-26-321-322-34-37-42-46-47-51-521-522
 11-12-131-133-136-21-22-23-26-321-322-34-37-41-46-47-51-521-522
 11-12-131-134-136-21-22-23-26-321-322-34-37-40-41-46-47-51-521-522

Fecha Inicio: 17/12/2012 Fecha Fin: 03/06/2013

Duración: 117 días laborables

Sobrecosto: 35 euros

Nombre	F. Inicio Temprano	F. Inicio Tardío	F. Fin Temprano	F. Fin Tardío	HT	HL	Sucesoras
29_46	08/05/2013	08/05/2013	21/05/2013	21/05/2013	0	0	47
30_47	22/05/2013	22/05/2013	24/05/2013	24/05/2013	0	0	51
31_51	27/05/2013	27/05/2013	29/05/2013	29/05/2013	0	0	521
32_521	30/05/2013	30/05/2013	31/05/2013	31/05/2013	0	0	522
33_522	03/06/2013	03/06/2013	03/06/2013	03/06/2013	0	0	
34							

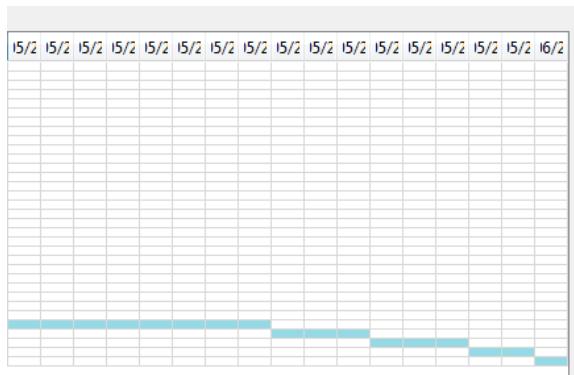
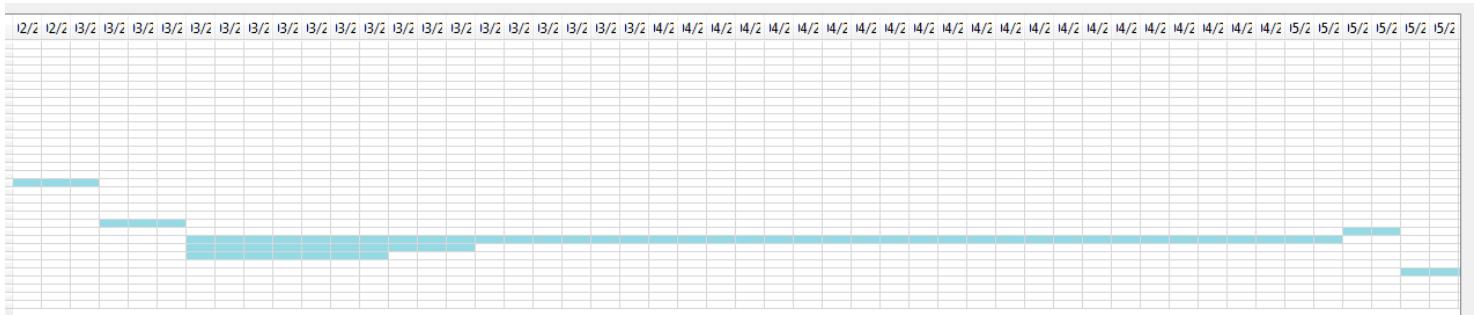
Calendario Proyecto

mayo 2013

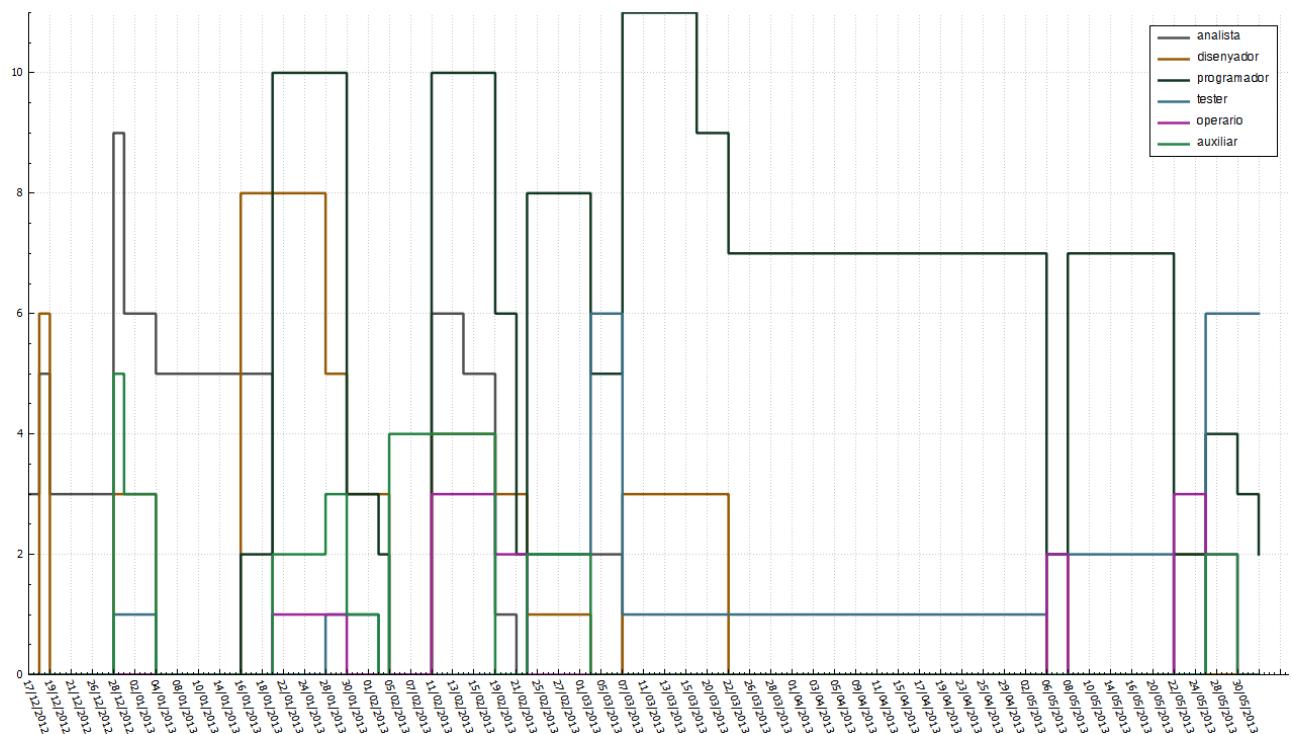
	lun	mar	mié	jue	vie	sáb	dom
18	29	30	1	2	3	4	5
19	6	7	8	9	10	11	12
20	13	14	15	16	17	18	19
21	20	21	22	23	24	25	26
22	27	28	29	30	31	1	2
23	3	4	5	6	7	8	9

17/12/2012

5.2.2.1 Gantt



5.2.2.2 Histograma de recursos



5.2.2.3 Informe debug

```
*****INICIALIZACION*****
Caminos:
 0°: 11 12 131 132 136 21 22 23 26 31 50
 1°: 11 12 131 132 136 21 22 23 26 321 322 34 37 41 46 47 51 521 522 84
 2°: 11 12 131 132 136 21 22 23 26 321 322 34 37 42 41 46 47 51 521 522 124
 3°: 11 12 131 132 136 21 22 23 26 321 322 34 37 43 41 46 47 51 521 522 94
 4°: 11 12 131 132 136 21 22 23 26 321 322 34 37 44 41 46 47 51 521 522 91
 5°: 11 12 131 132 136 21 22 23 26 351 352 34 37 41 46 47 51 521 522 81
 6°: 11 12 131 132 136 21 22 23 26 351 352 34 37 42 41 46 47 51 521 522 121
 7°: 11 12 131 132 136 21 22 23 26 351 352 34 37 43 41 46 47 51 521 522 91
 8°: 11 12 131 132 136 21 22 23 26 351 352 34 37 44 41 46 47 51 521 522 88
 9°: 11 12 131 132 136 21 22 23 26 361 362 52
10°: 11 12 131 132 136 21 24 26 31 46
11°: 11 12 131 132 136 21 24 26 321 322 34 37 41 46 47 51 521 522 80
12°: 11 12 131 132 136 21 24 26 321 322 34 37 42 41 46 47 51 521 522 120
13°: 11 12 131 132 136 21 24 26 321 322 34 37 43 41 46 47 51 521 522 90
14°: 11 12 131 132 136 21 24 26 321 322 34 37 44 41 46 47 51 521 522 87
15°: 11 12 131 132 136 21 24 26 351 352 34 37 41 46 47 51 521 522 77
16°: 11 12 131 132 136 21 24 26 351 352 34 37 42 41 46 47 51 521 522 117
17°: 11 12 131 132 136 21 24 26 351 352 34 37 43 41 46 47 51 521 522 87
18°: 11 12 131 132 136 21 24 26 351 352 34 37 44 41 46 47 51 521 522 84
19°: 11 12 131 132 136 21 24 26 361 362 48
20°: 11 12 131 132 136 21 24 361 362 44
21°: 11 12 131 132 136 21 25 26 31 44
22°: 11 12 131 132 136 21 25 26 321 322 34 37 41 46 47 51 521 522 78
23°: 11 12 131 132 136 21 25 26 321 322 34 37 42 41 46 47 51 521 522 118
24°: 11 12 131 132 136 21 25 26 321 322 34 37 43 41 46 47 51 521 522 88
25°: 11 12 131 132 136 21 25 26 321 322 34 37 44 41 46 47 51 521 522 85
26°: 11 12 131 132 136 21 25 26 351 352 34 37 41 46 47 51 521 522 75
27°: 11 12 131 132 136 21 25 26 351 352 34 37 42 41 46 47 51 521 522 115
28°: 11 12 131 132 136 21 25 26 351 352 34 37 43 41 46 47 51 521 522 85
29°: 11 12 131 132 136 21 25 26 351 352 34 37 44 41 46 47 51 521 522 82
30°: 11 12 131 132 136 21 25 26 361 362 46
31°: 11 12 131 132 136 21 25 321 322 34 37 41 46 47 51 521 522 74
32°: 11 12 131 132 136 21 25 321 322 34 37 42 41 46 47 51 521 522 114
33°: 11 12 131 132 136 21 25 321 322 34 37 43 41 46 47 51 521 522 84
34°: 11 12 131 132 136 21 25 321 322 34 37 44 41 46 47 51 521 522 81
35°: 11 12 131 132 136 21 25 45 41 46 47 51 521 522 61
36°: 11 12 131 132 136 21 25 361 362 42
37°: 11 12 131 132 136 24 26 31 39
38°: 11 12 131 132 136 24 26 321 322 34 37 41 46 47 51 521 522 73
39°: 11 12 131 132 136 24 26 321 322 34 37 42 41 46 47 51 521 522 113
40°: 11 12 131 132 136 24 26 321 322 34 37 43 41 46 47 51 521 522 83
41°: 11 12 131 132 136 24 26 321 322 34 37 44 41 46 47 51 521 522 80
42°: 11 12 131 132 136 24 26 351 352 34 37 41 46 47 51 521 522 70
43°: 11 12 131 132 136 24 26 351 352 34 37 42 41 46 47 51 521 522 110
44°: 11 12 131 132 136 24 26 351 352 34 37 43 41 46 47 51 521 522 80
45°: 11 12 131 132 136 24 26 351 352 34 37 44 41 46 47 51 521 522 77
46°: 11 12 131 132 136 24 26 361 362 41
47°: 11 12 131 132 136 24 361 362 37
48°: 11 12 131 133 136 21 22 23 26 31 50
49°: 11 12 131 133 136 21 22 23 26 321 322 34 37 41 46 47 51 521 522 84
50°: 11 12 131 133 136 21 22 23 26 321 322 34 37 42 41 46 47 51 521 522 124
51°: 11 12 131 133 136 21 22 23 26 321 322 34 37 43 41 46 47 51 521 522 94
52°: 11 12 131 133 136 21 22 23 26 321 322 34 37 44 41 46 47 51 521 522 91
53°: 11 12 131 133 136 21 22 23 26 351 352 34 37 41 46 47 51 521 522 81
54°: 11 12 131 133 136 21 22 23 26 351 352 34 37 42 41 46 47 51 521 522 121
55°: 11 12 131 133 136 21 22 23 26 351 352 34 37 43 41 46 47 51 521 522 91
56°: 11 12 131 133 136 21 22 23 26 351 352 34 37 44 41 46 47 51 521 522 88
57°: 11 12 131 133 136 21 22 23 26 361 362 52
58°: 11 12 131 133 136 21 24 26 31 46
59°: 11 12 131 133 136 21 24 26 321 322 34 37 41 46 47 51 521 522 80
60°: 11 12 131 133 136 21 24 26 321 322 34 37 42 41 46 47 51 521 522 120
61°: 11 12 131 133 136 21 24 26 321 322 34 37 43 41 46 47 51 521 522 90
62°: 11 12 131 133 136 21 24 26 321 322 34 37 44 41 46 47 51 521 522 87
63°: 11 12 131 133 136 21 24 26 351 352 34 37 41 46 47 51 521 522 77
64°: 11 12 131 133 136 21 24 26 351 352 34 37 42 41 46 47 51 521 522 117
65°: 11 12 131 133 136 21 24 26 351 352 34 37 43 41 46 47 51 521 522 87
66°: 11 12 131 133 136 21 24 26 351 352 34 37 44 41 46 47 51 521 522 84
67°: 11 12 131 133 136 21 24 26 361 362 48
68°: 11 12 131 133 136 21 24 361 362 44
69°: 11 12 131 133 136 21 25 26 31 44
70°: 11 12 131 133 136 21 25 26 321 322 34 37 41 46 47 51 521 522 78
71°: 11 12 131 133 136 21 25 26 321 322 34 37 42 41 46 47 51 521 522 118
72°: 11 12 131 133 136 21 25 26 321 322 34 37 43 41 46 47 51 521 522 88
73°: 11 12 131 133 136 21 25 26 321 322 34 37 44 41 46 47 51 521 522 85
74°: 11 12 131 133 136 21 25 26 351 352 34 37 41 46 47 51 521 522 75
75°: 11 12 131 133 136 21 25 26 351 352 34 37 42 41 46 47 51 521 522 115
76°: 11 12 131 133 136 21 25 26 351 352 34 37 43 41 46 47 51 521 522 85
77°: 11 12 131 133 136 21 25 26 351 352 34 37 44 41 46 47 51 521 522 82
78°: 11 12 131 133 136 21 25 26 361 362 46
79°: 11 12 131 133 136 21 25 321 322 34 37 41 46 47 51 521 522 74
80°: 11 12 131 133 136 21 25 321 322 34 37 42 41 46 47 51 521 522 114
81°: 11 12 131 133 136 21 25 321 322 34 37 43 41 46 47 51 521 522 84
82°: 11 12 131 133 136 21 25 321 322 34 37 44 41 46 47 51 521 522 81
```

83°: 11 12 131 133 136 21 25 45 41 46 47 51 521 522 61
 84°: 11 12 131 133 136 21 25 361 362 42
 85°: 11 12 131 133 136 24 26 31 39
 86°: 11 12 131 133 136 24 26 321 322 34 37 41 46 47 51 521 522 73
 87°: 11 12 131 133 136 24 26 321 322 34 37 42 41 46 47 51 521 522 113
 88°: 11 12 131 133 136 24 26 321 322 34 37 43 41 46 47 51 521 522 83
 89°: 11 12 131 133 136 24 26 321 322 34 37 44 41 46 47 51 521 522 80
 90°: 11 12 131 133 136 24 26 351 352 34 37 41 46 47 51 521 522 70
 91°: 11 12 131 133 136 24 26 351 352 34 37 42 41 46 47 51 521 522 110
 92°: 11 12 131 133 136 24 26 351 352 34 37 43 41 46 47 51 521 522 80
 93°: 11 12 131 133 136 24 26 351 352 34 37 44 41 46 47 51 521 522 77
 94°: 11 12 131 133 136 24 26 361 362 41
 95°: 11 12 131 133 136 24 361 362 37
 96°: 11 12 131 134 136 21 22 23 26 31 50
 97°: 11 12 131 134 136 21 22 23 26 321 322 34 37 41 46 47 51 521 522 84
 98°: 11 12 131 134 136 21 22 23 26 321 322 34 37 42 41 46 47 51 521 522 124
 99°: 11 12 131 134 136 21 22 23 26 321 322 34 37 43 41 46 47 51 521 522 94
 100°: 11 12 131 134 136 21 22 23 26 321 322 34 37 44 41 46 47 51 521 522 91
 101°: 11 12 131 134 136 21 22 23 26 351 352 34 37 41 46 47 51 521 522 81
 102°: 11 12 131 134 136 21 22 23 26 351 352 34 37 42 41 46 47 51 521 522 121
 103°: 11 12 131 134 136 21 22 23 26 351 352 34 37 43 41 46 47 51 521 522 91
 104°: 11 12 131 134 136 21 22 23 26 351 352 34 37 44 41 46 47 51 521 522 88
 105°: 11 12 131 134 136 21 22 23 26 361 362 52
 106°: 11 12 131 134 136 21 24 26 31 46
 107°: 11 12 131 134 136 21 24 26 321 322 34 37 41 46 47 51 521 522 80
 108°: 11 12 131 134 136 21 24 26 321 322 34 37 42 41 46 47 51 521 522 120
 109°: 11 12 131 134 136 21 24 26 321 322 34 37 43 41 46 47 51 521 522 90
 110°: 11 12 131 134 136 21 24 26 321 322 34 37 44 41 46 47 51 521 522 87
 111°: 11 12 131 134 136 21 24 26 351 352 34 37 41 46 47 51 521 522 77
 112°: 11 12 131 134 136 21 24 26 351 352 34 37 42 41 46 47 51 521 522 117
 113°: 11 12 131 134 136 21 24 26 351 352 34 37 43 41 46 47 51 521 522 87
 114°: 11 12 131 134 136 21 24 26 351 352 34 37 44 41 46 47 51 521 522 84
 115°: 11 12 131 134 136 21 24 26 361 362 48
 116°: 11 12 131 134 136 21 24 361 362 44
 117°: 11 12 131 134 136 21 25 26 31 44
 118°: 11 12 131 134 136 21 25 26 321 322 34 37 41 46 47 51 521 522 78
 119°: 11 12 131 134 136 21 25 26 321 322 34 37 42 41 46 47 51 521 522 118
 120°: 11 12 131 134 136 21 25 26 321 322 34 37 43 41 46 47 51 521 522 88
 121°: 11 12 131 134 136 21 25 26 321 322 34 37 44 41 46 47 51 521 522 85
 122°: 11 12 131 134 136 21 25 26 351 352 34 37 41 46 47 51 521 522 75
 123°: 11 12 131 134 136 21 25 26 351 352 34 37 42 41 46 47 51 521 522 115
 124°: 11 12 131 134 136 21 25 26 351 352 34 37 43 41 46 47 51 521 522 85
 125°: 11 12 131 134 136 21 25 26 351 352 34 37 44 41 46 47 51 521 522 82
 126°: 11 12 131 134 136 21 25 26 361 362 46
 127°: 11 12 131 134 136 21 25 321 322 34 37 41 46 47 51 521 522 74
 128°: 11 12 131 134 136 21 25 321 322 34 37 42 41 46 47 51 521 522 114
 129°: 11 12 131 134 136 21 25 321 322 34 37 43 41 46 47 51 521 522 84
 130°: 11 12 131 134 136 21 25 321 322 34 37 44 41 46 47 51 521 522 81
 131°: 11 12 131 134 136 21 25 45 41 46 47 51 521 522 61
 132°: 11 12 131 134 136 21 25 361 362 42
 133°: 11 12 131 134 136 24 26 31 39
 134°: 11 12 131 134 136 24 26 321 322 34 37 41 46 47 51 521 522 73
 135°: 11 12 131 134 136 24 26 321 322 34 37 42 41 46 47 51 521 522 113
 136°: 11 12 131 134 136 24 26 321 322 34 37 43 41 46 47 51 521 522 83
 137°: 11 12 131 134 136 24 26 321 322 34 37 44 41 46 47 51 521 522 80
 138°: 11 12 131 134 136 24 26 351 352 34 37 41 46 47 51 521 522 70
 139°: 11 12 131 134 136 24 26 351 352 34 37 42 41 46 47 51 521 522 110
 140°: 11 12 131 134 136 24 26 351 352 34 37 43 41 46 47 51 521 522 80
 141°: 11 12 131 134 136 24 26 351 352 34 37 44 41 46 47 51 521 522 77
 142°: 11 12 131 134 136 24 26 361 362 41
 143°: 11 12 131 134 136 24 361 362 37
 144°: 11 12 131 135 136 21 22 23 26 31 45
 145°: 11 12 131 135 136 21 22 23 26 321 322 34 37 41 46 47 51 521 522 79
 146°: 11 12 131 135 136 21 22 23 26 321 322 34 37 42 41 46 47 51 521 522 119
 147°: 11 12 131 135 136 21 22 23 26 321 322 34 37 43 41 46 47 51 521 522 89
 148°: 11 12 131 135 136 21 22 23 26 321 322 34 37 44 41 46 47 51 521 522 86
 149°: 11 12 131 135 136 21 22 23 26 351 352 34 37 41 46 47 51 521 522 76
 150°: 11 12 131 135 136 21 22 23 26 351 352 34 37 42 41 46 47 51 521 522 116
 151°: 11 12 131 135 136 21 22 23 26 351 352 34 37 43 41 46 47 51 521 522 86
 152°: 11 12 131 135 136 21 22 23 26 351 352 34 37 44 41 46 47 51 521 522 83
 153°: 11 12 131 135 136 21 22 23 26 361 362 47
 154°: 11 12 131 135 136 21 24 26 31 41
 155°: 11 12 131 135 136 21 24 26 321 322 34 37 41 46 47 51 521 522 75
 156°: 11 12 131 135 136 21 24 26 321 322 34 37 42 41 46 47 51 521 522 115
 157°: 11 12 131 135 136 21 24 26 321 322 34 37 43 41 46 47 51 521 522 85
 158°: 11 12 131 135 136 21 24 26 321 322 34 37 44 41 46 47 51 521 522 82
 159°: 11 12 131 135 136 21 24 26 351 352 34 37 41 46 47 51 521 522 72
 160°: 11 12 131 135 136 21 24 26 351 352 34 37 42 41 46 47 51 521 522 112
 161°: 11 12 131 135 136 21 24 26 351 352 34 37 43 41 46 47 51 521 522 82
 162°: 11 12 131 135 136 21 24 26 351 352 34 37 44 41 46 47 51 521 522 79
 163°: 11 12 131 135 136 21 24 26 361 362 43
 164°: 11 12 131 135 136 21 24 361 362 39
 165°: 11 12 131 135 136 21 25 26 31 39
 166°: 11 12 131 135 136 21 25 26 321 322 34 37 41 46 47 51 521 522 73
 167°: 11 12 131 135 136 21 25 26 321 322 34 37 42 41 46 47 51 521 522 113
 168°: 11 12 131 135 136 21 25 26 321 322 34 37 43 41 46 47 51 521 522 83
 169°: 11 12 131 135 136 21 25 26 321 322 34 37 44 41 46 47 51 521 522 80
 170°: 11 12 131 135 136 21 25 26 351 352 34 37 41 46 47 51 521 522 70

171°: 11 12 131 135 136 21 25 26 351 352 34 37 42 41 46 47 51 521 522 110
172°: 11 12 131 135 136 21 25 26 351 352 34 37 43 41 46 47 51 521 522 80
173°: 11 12 131 135 136 21 25 26 351 352 34 37 44 41 46 47 51 521 522 77
174°: 11 12 131 135 136 21 25 26 361 362 41
175°: 11 12 131 135 136 21 25 321 322 34 37 41 46 47 51 521 522 69
176°: 11 12 131 135 136 21 25 321 322 34 37 42 41 46 47 51 521 522 109
177°: 11 12 131 135 136 21 25 321 322 34 37 43 41 46 47 51 521 522 79
178°: 11 12 131 135 136 21 25 321 322 34 37 44 41 46 47 51 521 522 76
179°: 11 12 131 135 136 21 25 45 41 46 47 51 521 522 56
180°: 11 12 131 135 136 21 25 361 362 37
181°: 11 12 131 135 136 24 26 31 34
182°: 11 12 131 135 136 24 26 321 322 34 37 41 46 47 51 521 522 68
183°: 11 12 131 135 136 24 26 321 322 34 37 42 41 46 47 51 521 522 108
184°: 11 12 131 135 136 24 26 321 322 34 37 43 41 46 47 51 521 522 78
185°: 11 12 131 135 136 24 26 321 322 34 37 44 41 46 47 51 521 522 75
186°: 11 12 131 135 136 24 26 351 352 34 37 41 46 47 51 521 522 65
187°: 11 12 131 135 136 24 26 351 352 34 37 42 41 46 47 51 521 522 105
188°: 11 12 131 135 136 24 26 351 352 34 37 43 41 46 47 51 521 522 75
189°: 11 12 131 135 136 24 26 351 352 34 37 44 41 46 47 51 521 522 72
190°: 11 12 131 135 136 24 26 361 362 36
191°: 11 12 131 135 136 24 361 362 32
192°: 11 12 131 31 14
193°: 11 135 136 21 22 23 26 31 38
194°: 11 135 136 21 22 23 26 321 322 34 37 41 46 47 51 521 522 72
195°: 11 135 136 21 22 23 26 321 322 34 37 42 41 46 47 51 521 522 112
196°: 11 135 136 21 22 23 26 321 322 34 37 43 41 46 47 51 521 522 82
197°: 11 135 136 21 22 23 26 321 322 34 37 44 41 46 47 51 521 522 79
198°: 11 135 136 21 22 23 26 351 352 34 37 41 46 47 51 521 522 69
199°: 11 135 136 21 22 23 26 351 352 34 37 42 41 46 47 51 521 522 109
200°: 11 135 136 21 22 23 26 351 352 34 37 43 41 46 47 51 521 522 79
201°: 11 135 136 21 22 23 26 351 352 34 37 44 41 46 47 51 521 522 76
202°: 11 135 136 21 22 23 26 361 362 40
203°: 11 135 136 21 24 26 31 34
204°: 11 135 136 21 24 26 321 322 34 37 41 46 47 51 521 522 68
205°: 11 135 136 21 24 26 321 322 34 37 42 41 46 47 51 521 522 108
206°: 11 135 136 21 24 26 321 322 34 37 43 41 46 47 51 521 522 78
207°: 11 135 136 21 24 26 321 322 34 37 44 41 46 47 51 521 522 75
208°: 11 135 136 21 24 26 351 352 34 37 41 46 47 51 521 522 65
209°: 11 135 136 21 24 26 351 352 34 37 42 41 46 47 51 521 522 105
210°: 11 135 136 21 24 26 351 352 34 37 43 41 46 47 51 521 522 75
211°: 11 135 136 21 24 26 351 352 34 37 44 41 46 47 51 521 522 72
212°: 11 135 136 21 24 26 361 362 36
213°: 11 135 136 21 24 361 362 32
214°: 11 135 136 21 25 26 31 32
215°: 11 135 136 21 25 26 321 322 34 37 41 46 47 51 521 522 66
216°: 11 135 136 21 25 26 321 322 34 37 42 41 46 47 51 521 522 106
217°: 11 135 136 21 25 26 321 322 34 37 43 41 46 47 51 521 522 76
218°: 11 135 136 21 25 26 321 322 34 37 44 41 46 47 51 521 522 73
219°: 11 135 136 21 25 26 351 352 34 37 41 46 47 51 521 522 63
220°: 11 135 136 21 25 26 351 352 34 37 42 41 46 47 51 521 522 103
221°: 11 135 136 21 25 26 351 352 34 37 43 41 46 47 51 521 522 73
222°: 11 135 136 21 25 26 351 352 34 37 44 41 46 47 51 521 522 70
223°: 11 135 136 21 25 26 361 362 34
224°: 11 135 136 21 25 321 322 34 37 41 46 47 51 521 522 62
225°: 11 135 136 21 25 321 322 34 37 42 41 46 47 51 521 522 102
226°: 11 135 136 21 25 321 322 34 37 43 41 46 47 51 521 522 72
227°: 11 135 136 21 25 321 322 34 37 44 41 46 47 51 521 522 69
228°: 11 135 136 21 25 45 41 46 47 51 521 522 49
229°: 11 135 136 21 25 361 362 30
230°: 11 135 136 24 26 31 27
231°: 11 135 136 24 26 321 322 34 37 41 46 47 51 521 522 61
232°: 11 135 136 24 26 321 322 34 37 42 41 46 47 51 521 522 101
233°: 11 135 136 24 26 321 322 34 37 43 41 46 47 51 521 522 71
234°: 11 135 136 24 26 321 322 34 37 44 41 46 47 51 521 522 68
235°: 11 135 136 24 26 351 352 34 37 41 46 47 51 521 522 58
236°: 11 135 136 24 26 351 352 34 37 42 41 46 47 51 521 522 98
237°: 11 135 136 24 26 351 352 34 37 43 41 46 47 51 521 522 68
238°: 11 135 136 24 26 351 352 34 37 44 41 46 47 51 521 522 65
239°: 11 135 136 24 26 361 362 29
240°: 11 135 136 24 361 362 25
T.Normal-T.tope 11 = 0
T.Normal-T.tope 12 = 0
T.Normal-T.tope 131 = 0
T.Normal-T.tope 132 = 2
T.Normal-T.tope 133 = 3
T.Normal-T.tope 134 = 2
T.Normal-T.tope 135 = 0
T.Normal-T.tope 136 = 0
T.Normal-T.tope 21 = 4
T.Normal-T.tope 22 = 0
T.Normal-T.tope 23 = 0
T.Normal-T.tope 24 = 2
T.Normal-T.tope 25 = 0
T.Normal-T.tope 26 = 0
T.Normal-T.tope 31 = 2
T.Normal-T.tope 321 = 0
T.Normal-T.tope 322 = 0

```

T.Normal-T.tope 34 = 0
T.Normal-T.tope 351 = 0
T.Normal-T.tope 352 = 0
T.Normal-T.tope 361 = 2
T.Normal-T.tope 362 = 0
T.Normal-T.tope 37 = 0
T.Normal-T.tope 41 = 1
T.Normal-T.tope 42 = 0
T.Normal-T.tope 43 = 0
T.Normal-T.tope 44 = 0
T.Normal-T.tope 45 = 2
T.Normal-T.tope 46 = 0
T.Normal-T.tope 47 = 0
T.Normal-T.tope 51 = 0
T.Normal-T.tope 521 = 0
T.Normal-T.tope 522 = 0

***** EJECUCION *****

Maxima duracion a reducir = 124
Sobrecoste Minimo = 16
Actividades a modificar: 21

Decremento de dias de los caminos = 4
-----
Maxima duracion a reducir = 120
Sobrecoste Minimo = 12
Actividades a modificar: 132 133 134

Decremento de dias de los caminos = 2
-----
Maxima duracion a reducir = 118
Sobrecoste Minimo = 7
Actividades a modificar: 41

Decremento de dias de los caminos = 1
-----
Maxima duracion a reducir = 117

----- CAMINO CRITICO-----
Actividad:11 Tmin:0 Tmax:0
Actividad:12 Tmin:1 Tmax:1
Actividad:131 Tmin:2 Tmax:2
Actividad:132 Tmin:8 Tmax:8
Actividad:133 Tmin:8 Tmax:8
Actividad:134 Tmin:8 Tmax:8
Actividad:135 Tmin:8 Tmax:11
Actividad:136 Tmin:12 Tmax:12
Actividad:21 Tmin:20 Tmax:20
Actividad:22 Tmin:23 Tmax:23
Actividad:23 Tmin:28 Tmax:28
Actividad:24 Tmin:23 Tmax:27
Actividad:25 Tmin:23 Tmax:29
Actividad:26 Tmin:34 Tmax:34
Actividad:31 Tmin:38 Tmax:111
Actividad:321 Tmin:38 Tmax:38
Actividad:322 Tmin:44 Tmax:44
Actividad:34 Tmin:47 Tmax:47
Actividad:351 Tmin:38 Tmax:41
Actividad:352 Tmin:41 Tmax:44
Actividad:361 Tmin:38 Tmax:109
Actividad:362 Tmin:44 Tmax:115
Actividad:37 Tmin:53 Tmax:53
Actividad:41 Tmin:96 Tmax:96
Actividad:42 Tmin:56 Tmax:56
Actividad:43 Tmin:56 Tmax:86
Actividad:44 Tmin:56 Tmax:89
Actividad:45 Tmin:28 Tmax:91
Actividad:46 Tmin:98 Tmax:98
Actividad:47 Tmin:108 Tmax:108
Actividad:51 Tmin:111 Tmax:111
Actividad:521 Tmin:114 Tmax:114
Actividad:522 Tmin:116 Tmax:116

```

5.2.3 Limitación con esquema paralelo y regla NSUC

5.2.3.1 Informe

Informe Limitación de Recursos

Informe Actividades

Camino Crítico: 11-12-131-132-136-21-22-23-26-321-322-34-37-41-46-47-51-521-522
11-12-131-132-136-21-22-23-26-321-322-34-37-42-41-46-47-51-521-522
11-12-131-133-136-21-22-23-26-321-322-34-37-41-46-47-51-521-522
11-12-131-134-136-21-22-23-26-321-322-34-37-41-46-47-51-521-522
11-12-131-134-136-21-22-23-26-321-322-34-37-41-46-47-51-521-522
11-12-131-134-136-21-22-23-26-321-322-34-37-42-41-46-47-51-521-522

Fecha Inicio: 17/12/2012 Fecha fin: 10/06/2013

Duración: 122 días laborables

Sobrecoste: 35 euros

Nombre	F. Inicio Temprano	F. Inicio Tardío	F. Fin Temprano	F. Fin Tardío	HT	HL	Sucesoras
1 11	17/12/2012	24/12/2012	17/12/2012	24/12/2012	5	0	12-135
2 12	18/12/2012	26/12/2012	18/12/2012	26/12/2012	5	0	131
3 131	19/12/2012	27/12/2012	27/12/2012	04/01/2013	5	0	132-133-134-13...
4 132	28/12/2012	07/01/2013	03/01/2013	10/01/2013	5	5	136
5 133	28/12/2012	07/01/2013	03/01/2013	10/01/2013	5	5	136
6 134	04/01/2013	07/01/2013	09/01/2013	10/01/2013	1	1	136

Calendario Proyecto

diciembre 2012

	lun	mar	mié	jue	vie	sáb	dom
48	26	27	28	29	30	1	2
49	3	4	5	6	7	8	9
50	10	11	12	13	14	15	16
51	17	18	19	20	21	22	23
52	24	25	26	27	28	29	30
1	31	1	2	3	4	5	6

ES 2022 17/12/2012

Informe Limitación de Recursos

Informe Actividades

Camino Crítico: 11-12-131-132-136-21-22-23-26-321-322-34-37-41-46-47-51-521-522
11-12-131-132-136-21-22-23-26-321-322-34-37-42-41-46-47-51-521-522
11-12-131-133-136-21-22-23-26-321-322-34-37-42-41-46-47-51-521-522
11-12-131-134-136-21-22-23-26-321-322-34-37-41-46-47-51-521-522
11-12-131-134-136-21-22-23-26-321-322-34-37-42-41-46-47-51-521-522

Fecha Inicio: 17/12/2012 Fecha fin: 10/06/2013

Duración: 122 días laborables

Sobrecoste: 35 euros

Nombre	F. Inicio Temprano	F. Inicio Tardío	F. Fin Temprano	F. Fin Tardío	HT	HL	Sucesoras
7 135	10/01/2013	10/01/2013	10/01/2013	10/01/2013	0	0	136
8 136	11/01/2013	11/01/2013	22/01/2013	22/01/2013	0	0	21-24
9 21	23/01/2013	23/01/2013	25/01/2013	25/01/2013	0	0	22-24-25
10 22	28/01/2013	28/01/2013	01/02/2013	01/02/2013	0	0	23
11 23	04/02/2013	04/02/2013	11/02/2013	11/02/2013	0	0	26
12 24	28/01/2013	01/02/2013	05/02/2013	11/02/2013	4	4	26,361

Calendario Proyecto

enero 2013

	lun	mar	mié	jue	vie	sáb	dom
1	31	1	2	3	4	5	6
2	7	8	9	10	11	12	13
3	14	15	16	17	18	19	20
4	21	22	23	24	25	26	27
5	28	29	30	31	1	2	3
6	4	5	6	7	8	9	10

ES 20-22 17/12/2012

Informe Limitación de Recursos

Informe Actividades

Camino Crítico: 11-12-131-132-136-21-22-23-26-321-322-34-37-41-46-47-51-521-522
 11-12-131-132-136-21-22-23-26-321-322-34-37-41-46-47-51-521-522
 11-12-131-133-136-21-22-23-26-321-322-34-37-41-46-47-51-521-522
 11-12-131-133-136-21-22-23-26-321-322-34-37-41-46-47-51-521-522
 11-12-131-134-136-21-22-23-26-321-322-34-37-41-46-47-51-521-522
 11-12-131-134-136-21-22-23-26-321-322-34-37-41-46-47-51-521-522

Fecha Inicio: 17/12/2012 Fecha fin: 10/06/2013

Duración: 122 días laborables

Sobrecoste:35 euros

Nombre	F. Inicio Temprano	F. Inicio Tardío	F. Fin Temprano	F. Fin Tardío	HT	HL	Sucesoras
13_25	28/01/2013	05/02/2013	01/02/2013	11/02/2013	6	0	26-321-45-361
14_26	12/02/2013	12/02/2013	15/02/2013	15/02/2013	0	0	31-321-351-361
15_31	18/02/2013	03/06/2013	25/02/2013	10/06/2013	73	73	
16_321	18/02/2013	18/02/2013	25/02/2013	25/02/2013	0	0	322
17_322	26/02/2013	26/02/2013	28/02/2013	28/02/2013	0	0	34
18_34	01/03/2013	01/03/2013	08/03/2013	08/03/2013	0	0	37

Calendario Proyecto

febrero, 2013

	lun	mar	mié	jue	vie	sáb	dom
5	28	29	30	31	1	2	3
6	4	5	6	7	8	9	10
7	11	12	13	14	15	16	17
8	18	19	20	21	22	23	24
9	25	26	27	28	1	2	3
10	4	5	6	7	8	9	10

ES 20:22 17/12/2012

Informe Limitación de Recursos

Informe Actividades

Camino Crítico: 11-12-131-132-136-21-22-23-26-321-322-34-37-41-46-47-51-521-522
 11-12-131-132-136-21-22-23-26-321-322-34-37-42-41-46-47-51-521-522
 11-12-131-133-136-21-22-23-26-321-322-34-37-41-46-47-51-521-522
 11-12-131-133-136-21-22-23-26-321-322-34-37-41-46-47-51-521-522
 11-12-131-134-136-21-22-23-26-321-322-34-37-41-46-47-51-521-522
 11-12-131-134-136-21-22-23-26-321-322-34-37-42-41-46-47-51-521-522

Fecha Inicio: 17/12/2012 Fecha fin: 10/06/2013

Duración: 122 días laborables

Sobrecoste:35 euros

Nombre	F. Inicio Temprano	F. Inicio Tardío	F. Fin Temprano	F. Fin Tardío	HT	HL	Sucesoras
19_351	18/02/2013	21/02/2013	20/02/2013	25/02/2013	3	0	352
20_352	21/02/2013	26/02/2013	25/02/2013	28/02/2013	3	3	34
21_361	21/02/2013	30/05/2013	28/02/2013	06/06/2013	68	0	362
22_362	01/03/2013	07/06/2013	04/03/2013	10/06/2013	68	68	
23_37	11/03/2013	11/03/2013	13/03/2013	13/03/2013	0	0	41-42-43-44
24_41	13/05/2013	13/05/2013	14/05/2013	14/05/2013	0	0	46

Calendario Proyecto

marzo, 2013

	lun	mar	mié	jue	vie	sáb	dom
9	25	26	27	28	1	2	3
10	4	5	6	7	8	9	10
11	11	12	13	14	15	16	17
12	18	19	20	21	22	23	24
13	25	26	27	28	29	30	31
14	1	2	3	4	5	6	7

ES 20:22 17/12/2012

Informe Limitación de Recursos

Informe Actividades

Camino Crítico: 11-12-131-132-136-21-22-23-26-321-322-34-37-41-46-47-51-521-522
 11-12-131-132-136-21-22-23-26-321-322-34-37-42-46-47-51-521-522
 11-12-131-133-136-21-22-23-26-321-322-34-37-41-46-47-51-521-522
 11-12-131-133-136-21-22-23-26-321-322-34-37-40-41-46-47-51-521-522
 11-12-131-134-136-21-22-23-26-321-322-34-37-41-46-47-51-521-522
 11-12-131-134-136-21-22-23-26-321-322-34-37-42-46-47-51-521-522

Fecha Inicio: 17/12/2012 Fecha fin: 10/06/2013

Duración: 122 días laborables

Sobrecoste: 35 euros

Nombre	F. Inicio Temprano	F. Inicio Tardío	F. Fin Temprano	F. Fin Tardío	HT	HL	Sucesoras
25_42	14/03/2013	14/03/2013	10/05/2013	10/05/2013	0	0	41
26_43	14/03/2013	26/04/2013	28/03/2013	10/05/2013	30	30	41
27_44	14/03/2013	02/05/2013	25/03/2013	10/05/2013	33	33	41
28_45	04/02/2013	06/05/2013	08/02/2013	10/05/2013	63	63	41
29_46	15/05/2013	15/05/2013	28/05/2013	28/05/2013	0	0	47
30_47	29/05/2013	29/05/2013	31/05/2013	31/05/2013	0	0	51

Calendario Proyecto

	lun	mar	mié	jue	vie	sáb	dom
13	25	26	27	28	29	30	31
14	1	2	3	4	5	6	7
15	8	9	10	11	12	13	14
16	15	16	17	18	19	20	21
17	22	23	24	25	26	27	28
18	29	30	1	2	3	4	5

ES 20-23 17/12/2012

Informe Limitación de Recursos

Informe Actividades

Camino Crítico: 11-12-131-132-136-21-22-23-26-321-322-34-37-41-46-47-51-521-522
 11-12-131-132-136-21-22-23-26-321-322-34-37-42-46-47-51-521-522
 11-12-131-133-136-21-22-23-26-321-322-34-37-41-46-47-51-521-522
 11-12-131-133-136-21-22-23-26-321-322-34-37-42-46-47-51-521-522
 11-12-131-134-136-21-22-23-26-321-322-34-37-41-46-47-51-521-522
 Fecha Inicio: 17/12/2012 Fecha fin: 10/06/2013

Duración: 122 días laborables

Sobrecoste: 35 euros

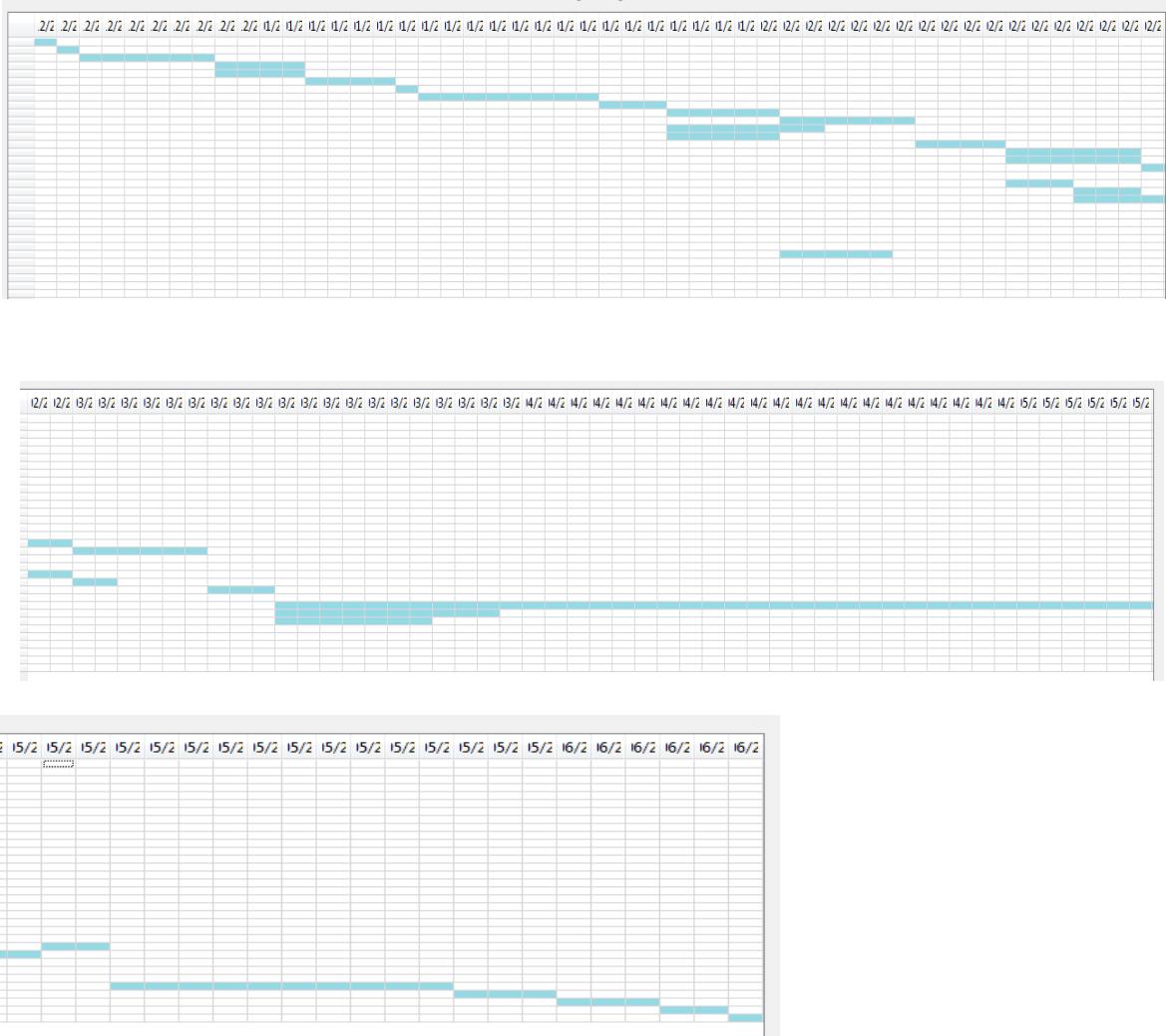
Nombre	F. Inicio Temprano	F. Inicio Tardío	F. Fin Temprano	F. Fin Tardío	HT	HL	Sucesoras
29_46	15/05/2013	15/05/2013	28/05/2013	28/05/2013	0	0	47
30_47	29/05/2013	29/05/2013	31/05/2013	31/05/2013	0	0	51
31_51	03/06/2013	03/06/2013	05/06/2013	05/06/2013	0	0	521
32_521	06/06/2013	06/06/2013	07/06/2013	07/06/2013	0	0	522
33_522	10/06/2013	10/06/2013	10/06/2013	10/06/2013	0	0	
34							

Calendario Proyecto

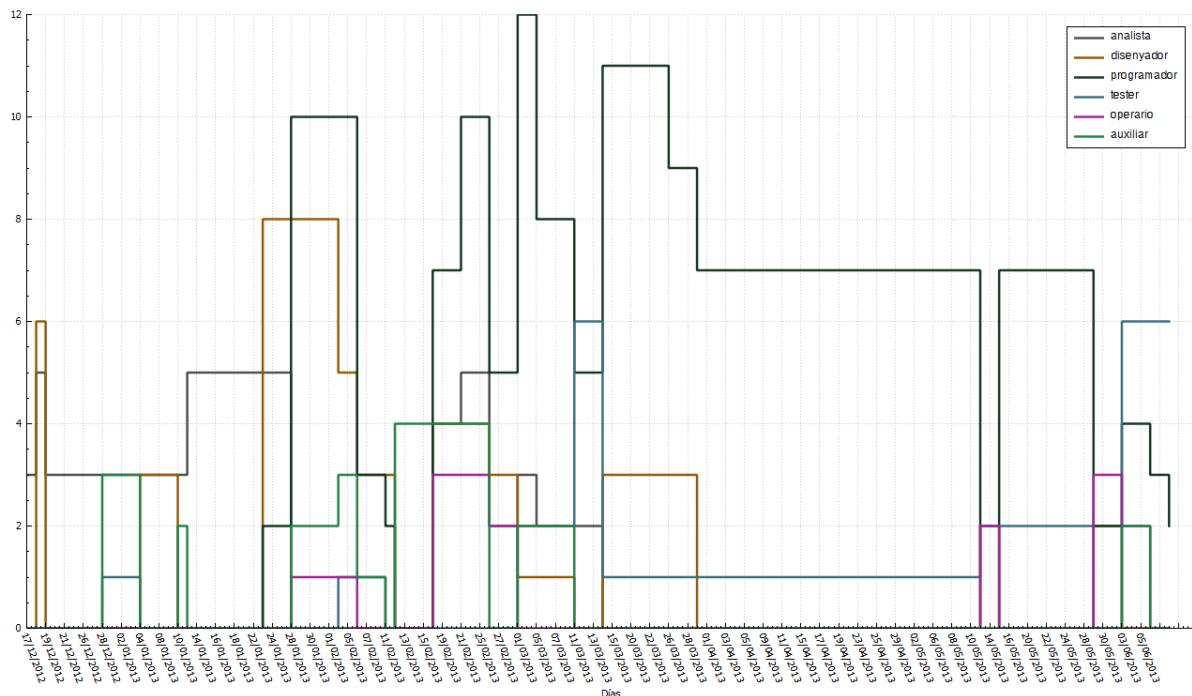
	lun	mar	mié	jue	vie	sáb	dom
18	29	30	1	2	3	4	5
19	6	7	8	9	10	11	12
20	13	14	15	16	17	18	19
21	20	21	22	23	24	25	26
22	27	28	29	30	31	1	2
23	3	4	5	6	7	8	9

ES 20-23 17/12/2012

5.2.3.3 Gantt



5.2.3.4 Histograma recursos



5.2.3.5 Informe debug

```
*****PARALELO*****
ITERACION __T=0
    disp0{ analista=5 disenayador=8 programador=13 tester=6 operario=8 auxiliar=5}
    elegir={11 }
    Se secuencia 11 en 0 y termina en 1
    disp0{ analista=8 disenayador=8 programador=13 tester=6 operario=8 auxiliar=5}

ITERACION __T=1
    La actividad 11 termina de procesarse
    disp1{ analista=5 disenayador=8 programador=13 tester=6 operario=8 auxiliar=5}
    elegir={12 }
    Se secuencia 12 en 1 y termina en 2
    disp1{ analista=0 disenayador=2 programador=13 tester=6 operario=8 auxiliar=5}

ITERACION __T=2
    La actividad 12 termina de procesarse
    disp2{ analista=5 disenayador=8 programador=13 tester=6 operario=8 auxiliar=5}
    elegir={131 }
    Se secuencia 131 en 2 y termina en 8
    disp2{ analista=2 disenayador=8 programador=13 tester=6 operario=8 auxiliar=5}

ITERACION __T=8
    La actividad 131 termina de procesarse
    disp8{ analista=5 disenayador=8 programador=13 tester=6 operario=8 auxiliar=5}
    elegir={132 133 134 135 }
    Se secuencia 132 en 8 y termina en 12
    Se secuencia 133 en 8 y termina en 12
    disp8{ analista=2 disenayador=8 programador=13 tester=5 operario=8 auxiliar=2}

ITERACION __T=12
    La actividad 132 termina de procesarse
    La actividad 133 termina de procesarse
    disp12{ analista=5 disenayador=8 programador=13 tester=6 operario=8 auxiliar=5}
    elegir={134 135 }
    Se secuencia 134 en 12 y termina en 16
    disp12{ analista=2 disenayador=5 programador=13 tester=6 operario=8 auxiliar=5}
```

```

ITERACION __ T=16
La actividad 134 termina de procesarse
disp16{ analista=5 disenador=8 programador=13 tester=6 operario=8 auxiliar=5}
elegir={135 }
Se secuencia 135 en 16 y termina en 17
disp16{ analista=2 disenador=8 programador=13 tester=6 operario=8 auxiliar=3}

ITERACION __ T=17
La actividad 135 termina de procesarse
disp17{ analista=5 disenador=8 programador=13 tester=6 operario=8 auxiliar=5}
elegir={136 }
Se secuencia 136 en 17 y termina en 25
disp17{ analista=0 disenador=8 programador=13 tester=6 operario=8 auxiliar=5}

ITERACION __ T=25
La actividad 136 termina de procesarse
disp25{ analista=5 disenador=8 programador=13 tester=6 operario=8 auxiliar=5}
elegir={21 }
Se secuencia 21 en 25 y termina en 28
disp25{ analista=0 disenador=0 programador=11 tester=6 operario=8 auxiliar=5}

ITERACION __ T=28
La actividad 21 termina de procesarse
disp28{ analista=5 disenador=8 programador=13 tester=6 operario=8 auxiliar=5}
elegir={25 24 22 }
Se secuencia 25 en 28 y termina en 33
Se secuencia 24 en 28 y termina en 35
Se secuencia 22 en 28 y termina en 33
disp28{ analista=5 disenador=0 programador=3 tester=6 operario=7 auxiliar=3}

ITERACION __ T=33
La actividad 25 termina de procesarse
La actividad 22 termina de procesarse
disp33{ analista=5 disenador=6 programador=6 tester=6 operario=7 auxiliar=3}
elegir={45 23 }
Se secuencia 45 en 33 y termina en 38
Se secuencia 23 en 33 y termina en 39
disp33{ analista=5 disenador=3 programador=3 tester=5 operario=7 auxiliar=2}

ITERACION __ T=35
La actividad 24 termina de procesarse
disp35{ analista=5 disenador=5 programador=10 tester=5 operario=8 auxiliar=4}
elegir={}
disp35{ analista=5 disenador=5 programador=10 tester=5 operario=8 auxiliar=4}

ITERACION __ T=38
La actividad 45 termina de procesarse
disp38{ analista=5 disenador=5 programador=11 tester=6 operario=8 auxiliar=5}
elegir={}
disp38{ analista=5 disenador=5 programador=11 tester=6 operario=8 auxiliar=5}

ITERACION __ T=39
La actividad 23 termina de procesarse
disp39{ analista=5 disenador=8 programador=13 tester=6 operario=8 auxiliar=5}
elegir={26 }
Se secuencia 26 en 39 y termina en 43
disp39{ analista=5 disenador=8 programador=13 tester=6 operario=8 auxiliar=1}

ITERACION __ T=43
La actividad 26 termina de procesarse
disp43{ analista=5 disenador=8 programador=13 tester=6 operario=8 auxiliar=5}
elegir={31 321 351 361 }
Se secuencia 31 en 43 y termina en 49
Se secuencia 321 en 43 y termina en 49
Se secuencia 351 en 43 y termina en 46
disp43{ analista=1 disenador=4 programador=6 tester=6 operario=5 auxiliar=1}

ITERACION __ T=46
La actividad 351 termina de procesarse
disp46{ analista=3 disenador=4 programador=8 tester=6 operario=5 auxiliar=1}
elegir={361 352 }
Se secuencia 361 en 46 y termina en 52
Se secuencia 352 en 46 y termina en 49
disp46{ analista=0 disenador=4 programador=3 tester=6 operario=5 auxiliar=1}

ITERACION __ T=49
La actividad 31 termina de procesarse

```

```

La actividad 321 termina de procesarse
La actividad 352 termina de procesarse
disp49{ analista=3 disenayador=8 programador=10 tester=6 operario=8 auxiliar=5}
elegir={322 }
Se secuencia 322 en 49 y termina en 52
disp49{ analista=3 disenayador=5 programador=8 tester=6 operario=6 auxiliar=5}

ITERACION __ T=52
La actividad 361 termina de procesarse
La actividad 322 termina de procesarse
disp52{ analista=5 disenayador=8 programador=13 tester=6 operario=8 auxiliar=5}
elegir={362 34 }
Se secuencia 362 en 52 y termina en 54
Se secuencia 34 en 52 y termina en 58
disp52{ analista=2 disenayador=7 programador=1 tester=4 operario=8 auxiliar=3}

ITERACION __ T=54
La actividad 362 termina de procesarse
disp54{ analista=3 disenayador=7 programador=5 tester=4 operario=8 auxiliar=3}
elegir={}
disp54{ analista=3 disenayador=7 programador=5 tester=4 operario=8 auxiliar=3}

ITERACION __ T=58
La actividad 34 termina de procesarse
disp58{ analista=5 disenayador=8 programador=13 tester=6 operario=8 auxiliar=5}
elegir={37 }
Se secuencia 37 en 58 y termina en 61
disp58{ analista=3 disenayador=8 programador=8 tester=0 operario=8 auxiliar=5}

ITERACION __ T=61
La actividad 37 termina de procesarse
disp61{ analista=5 disenayador=8 programador=13 tester=6 operario=8 auxiliar=5}
elegir={42 43 44 }
Se secuencia 42 en 61 y termina en 101
Se secuencia 43 en 61 y termina en 71
Se secuencia 44 en 61 y termina en 68
disp61{ analista=5 disenayador=5 programador=2 tester=5 operario=8 auxiliar=5}

ITERACION __ T=68
La actividad 44 termina de procesarse
disp68{ analista=5 disenayador=5 programador=4 tester=5 operario=8 auxiliar=5}
elegir={}
disp68{ analista=5 disenayador=5 programador=4 tester=5 operario=8 auxiliar=5}

ITERACION __ T=71
La actividad 43 termina de procesarse
disp71{ analista=5 disenayador=8 programador=6 tester=5 operario=8 auxiliar=5}
elegir={}
disp71{ analista=5 disenayador=8 programador=6 tester=5 operario=8 auxiliar=5}

ITERACION __ T=101
La actividad 42 termina de procesarse
disp101{ analista=5 disenayador=8 programador=13 tester=6 operario=8 auxiliar=5}
elegir={41 }
Se secuencia 41 en 101 y termina en 103
disp101{ analista=3 disenayador=6 programador=11 tester=4 operario=6 auxiliar=5}

ITERACION __ T=103
La actividad 41 termina de procesarse
disp103{ analista=5 disenayador=8 programador=13 tester=6 operario=8 auxiliar=5}
elegir={46 }
Se secuencia 46 en 103 y termina en 113
disp103{ analista=5 disenayador=8 programador=6 tester=4 operario=8 auxiliar=5}

ITERACION __ T=113
La actividad 46 termina de procesarse
disp113{ analista=5 disenayador=8 programador=13 tester=6 operario=8 auxiliar=5}
elegir={47 }
Se secuencia 47 en 113 y termina en 116
disp113{ analista=5 disenayador=6 programador=11 tester=6 operario=5 auxiliar=5}

ITERACION __ T=116
La actividad 47 termina de procesarse
disp116{ analista=5 disenayador=8 programador=13 tester=6 operario=8 auxiliar=5}
elegir={51 }
Se secuencia 51 en 116 y termina en 119
disp116{ analista=5 disenayador=8 programador=9 tester=0 operario=6 auxiliar=3}

```

```
ITERACION __T=119_____
La actividad 51 termina de procesarse
disp119{ analista=5 disenayador=8 programador=13 tester=6 operario=8 auxiliar=5}
elegir={521 }
Se secuencia 521 en 119 y termina en 121
disp119{ analista=5 disenayador=8 programador=10 tester=0 operario=8 auxiliar=5}

ITERACION __T=121_____
La actividad 521 termina de procesarse
disp121{ analista=5 disenayador=8 programador=13 tester=6 operario=8 auxiliar=5}
elegir={522 }
Se secuencia 522 en 121 y termina en 122
disp121{ analista=5 disenayador=8 programador=11 tester=0 operario=8 auxiliar=5}

ITERACION __T=122_____
La actividad 522 termina de procesarse
disp122{ analista=5 disenayador=8 programador=13 tester=6 operario=8 auxiliar=5}
elegir={}
disp122{ analista=5 disenayador=8 programador=13 tester=6 operario=8 auxiliar=5}
```

5.2.4 Retraso con esquema paralelo y adelanto con esquema serie

5.2.4.1 Informe

Informe Atraso-Adelanto

Informe Actividades

Camino Crítico: 11-12-131-132-136-21-22-26-31-322-34-37-41-46-47-51-521-522
 11-12-131-133-136-21-22-23-26-321-322-34-37-41-46-47-51-521-522
 11-12-131-133-136-21-22-23-26-321-322-34-37-41-46-47-51-521-522
 11-12-131-134-136-21-22-23-26-321-322-34-37-41-46-47-51-521-522
 11-12-131-134-136-21-22-23-26-321-322-34-37-41-46-47-51-521-522

Fecha Inicio: 17/12/2012 Fecha Fin: 07/06/2013

Duración: 121 días laborables

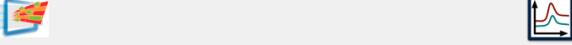
Sobrecosto: 35 euros

Nombre	F. Inicio Temprano	F. Inicio Tardío	F. Fin Temprano	F. Fin Tardío	HT	HL	Sucesoras
1 11	17/12/2012	21/12/2012	17/12/2012	21/12/2012	4	0	12-135
2 12	18/12/2012	24/12/2012	18/12/2012	24/12/2012	4	0	131
3 131	19/12/2012	26/12/2012	27/12/2012	03/01/2013	4	0	132-133-134-13...
4 132	28/12/2012	04/01/2013	03/01/2013	09/01/2013	4	4	136
5 133	04/01/2013	04/01/2013	03/01/2013	09/01/2013	0	0	136
6 134	28/12/2012	04/01/2013	03/01/2013	09/01/2013	4	4	136

Calendario Proyecto

diciembre, 2012

	lun	mar	mié	jue	vie	sáb	dom
48	26	27	28	29	30	1	2
49	3	4	5	6	7	8	9
50	10	11	12	13	14	15	16
51	17	18	19	20	21	22	23
52	24	25	26	27	28	29	30
1	31	1	2	3	4	5	6



20:27 17/12/2012

Informe Atraso-Adelanto

Informe Actividades

Camino Crítico: 11-12-131-132-136-21-22-26-321-322-34-37-41-46-47-51-521-522
 11-12-131-132-136-21-22-23-26-321-322-34-37-41-46-47-51-521-522
 11-12-131-133-136-21-22-23-26-321-322-34-37-41-46-47-51-521-522
 11-12-131-133-136-21-22-23-26-321-322-34-37-41-46-47-51-521-522
 11-12-131-134-136-21-22-23-26-321-322-34-37-41-46-47-51-521-522

Fecha Inicio: 17/12/2012 Fecha Fin: 07/06/2013

Duración: 121 días laborables

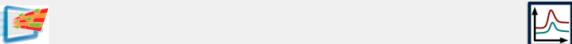
Sobrecosto: 35 euros

Nombre	F. Inicio Temprano	F. Inicio Tardío	F. Fin Temprano	F. Fin Tardío	HT	HL	Sucesoras
7 135	04/01/2013	09/01/2013	04/01/2013	09/01/2013	3	3	136
8 136	10/01/2013	10/01/2013	21/01/2013	21/01/2013	0	0	21-24
9 21	22/01/2013	22/01/2013	24/01/2013	24/01/2013	0	0	22-24-25
10 22	25/01/2013	25/01/2013	31/01/2013	31/01/2013	0	0	23
11 23	01/02/2013	01/02/2013	08/02/2013	08/02/2013	0	0	26
12 24	25/01/2013	31/01/2013	04/02/2013	08/02/2013	4	4	26-361

Calendario Proyecto

enero, 2013

	lun	mar	mié	jue	vie	sáb	dom
1	31	1	2	3	4	5	6
2	7	8	9	10	11	12	13
3	14	15	16	17	18	19	20
4	21	22	23	24	25	26	27
5	28	29	30	31	1	2	3
6	4	5	6	7	8	9	10



20:27 17/12/2012

Informe Atraso-Adelanto

Informe Actividades

Camino Crítico: 11-12-131-132-136-21-22-23-26-321-322-34-37-41-46-47-51-521-522
 11-12-131-132-136-21-22-23-26-321-322-34-37-42-46-47-51-521-522
 11-12-131-133-136-21-22-23-26-321-322-34-37-41-46-47-51-521-522
 11-12-131-133-136-21-22-23-26-321-322-34-37-40-41-46-47-51-521-522
 11-12-131-134-136-21-22-23-26-321-322-34-37-40-46-47-51-521-522
 11-12-131-134-136-21-22-23-26-321-322-34-37-42-46-47-51-521-522

Fecha Inicio: 17/12/2012 Fecha fin: 07/06/2013

Duración: 121 días laborables

Sobrecoste: 35 euros

Nombre	F. Inicio Temprano	F. Inicio Tardío	F. Fin Temprano	F. Fin Tardío	HT	HL	Sucesoras
13_25	25/01/2013	04/02/2013	31/01/2013	08/02/2013	6	0	26-321-45-361
14_26	11/02/2013	11/02/2013	14/02/2013	14/02/2013	0	0	31-321-351-361
15_31	15/02/2013	31/05/2013	22/02/2013	07/06/2013	73	73	
16_321	15/02/2013	15/02/2013	22/02/2013	22/02/2013	0	0	322
17_322	25/02/2013	25/02/2013	27/02/2013	27/02/2013	0	0	34
18_34	28/02/2013	28/02/2013	07/03/2013	07/03/2013	0	0	37

Calendario Proyecto

	lun	mar	mié	jue	vie	sáb	dom
5	28	29	30	31	1	2	3
6	4	5	6	7	8	9	10
7	11	12	13	14	15	16	17
8	18	19	20	21	22	23	24
9	25	26	27	28	1	2	3
10	4	5	6	7	8	9	10

20-28
17/12/2012

Informe Atraso-Adelanto

Informe Actividades

Camino Crítico: 11-12-131-132-136-21-22-23-26-321-322-34-37-41-46-47-51-521-522
 11-12-131-132-136-21-22-23-26-321-322-34-37-42-46-47-51-521-522
 11-12-131-133-136-21-22-23-26-321-322-34-37-41-46-47-51-521-522
 11-12-131-133-136-21-22-23-26-321-322-34-37-40-41-46-47-51-521-522
 11-12-131-134-136-21-22-23-26-321-322-34-37-40-46-47-51-521-522
 11-12-131-134-136-21-22-23-26-321-322-34-37-42-46-47-51-521-522

Fecha Inicio: 17/12/2012 Fecha fin: 07/06/2013

Duración: 121 días laborables

Sobrecoste: 35 euros

Nombre	F. Inicio Temprano	F. Inicio Tardío	F. Fin Temprano	F. Fin Tardío	HT	HL	Sucesoras
19_351	15/02/2013	20/02/2013	19/02/2013	22/02/2013	3	0	352
20_352	20/02/2013	25/02/2013	22/02/2013	27/02/2013	3	3	34
21_361	20/02/2013	29/05/2013	27/02/2013	05/06/2013	68	0	362
22_362	28/02/2013	06/06/2013	01/03/2013	07/06/2013	68	68	
23_37	08/03/2013	08/03/2013	12/03/2013	12/03/2013	0	0	41-42-43-44
24_41	10/05/2013	10/05/2013	13/05/2013	13/05/2013	0	0	46

Calendario Proyecto

	lun	mar	mié	jue	vie	sáb	dom
9	25	26	27	28	1	2	3
10	4	5	6	7	8	9	10
11	11	12	13	14	15	16	17
12	18	19	20	21	22	23	24
13	25	26	27	28	29	30	31
14	1	2	3	4	5	6	7

20-28
17/12/2012

Informe Atraso-Adelanto

Informe Actividades

Camino Crítico: 11-12-131-132-136-21-22-23-26-321-322-34-37-41-46-47-51-521-522
 11-12-131-132-136-21-22-23-26-321-322-34-37-42-46-47-51-521-522
 11-12-131-133-136-21-22-23-26-321-322-34-37-41-46-47-51-521-522
 11-12-131-133-136-21-22-23-26-321-322-34-37-42-46-47-51-521-522
 11-12-131-134-136-21-22-23-26-321-322-34-37-41-46-47-51-521-522
 11-12-131-134-136-21-22-23-26-321-322-34-37-42-46-47-51-521-522

Fecha Inicio: 17/12/2012 Fecha fin: 07/06/2013

Duración: 121 días laborables

Sobrecoste: 35 euros

Nombre	F. Inicio Temprano	F. Inicio Tardío	F. Fin Temprano	F. Fin Tardío	HT	HL	Sucesoras
25_42	13/03/2013	13/03/2013	09/05/2013	09/05/2013	0	0	41
26_43	13/03/2013	25/04/2013	27/03/2013	09/05/2013	30	30	41
27_44	13/03/2013	30/04/2013	22/03/2013	09/05/2013	33	33	41
28_45	01/02/2013	03/05/2013	07/02/2013	09/05/2013	63	63	41
29_46	14/05/2013	14/05/2013	27/05/2013	27/05/2013	0	0	47
30_47	28/05/2013	28/05/2013	30/05/2013	30/05/2013	0	0	51

Calendario Proyecto

abril 2013

	lun	mar	mié	jue	vie	sáb	dom
13	25	26	27	28	29	30	31
14	1	2	3	4	5	6	7
15	8	9	10	11	12	13	14
16	15	16	17	18	19	20	21
17	22	23	24	25	26	27	28
18	29	30	1	2	3	4	5

ES

20:28

17/12/2012

Informe Atraso-Adelanto

Informe Actividades

Camino Crítico: 11-12-131-132-136-21-22-23-26-321-322-34-37-41-46-47-51-521-522
 11-12-131-132-136-21-22-23-26-321-322-34-37-42-46-47-51-521-522
 11-12-131-133-136-21-22-23-26-321-322-34-37-41-46-47-51-521-522
 11-12-131-133-136-21-22-23-26-321-322-34-37-42-46-47-51-521-522
 11-12-131-134-136-21-22-23-26-321-322-34-37-41-46-47-51-521-522
 Fecha Inicio: 17/12/2012 Fecha fin: 07/06/2013

Duración: 121 días laborables

Sobrecoste: 35 euros

Nombre	F. Inicio Temprano	F. Inicio Tardío	F. Fin Temprano	F. Fin Tardío	HT	HL	Sucesoras
30_47	28/05/2013	28/05/2013	30/05/2013	30/05/2013	0	0	51
31_51	31/05/2013	31/05/2013	04/06/2013	04/06/2013	0	0	521
32_521	05/06/2013	05/06/2013	06/06/2013	06/06/2013	0	0	522
33_522	07/06/2013	07/06/2013	07/06/2013	07/06/2013	0	0	
34							
35							

Calendario Proyecto

mayo 2013

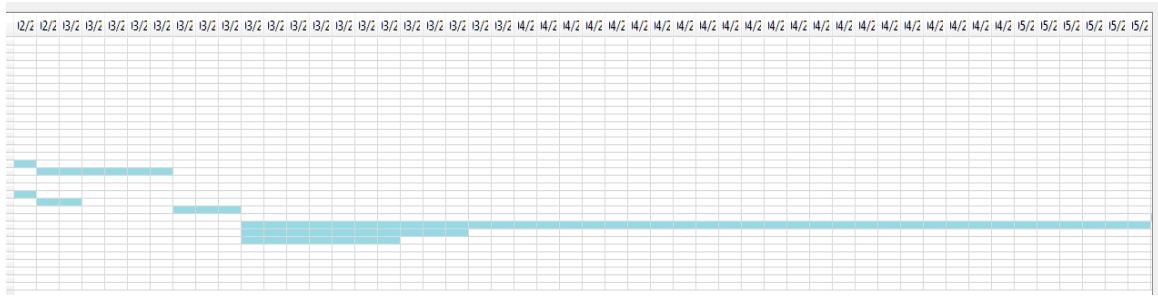
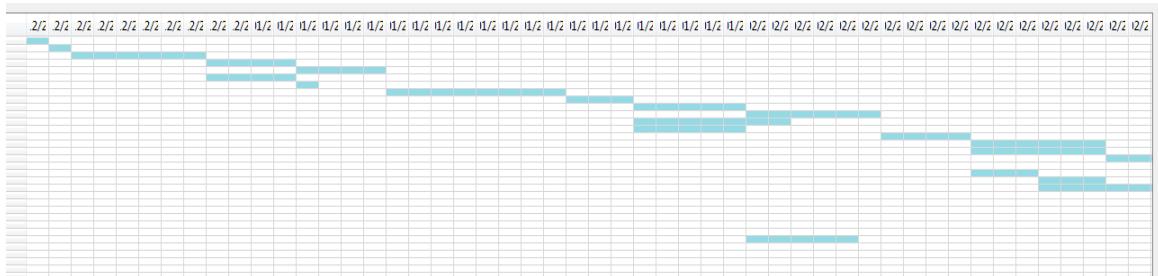
	lun	mar	mié	jue	vie	sáb	dom
18	29	30	1	2	3	4	5
19	6	7	8	9	10	11	12
20	13	14	15	16	17	18	19
21	20	21	22	23	24	25	26
22	27	28	29	30	31	1	2
23	3	4	5	6	7	8	9

ES

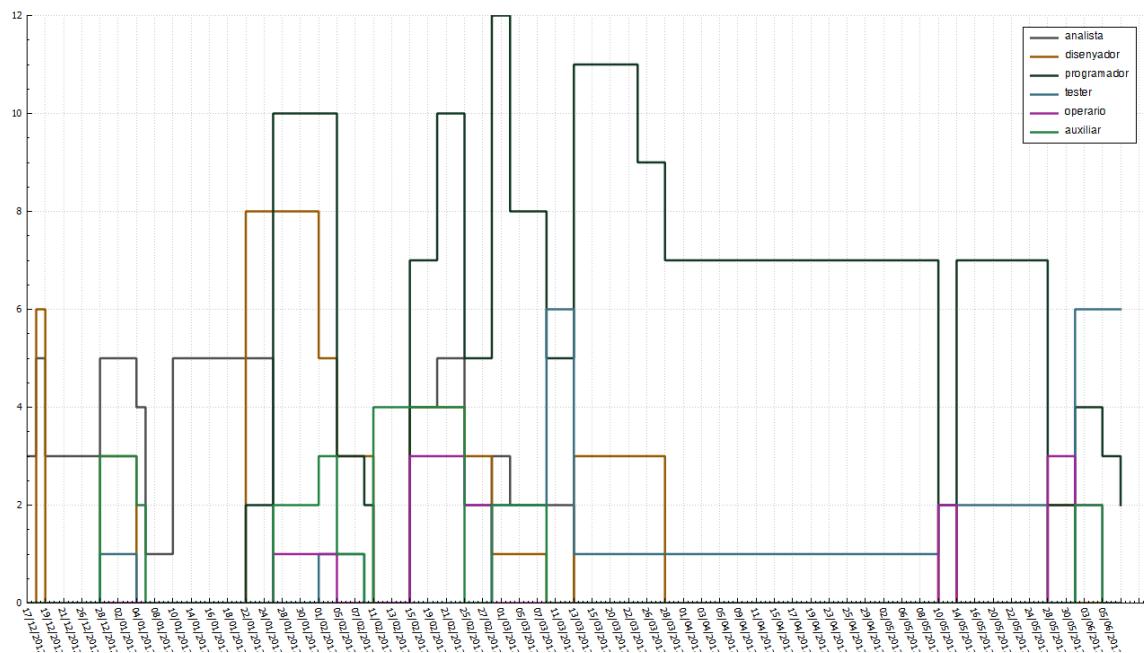
20:28

17/12/2012

5.2.4.2 Gant



5.2.4.3 Histograma recursos



5.2.4.4 Informe debug

```
#####
##### ADELANTO/ATRASO#####
||||||||| RETRASO||||||| 134 133 132 131 12 11 }

*****SERIE*****
ITERACION
elegir={522 521 51 47 46 41 42 43 44 37 34 362 361 322 352 321 31 351 26 23 45 24 25 22 21 136 135
136 135 134 133 132 131 12 11 }
La actividad 522 se secuencia en 0
disp0{ analista=5 disenador=8 programador=11 tester=0 operario=8 auxiliar=5}
disp1{ analista=5 disenador=8 programador=13 tester=6 operario=8 auxiliar=5}

ITERACION
elegir={521 51 47 46 41 42 43 44 37 34 362 361 322 352 321 31 351 26 23 45 24 25 22 21 136
135 134 133 132 131 12 11 }
La actividad 521 se secuencia en 1
disp0{ analista=5 disenador=8 programador=11 tester=0 operario=8 auxiliar=5}
disp1{ analista=5 disenador=8 programador=10 tester=0 operario=8 auxiliar=5}
disp3{ analista=5 disenador=8 programador=13 tester=6 operario=8 auxiliar=5}

ITERACION
elegir={51 47 46 41 42 43 44 37 34 362 361 322 352 321 31 351 26 23 45 24 25 22 21 136 135
134 133 132 131 12 11 }
La actividad 51 se secuencia en 3
disp0{ analista=5 disenador=8 programador=11 tester=0 operario=8 auxiliar=5}
disp1{ analista=5 disenador=8 programador=10 tester=0 operario=8 auxiliar=5}
disp3{ analista=5 disenador=8 programador=9 tester=0 operario=6 auxiliar=3}
disp6{ analista=5 disenador=8 programador=13 tester=6 operario=8 auxiliar=5}

ITERACION
elegir={47 46 41 42 43 44 37 34 362 361 322 352 321 31 351 26 23 45 24 25 22 21 136 135 134
133 132 131 12 11 }
La actividad 47 se secuencia en 6
disp0{ analista=5 disenador=8 programador=11 tester=0 operario=8 auxiliar=5}
```



```
disp113{ analista=5 disenador=8 programador=13 tester=6 operario=8 auxiliar=5}
```

INTERACION

```
elegir{131 12 11 }

La actividad 131 se secuencia en 113
disp0{ analista=4 disenador=6 programador=3 tester=0 operario=8 auxiliar=2}
disp1{ analista=4 disenador=6 programador=2 tester=0 operario=8 auxiliar=2}
disp2{ analista=3 disenador=6 programador=3 tester=0 operario=8 auxiliar=2}
disp3{ analista=3 disenador=6 programador=2 tester=0 operario=6 auxiliar=0}
disp6{ analista=3 disenador=6 programador=8 tester=6 operario=5 auxiliar=5}
disp8{ analista=5 disenador=6 programador=11 tester=6 operario=5 auxiliar=5}
disp9{ analista=5 disenador=8 programador=6 tester=4 operario=8 auxiliar=5}
disp19{ analista=3 disenador=6 programador=11 tester=4 operario=6 auxiliar=5}
disp21{ analista=5 disenador=5 programador=1 tester=4 operario=8 auxiliar=4}
disp26{ analista=5 disenador=5 programador=2 tester=5 operario=8 auxiliar=5}
disp28{ analista=5 disenador=5 programador=4 tester=5 operario=8 auxiliar=5}
disp31{ analista=5 disenador=8 programador=6 tester=5 operario=8 auxiliar=5}
disp61{ analista=3 disenador=8 programador=8 tester=0 operario=8 auxiliar=5}
disp64{ analista=3 disenador=7 programador=5 tester=4 operario=8 auxiliar=3}
disp70{ analista=4 disenador=5 programador=9 tester=6 operario=6 auxiliar=5}
disp73{ analista=1 disenador=6 programador=10 tester=6 operario=5 auxiliar=4}
disp76{ analista=3 disenador=6 programador=12 tester=6 operario=5 auxiliar=4}
disp79{ analista=5 disenador=8 programador=13 tester=6 operario=8 auxiliar=1}
disp83{ analista=5 disenador=1 programador=2 tester=6 operario=7 auxiliar=3}
disp88{ analista=5 disenador=3 programador=4 tester=6 operario=7 auxiliar=3}
disp89{ analista=5 disenador=2 programador=5 tester=6 operario=7 auxiliar=3}
disp90{ analista=5 disenador=4 programador=12 tester=6 operario=8 auxiliar=5}
disp94{ analista=0 disenador=0 programador=11 tester=6 operario=8 auxiliar=5}
disp97{ analista=0 disenador=8 programador=13 tester=6 operario=8 auxiliar=5}
disp105{ analista=1 disenador=8 programador=13 tester=6 operario=8 auxiliar=3}
disp106{ analista=1 disenador=5 programador=13 tester=6 operario=8 auxiliar=5}
disp109{ analista=0 disenador=5 programador=13 tester=5 operario=8 auxiliar=2}
disp110{ analista=3 disenador=8 programador=13 tester=5 operario=8 auxiliar=2}
disp113{ analista=2 disenador=8 programador=13 tester=6 operario=8 auxiliar=5}
disp119{ analista=5 disenador=8 programador=13 tester=6 operario=8 auxiliar=5}
```

INTERACION

```
elegir=12 11 }  
La actividad 12 se secuencia en 119  
disp0{ analista=4 disenayador=6 programador=3 tester=0 operario=8 auxiliar=2}  
disp1{ analista=4 disenayador=6 programador=2 tester=0 operario=8 auxiliar=2}  
disp2{ analista=3 disenayador=6 programador=3 tester=0 operario=8 auxiliar=2}  
disp3{ analista=3 disenayador=6 programador=2 tester=0 operario=6 auxiliar=0}  
disp6{ analista=3 disenayador=6 programador=8 tester=6 operario=5 auxiliar=5}  
disp8{ analista=5 disenayador=6 programador=11 tester=6 operario=5 auxiliar=5}  
disp9{ analista=5 disenayador=8 programador=6 tester=4 operario=8 auxiliar=5}  
disp19{ analista=3 disenayador=6 programador=11 tester=4 operario=6 auxiliar=5}  
disp21{ analista=5 disenayador=5 programador=1 tester=4 operario=8 auxiliar=4}  
disp26{ analista=5 disenayador=5 programador=2 tester=5 operario=8 auxiliar=5}  
disp28{ analista=5 disenayador=5 programador=4 tester=5 operario=8 auxiliar=5}  
disp31{ analista=5 disenayador=8 programador=6 tester=5 operario=8 auxiliar=5}  
disp61{ analista=3 disenayador=8 programador=8 tester=0 operario=8 auxiliar=5}  
disp64{ analista=3 disenayador=7 programador=5 tester=4 operario=8 auxiliar=3}  
disp70{ analista=4 disenayador=5 programador=9 tester=6 operario=6 auxiliar=5}  
disp73{ analista=1 disenayador=6 programador=10 tester=6 operario=5 auxiliar=4}  
disp76{ analista=3 disenayador=6 programador=12 tester=6 operario=5 auxiliar=4}  
disp79{ analista=5 disenayador=8 programador=13 tester=6 operario=8 auxiliar=1}  
disp83{ analista=5 disenayador=1 programador=2 tester=6 operario=7 auxiliar=3}  
disp88{ analista=5 disenayador=3 programador=4 tester=6 operario=7 auxiliar=3}  
disp89{ analista=5 disenayador=2 programador=5 tester=6 operario=7 auxiliar=3}  
disp90{ analista=5 disenayador=4 programador=12 tester=6 operario=8 auxiliar=5}  
disp94{ analista=0 disenayador=0 programador=11 tester=6 operario=8 auxiliar=5}  
disp97{ analista=0 disenayador=8 programador=13 tester=6 operario=8 auxiliar=5}  
disp105{ analista=1 disenayador=8 programador=13 tester=6 operario=8 auxiliar=3}  
disp106{ analista=1 disenayador=5 programador=13 tester=6 operario=8 auxiliar=5}  
disp109{ analista=0 disenayador=5 programador=13 tester=5 operario=8 auxiliar=2}  
disp110{ analista=3 disenayador=8 programador=13 tester=5 operario=8 auxiliar=2}  
disp113{ analista=2 disenayador=8 programador=13 tester=6 operario=8 auxiliar=5}  
disp119{ analista=0 disenayador=2 programador=13 tester=6 operario=8 auxiliar=5}  
disp120{ analista=5 disenayador=8 programador=13 tester=6 operario=8 auxiliar=5}
```

INTERACION

```
elegr={11 }
La actividad 11 se secuencia en 120
disp0{ analista=4 disenayador=6 programador=3 tester=0 operario=8 auxiliar=2}
disp1{ analista=4 disenayador=6 programador=2 tester=0 operario=8 auxiliar=2}
disp2{ analista=3 disenayador=6 programador=3 tester=0 operario=8 auxiliar=2}
disp3{ analista=3 disenayador=6 programador=2 tester=0 operario=6 auxiliar=0}
disp6{ analista=3 disenayador=6 programador=8 tester=6 operario=5 auxiliar=5}
disp8{ analista=5 disenayador=6 programador=11 tester=6 operario=5 auxiliar=5}
disp9{ analista=5 disenayador=8 programador=6 tester=4 operario=8 auxiliar=5}
disp19{ analista=3 disenayador=6 programador=11 tester=4 operario=6 auxiliar=5}
```

```

disp21{ analista=5 disenayador=5 programador=1 tester=4 operario=8 auxiliar=4}
disp26{ analista=5 disenayador=5 programador=2 tester=5 operario=8 auxiliar=5}
disp28{ analista=5 disenayador=5 programador=4 tester=5 operario=8 auxiliar=5}
disp31{ analista=5 disenayador=8 programador=6 tester=5 operario=8 auxiliar=5}
disp61{ analista=3 disenayador=8 programador=8 tester=0 operario=8 auxiliar=5}
disp64{ analista=3 disenayador=7 programador=5 tester=4 operario=8 auxiliar=3}
disp70{ analista=4 disenayador=5 programador=9 tester=6 operario=6 auxiliar=5}
disp73{ analista=1 disenayador=6 programador=10 tester=6 operario=5 auxiliar=4}
disp76{ analista=3 disenayador=6 programador=12 tester=6 operario=5 auxiliar=4}
disp79{ analista=5 disenayador=8 programador=13 tester=6 operario=8 auxiliar=1}
disp83{ analista=5 disenayador=1 programador=2 tester=6 operario=7 auxiliar=3}
disp88{ analista=5 disenayador=3 programador=4 tester=6 operario=7 auxiliar=3}
disp89{ analista=5 disenayador=2 programador=5 tester=6 operario=7 auxiliar=3}
disp90{ analista=5 disenayador=4 programador=12 tester=6 operario=8 auxiliar=5}
disp94{ analista=0 disenayador=0 programador=11 tester=6 operario=8 auxiliar=5}
disp97{ analista=0 disenayador=8 programador=13 tester=6 operario=8 auxiliar=5}
disp105{ analista=1 disenayador=8 programador=13 tester=6 operario=8 auxiliar=3}
disp106{ analista=1 disenayador=5 programador=13 tester=6 operario=8 auxiliar=5}
disp109{ analista=0 disenayador=5 programador=13 tester=5 operario=8 auxiliar=2}
disp110{ analista=3 disenayador=8 programador=13 tester=5 operario=8 auxiliar=2}
disp113{ analista=2 disenayador=8 programador=13 tester=6 operario=8 auxiliar=5}
disp119{ analista=0 disenayador=2 programador=13 tester=6 operario=8 auxiliar=5}
disp120{ analista=2 disenayador=8 programador=13 tester=6 operario=8 auxiliar=5}
disp121{ analista=5 disenayador=8 programador=13 tester=6 operario=8 auxiliar=5}

||||||||||||||||| ADELANTO|||||||||||||||||
elegir={11 12 131 132 134 133 135 136 21 22 24 23 25 26 321 351 322 352 34 37 42 43 44 45 41 46 47
361 31 51 521 362 522 }

*****PARALELO*****
```

ITERACION T=0

```

disp0{ analista=5 disenayador=8 programador=13 tester=6 operario=8 auxiliar=5}
elegir={11 12 131 132 134 133 135 136 21 22 24 23 25 26 321 351 322 352 34 37 42 43 44 45 41
46 47 361 31 51 521 362 522 }
Se secuencia 11 en 0 y termina en 1
disp0{ analista=2 disenayador=8 programador=13 tester=6 operario=8 auxiliar=5}
```

ITERACION T=0

```

La actividad 11 termina de procesarse
disp1{ analista=5 disenayador=8 programador=13 tester=6 operario=8 auxiliar=5}
elegir={12 131 132 134 133 135 136 21 22 24 23 25 26 321 351 322 352 34 37 42 43 44 45 41 46
47 361 31 51 521 362 522 }
Se secuencia 12 en 1 y termina en 2
disp1{ analista=0 disenayador=2 programador=13 tester=6 operario=8 auxiliar=5}
```

ITERACION T=1

```

La actividad 12 termina de procesarse
disp2{ analista=5 disenayador=8 programador=13 tester=6 operario=8 auxiliar=5}
elegir={131 132 134 133 135 136 21 22 24 23 25 26 321 351 322 352 34 37 42 43 44 45 41 46 47
361 31 51 521 362 522 }
Se secuencia 131 en 2 y termina en 8
disp2{ analista=2 disenayador=8 programador=13 tester=6 operario=8 auxiliar=5}
```

ITERACION T=2

```

La actividad 131 termina de procesarse
disp8{ analista=5 disenayador=8 programador=13 tester=6 operario=8 auxiliar=5}
elegir={132 134 133 135 136 21 22 24 23 25 26 321 351 322 352 34 37 42 43 44 45 41 46 47 361
31 51 521 362 522 }
Se secuencia 132 en 8 y termina en 12
Se secuencia 134 en 8 y termina en 12
disp8{ analista=0 disenayador=5 programador=13 tester=5 operario=8 auxiliar=2}
```

ITERACION T=8

```

La actividad 132 termina de procesarse
La actividad 134 termina de procesarse
disp12{ analista=5 disenayador=8 programador=13 tester=6 operario=8 auxiliar=5}
elegir={133 135 136 21 22 24 23 25 26 321 351 322 352 34 37 42 43 44 45 41 46 47 361 31 51
521 362 522 }
Se secuencia 133 en 12 y termina en 16
Se secuencia 135 en 12 y termina en 13
disp12{ analista=1 disenayador=8 programador=13 tester=6 operario=8 auxiliar=3}
```

ITERACION T=12

```

La actividad 135 termina de procesarse
disp13{ analista=4 disenayador=8 programador=13 tester=6 operario=8 auxiliar=5}
elegir={136 21 22 24 23 25 26 321 351 322 352 34 37 42 43 44 45 41 46 47 361 31 51 521 362
522 }
disp13{ analista=4 disenayador=8 programador=13 tester=6 operario=8 auxiliar=5}
```

```

ITERACION __T=13
La actividad 133 termina de procesarse
disp16{ analista=5 disenayador=8 programador=13 tester=6 operario=8 auxiliar=5}
elegir={136 21 22 24 23 25 26 321 351 322 352 34 37 42 43 44 45 41 46 47 361 31 51 521 362
522 }
Se secuencia 136 en 16 y termina en 24
disp16{ analista=0 disenayador=8 programador=13 tester=6 operario=8 auxiliar=5}

ITERACION __T=16
La actividad 136 termina de procesarse
disp24{ analista=5 disenayador=8 programador=13 tester=6 operario=8 auxiliar=5}
elegir={21 22 24 23 25 26 321 351 322 352 34 37 42 43 44 45 41 46 47 361 31 51 521 362 522 }
Se secuencia 21 en 24 y termina en 27
disp24{ analista=0 disenayador=0 programador=11 tester=6 operario=8 auxiliar=5}

ITERACION __T=24
La actividad 21 termina de procesarse
disp27{ analista=5 disenayador=8 programador=13 tester=6 operario=8 auxiliar=5}
elegir={22 24 23 25 26 321 351 322 352 34 37 42 43 44 45 41 46 47 361 31 51 521 362 522 }
Se secuencia 22 en 27 y termina en 32
Se secuencia 24 en 27 y termina en 34
Se secuencia 25 en 27 y termina en 32
disp27{ analista=5 disenayador=0 programador=3 tester=6 operario=7 auxiliar=3}

ITERACION __T=27
La actividad 22 termina de procesarse
La actividad 25 termina de procesarse
disp32{ analista=5 disenayador=6 programador=6 tester=6 operario=7 auxiliar=3}
elegir={23 26 321 351 322 352 34 37 42 43 44 45 41 46 47 361 31 51 521 362 522 }
Se secuencia 23 en 32 y termina en 38
Se secuencia 45 en 32 y termina en 37
disp32{ analista=5 disenayador=3 programador=3 tester=5 operario=7 auxiliar=2}

ITERACION __T=32
La actividad 24 termina de procesarse
disp34{ analista=5 disenayador=5 programador=10 tester=5 operario=8 auxiliar=4}
elegir={26 321 351 322 352 34 37 42 43 44 41 46 47 361 31 51 521 362 522 }
disp34{ analista=5 disenayador=5 programador=10 tester=5 operario=8 auxiliar=4}

ITERACION __T=34
La actividad 45 termina de procesarse
disp37{ analista=5 disenayador=5 programador=11 tester=6 operario=8 auxiliar=5}
elegir={26 321 351 322 352 34 37 42 43 44 41 46 47 361 31 51 521 362 522 }
disp37{ analista=5 disenayador=5 programador=11 tester=6 operario=8 auxiliar=5}

ITERACION __T=37
La actividad 23 termina de procesarse
disp38{ analista=5 disenayador=8 programador=13 tester=6 operario=8 auxiliar=5}
elegir={26 321 351 322 352 34 37 42 43 44 41 46 47 361 31 51 521 362 522 }
Se secuencia 26 en 38 y termina en 42
disp38{ analista=5 disenayador=8 programador=13 tester=6 operario=8 auxiliar=1}

ITERACION __T=38
La actividad 26 termina de procesarse
disp42{ analista=5 disenayador=8 programador=13 tester=6 operario=8 auxiliar=5}
elegir={321 351 322 352 34 37 42 43 44 41 46 47 361 31 51 521 362 522 }
Se secuencia 321 en 42 y termina en 48
Se secuencia 351 en 42 y termina en 45
Se secuencia 31 en 42 y termina en 48
disp42{ analista=1 disenayador=4 programador=6 tester=6 operario=5 auxiliar=1}

ITERACION __T=42
La actividad 351 termina de procesarse
disp45{ analista=3 disenayador=4 programador=8 tester=6 operario=5 auxiliar=1}
elegir={322 352 34 37 42 43 44 41 46 47 361 51 521 362 522 }
Se secuencia 352 en 45 y termina en 48
Se secuencia 361 en 45 y termina en 51
disp45{ analista=0 disenayador=4 programador=3 tester=6 operario=5 auxiliar=1}

ITERACION __T=45
La actividad 321 termina de procesarse
La actividad 31 termina de procesarse
La actividad 352 termina de procesarse
disp48{ analista=3 disenayador=8 programador=10 tester=6 operario=8 auxiliar=5}
elegir={322 34 37 42 43 44 41 46 47 51 521 362 522 }
Se secuencia 322 en 48 y termina en 51
disp48{ analista=3 disenayador=5 programador=8 tester=6 operario=6 auxiliar=5}

```

```

ITERACION __ T=48
La actividad 361 termina de procesarse
La actividad 322 termina de procesarse
disp51{ analista=5 disenador=8 programador=13 tester=6 operario=8 auxiliar=5}
elegir=(34 37 42 43 44 41 46 47 51 521 362 522 )
Se secuencia 34 en 51 y termina en 57
Se secuencia 362 en 51 y termina en 53
disp51{ analista=2 disenador=7 programador=1 tester=4 operario=8 auxiliar=3}

ITERACION __ T=51
La actividad 362 termina de procesarse
disp53{ analista=3 disenador=7 programador=5 tester=4 operario=8 auxiliar=3}
elegir=(37 42 43 44 41 46 47 51 521 522 )
disp53{ analista=3 disenador=7 programador=5 tester=4 operario=8 auxiliar=3}

ITERACION __ T=53
La actividad 34 termina de procesarse
disp57{ analista=5 disenador=8 programador=13 tester=6 operario=8 auxiliar=5}
elegir=(37 42 43 44 41 46 47 51 521 522 )
Se secuencia 37 en 57 y termina en 60
disp57{ analista=3 disenador=8 programador=8 tester=0 operario=8 auxiliar=5}

ITERACION __ T=57
La actividad 37 termina de procesarse
disp60{ analista=5 disenador=8 programador=13 tester=6 operario=8 auxiliar=5}
elegir=(42 43 44 41 46 47 51 521 522 )
Se secuencia 42 en 60 y termina en 100
Se secuencia 43 en 60 y termina en 70
Se secuencia 44 en 60 y termina en 67
disp60{ analista=5 disenador=5 programador=2 tester=5 operario=8 auxiliar=5}

ITERACION __ T=60
La actividad 44 termina de procesarse
disp67{ analista=5 disenador=5 programador=4 tester=5 operario=8 auxiliar=5}
elegir=(41 46 47 51 521 522 )
disp67{ analista=5 disenador=5 programador=4 tester=5 operario=8 auxiliar=5}

ITERACION __ T=67
La actividad 43 termina de procesarse
disp70{ analista=5 disenador=8 programador=6 tester=5 operario=8 auxiliar=5}
elegir=(41 46 47 51 521 522 )
disp70{ analista=5 disenador=8 programador=6 tester=5 operario=8 auxiliar=5}

ITERACION __ T=70
La actividad 42 termina de procesarse
disp100{ analista=5 disenador=8 programador=13 tester=6 operario=8 auxiliar=5}
elegir=(41 46 47 51 521 522 )
Se secuencia 41 en 100 y termina en 102
disp100{ analista=3 disenador=6 programador=11 tester=4 operario=6 auxiliar=5}

ITERACION __ T=100
La actividad 41 termina de procesarse
disp102{ analista=5 disenador=8 programador=13 tester=6 operario=8 auxiliar=5}
elegir=(46 47 51 521 522 )
Se secuencia 46 en 102 y termina en 112
disp102{ analista=5 disenador=8 programador=6 tester=4 operario=8 auxiliar=5}

ITERACION __ T=102
La actividad 46 termina de procesarse
disp112{ analista=5 disenador=8 programador=13 tester=6 operario=8 auxiliar=5}
elegir=(47 51 521 522 )
Se secuencia 47 en 112 y termina en 115
disp112{ analista=5 disenador=6 programador=11 tester=6 operario=5 auxiliar=5}

ITERACION __ T=112
La actividad 47 termina de procesarse
disp115{ analista=5 disenador=8 programador=13 tester=6 operario=8 auxiliar=5}
elegir=(51 521 522 )
Se secuencia 51 en 115 y termina en 118
disp115{ analista=5 disenador=8 programador=9 tester=0 operario=6 auxiliar=3}

ITERACION __ T=115
La actividad 51 termina de procesarse
disp118{ analista=5 disenador=8 programador=13 tester=6 operario=8 auxiliar=5}
elegir=(521 522 )
Se secuencia 521 en 118 y termina en 120
disp118{ analista=5 disenador=8 programador=10 tester=0 operario=8 auxiliar=5}

```

```
ITERACION __T=118
La actividad 521 termina de procesarse
disp120{ analista=5 disenayador=8 programador=13 tester=6 operario=8 auxiliar=5}
elegir={522 }
Se secuencia 522 en 120 y termina en 121
disp120{ analista=5 disenayador=8 programador=11 tester=0 operario=8 auxiliar=5}
```

```
ITERACION __T=120
La actividad 522 termina de procesarse
disp121{ analista=5 disenayador=8 programador=13 tester=6 operario=8 auxiliar=5}
elegir={}
disp121{ analista=5 disenayador=8 programador=13 tester=6 operario=8 auxiliar=5}
```

5.2.5 Nivelación de todos los recursos

5.2.5.1 Informe

Informe Nivelación de Recursos

Informe Actividades

Camino Crítico: 11-12-131-132-136-21-22-23-26-321-322-34-37-41-46-47-51-521-522
 11-12-131-132-136-21-22-23-26-321-322-34-37-41-46-47-51-521-522
 11-12-131-133-136-21-22-23-26-321-322-34-37-41-46-47-51-521-522
 11-12-131-133-136-21-22-23-26-321-322-34-37-41-46-47-51-521-522
 11-12-131-134-136-21-22-23-26-321-322-34-37-41-46-47-51-521-522
 11-12-131-134-136-21-22-23-26-321-322-34-37-42-41-46-47-51-521-522

Fecha Inicio: 17/12/2012 Fecha fin: 07/06/2013

Duración: 121 días laborables

Sobrecosto: 35 euros

Nombre	F. Inicio Temprano	F. Inicio Tardío	F. Fin Temprano	F. Fin Tardío	HT	HL	Sucesoras
1 11	17/12/2012	21/12/2012	17/12/2012	21/12/2012	4	0	12-135
2 12	18/12/2012	24/12/2012	18/12/2012	24/12/2012	4	0	131
3 131	19/12/2012	26/12/2012	27/12/2012	03/01/2013	4	0	132-133-134-13...
4 132	28/12/2012	04/01/2013	03/01/2013	09/01/2013	4	4	136
5 133	04/01/2013	04/01/2013	09/01/2013	09/01/2013	0	0	136
6 134	03/01/2013	04/01/2013	08/01/2013	09/01/2013	1	1	136

Calendario Proyecto

diciembre 2012

	lun	mar	mié	jue	vie	sáb	dom
48	26	27	28	29	30	1	2
49	3	4	5	6	7	8	9
50	10	11	12	13	14	15	16
51	17	18	19	20	21	22	23
52	24	25	26	27	28	29	30
1	31	1	2	3	4	5	6

17/12/2012

Informe Nivelación de Recursos

Informe Actividades

Camino Crítico: 11-12-131-132-136-21-22-23-26-321-322-34-37-41-46-47-51-521-522
 11-12-131-132-136-21-22-23-26-321-322-34-37-41-46-47-51-521-522
 11-12-131-133-136-21-22-23-26-321-322-34-37-41-46-47-51-521-522
 11-12-131-133-136-21-22-23-26-321-322-34-37-41-46-47-51-521-522
 11-12-131-134-136-21-22-23-26-321-322-34-37-41-46-47-51-521-522

Fecha Inicio: 17/12/2012 Fecha fin: 07/06/2013

Duración: 121 días laborables

Sobrecosto: 35 euros

Nombre	F. Inicio Temprano	F. Inicio Tardío	F. Fin Temprano	F. Fin Tardío	HT	HL	Sucesoras
7 135	09/01/2013	09/01/2013	09/01/2013	09/01/2013	0	0	136
8 136	10/01/2013	10/01/2013	21/01/2013	21/01/2013	0	0	21-24
9 21	22/01/2013	22/01/2013	24/01/2013	24/01/2013	0	0	22-24-25
10 22	25/01/2013	25/01/2013	31/01/2013	31/01/2013	0	0	23
11 23	01/02/2013	01/02/2013	08/02/2013	08/02/2013	0	0	26
12 24	25/01/2013	31/01/2013	04/02/2013	08/02/2013	4	4	26-361

Calendario Proyecto

enero 2013

	lun	mar	mié	jue	vie	sáb	dom
1	31	1	2	3	4	5	6
2	7	8	9	10	11	12	13
3	14	15	16	17	18	19	20
4	21	22	23	24	25	26	27
5	28	29	30	31	1	2	3
6	4	5	6	7	8	9	10

17/12/2012

Informe Nivelación de Recursos

Informe Actividades

Camino Crítico: 11-12-131-132-136-21-22-23-26-321-322-34-37-41-46-47-51-521-522
 11-12-131-132-136-21-22-23-26-321-322-34-37-41-46-47-51-521-522
 11-12-131-133-136-21-22-23-26-321-322-34-37-41-46-47-51-521-522
 11-12-131-133-136-21-22-23-26-321-322-34-37-41-46-47-51-521-522
 11-12-131-134-136-21-22-23-26-321-322-34-37-41-46-47-51-521-522
 Fecha Inicio: 17/12/2012 Fecha Fin: 07/06/2013

Duración: 121 días laborables

Sobrecoste: 35 euros

Nombre	F. Inicio Temprano	F. Inicio Tardío	F. Fin Temprano	F. Fin Tardío	HT	HL	Sucesoras
13_25	25/01/2013	04/02/2013	31/01/2013	08/02/2013	6	1	26-321-45-361
14_26	11/02/2013	11/02/2013	14/02/2013	14/02/2013	0	0	31-321-351-361
15_31	08/03/2013	31/05/2013	15/03/2013	07/06/2013	58	58	
16_321	15/02/2013	15/02/2013	22/02/2013	22/02/2013	0	0	322
17_322	25/02/2013	25/02/2013	27/02/2013	27/02/2013	0	0	34
18_34	28/02/2013	28/02/2013	07/03/2013	07/03/2013	0	0	37

Calendario Proyecto

febrero, 2013

	lun	mar	mié	jue	vie	sáb	dom
5	28	29	30	31	1	2	3
6	4	5	6	7	8	9	10
7	11	12	13	14	15	16	17
8	18	19	20	21	22	23	24
9	25	26	27	28	1	2	3
10	4	5	6	7	8	9	10

ES 20:32 17/12/2012

Informe Nivelación de Recursos

Informe Actividades

Camino Crítico: 11-12-131-132-136-21-22-23-26-321-322-34-37-41-46-47-51-521-522
 11-12-131-132-136-21-22-23-26-321-322-34-37-41-46-47-51-521-522
 11-12-131-133-136-21-22-23-26-321-322-34-37-41-46-47-51-521-522
 11-12-131-133-136-21-22-23-26-321-322-34-37-41-46-47-51-521-522
 11-12-131-134-136-21-22-23-26-321-322-34-37-41-46-47-51-521-522
 Fecha Inicio: 17/12/2012 Fecha Fin: 07/06/2013

Duración: 121 días laborables

Sobrecoste: 35 euros

Nombre	F. Inicio Temprano	F. Inicio Tardío	F. Fin Temprano	F. Fin Tardío	HT	HL	Sucesoras
19_351	15/02/2013	20/02/2013	19/02/2013	22/02/2013	3	3	352
20_352	25/02/2013	25/02/2013	27/02/2013	27/02/2013	0	0	34
21_361	20/02/2013	29/05/2013	27/02/2013	05/06/2013	68	62	362
22_362	29/05/2013	06/06/2013	30/05/2013	07/06/2013	6	6	
23_37	08/03/2013	08/03/2013	12/03/2013	12/03/2013	0	0	41-42-43-44
24_41	10/05/2013	10/05/2013	13/05/2013	13/05/2013	0	0	46

Calendario Proyecto

marzo, 2013

	lun	mar	mié	jue	vie	sáb	dom
9	25	26	27	28	1	2	3
10	4	5	6	7	8	9	10
11	11	12	13	14	15	16	17
12	18	19	20	21	22	23	24
13	25	26	27	28	29	30	31
14	1	2	3	4	5	6	7

ES 20:32 17/12/2012

Informe Nivelación de Recursos

Informe Actividades

Camino Crítico: 11-12-131-132-136-21-22-23-26-321-322-34-37-41-46-47-51-521-522
 11-12-131-132-136-21-22-23-26-321-322-34-37-42-46-47-51-521-522
 11-12-131-133-136-21-22-23-26-321-322-34-37-41-46-47-51-521-522
 11-12-131-133-136-21-22-23-26-321-322-34-37-40-41-46-47-51-521-522
 11-12-131-134-136-21-22-23-26-321-322-34-37-40-46-47-51-521-522
 11-12-131-134-136-21-22-23-26-321-322-34-37-42-46-47-51-521-522

Fecha Inicio: 17/12/2012 Fecha Fin: 07/06/2013

Duración: 121 días laborables

Sobrecoste: 35 euros

Nombre	F. Inicio Temprano	F. Inicio Tardío	F. Fin Temprano	F. Fin Tardío	HT	HL	Sucesoras
25_42	13/03/2013	13/03/2013	09/05/2013	09/05/2013	0	0	41
26_43	25/04/2013	25/04/2013	09/05/2013	09/05/2013	0	0	41
27_44	16/04/2013	30/04/2013	24/04/2013	09/05/2013	10	10	41
28_45	04/02/2013	03/05/2013	08/02/2013	09/05/2013	62	62	41
29_46	14/05/2013	14/05/2013	27/05/2013	27/05/2013	0	0	47
30_47	28/05/2013	28/05/2013	30/05/2013	30/05/2013	0	0	51

Calendario Proyecto

abril 2013

	lun	mar	mié	jue	vie	sáb	dom
13	25	26	27	28	29	30	31
14	1	2	3	4	5	6	7
15	8	9	10	11	12	13	14
16	15	16	17	18	19	20	21
17	22	23	24	25	26	27	28
18	29	30	1	2	3	4	5

17/12/2012 20:32

Informe Nivelación de Recursos

Informe Actividades

Camino Crítico: 11-12-131-132-136-21-22-23-26-321-322-34-37-41-46-47-51-521-522
 11-12-131-132-136-21-22-23-26-321-322-34-37-42-46-47-51-521-522
 11-12-131-133-136-21-22-23-26-321-322-34-37-42-46-47-51-521-522
 11-12-131-133-136-21-22-23-26-321-322-34-37-41-46-47-51-521-522
 11-12-131-134-136-21-22-23-26-321-322-34-37-41-46-47-51-521-522

Fecha Inicio: 17/12/2012 Fecha Fin: 07/06/2013

Duración: 121 días laborables

Sobrecoste: 35 euros

Nombre	F. Inicio Temprano	F. Inicio Tardío	F. Fin Temprano	F. Fin Tardío	HT	HL	Sucesoras
29_46	14/05/2013	14/05/2013	27/05/2013	27/05/2013	0	0	47
30_47	28/05/2013	28/05/2013	30/05/2013	30/05/2013	0	0	51
31_51	31/05/2013	31/05/2013	04/06/2013	04/06/2013	0	0	521
32_521	05/06/2013	05/06/2013	06/06/2013	06/06/2013	0	0	522
33_522	07/06/2013	07/06/2013	07/06/2013	07/06/2013	0	0	
34							

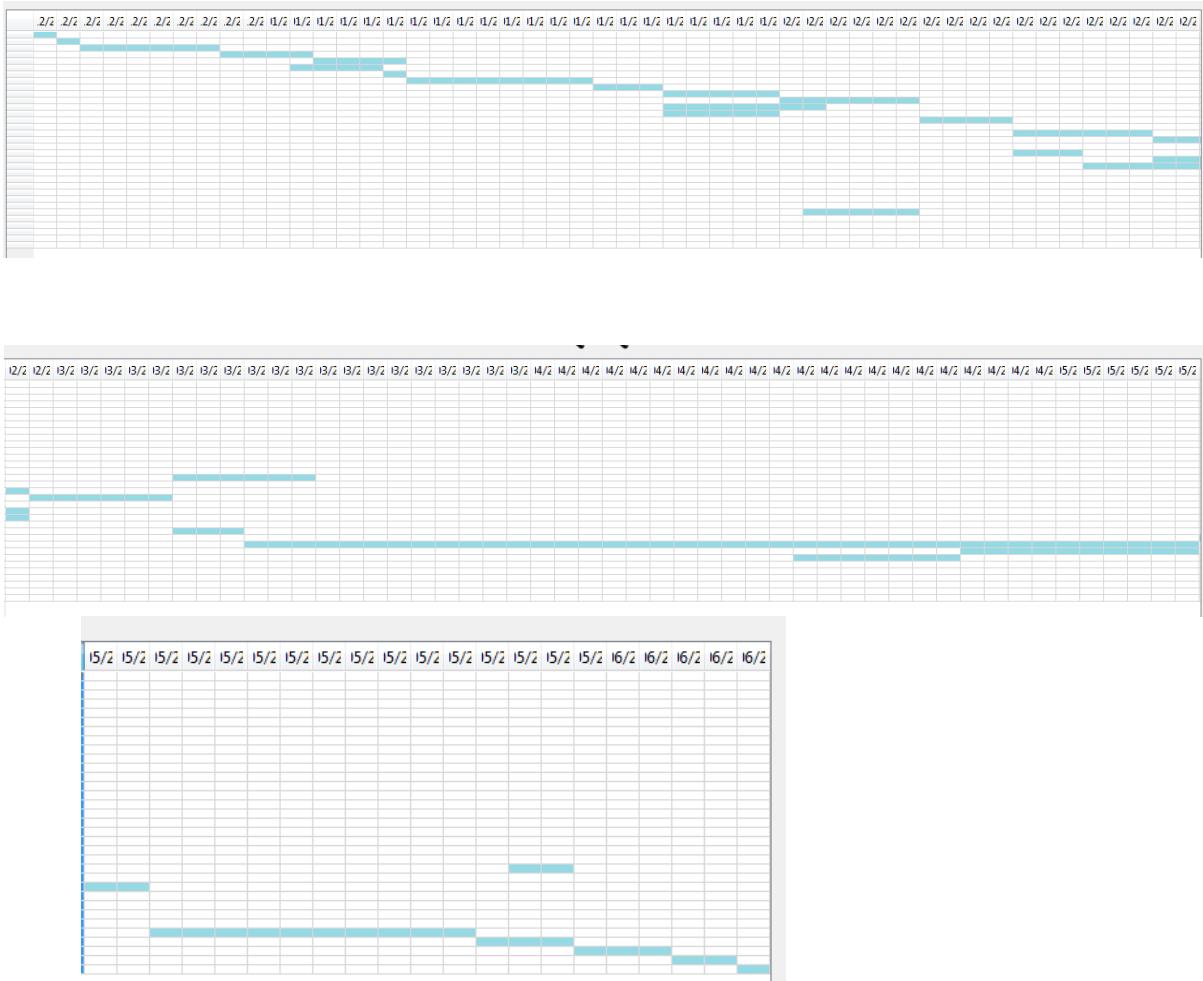
Calendario Proyecto

mayo 2013

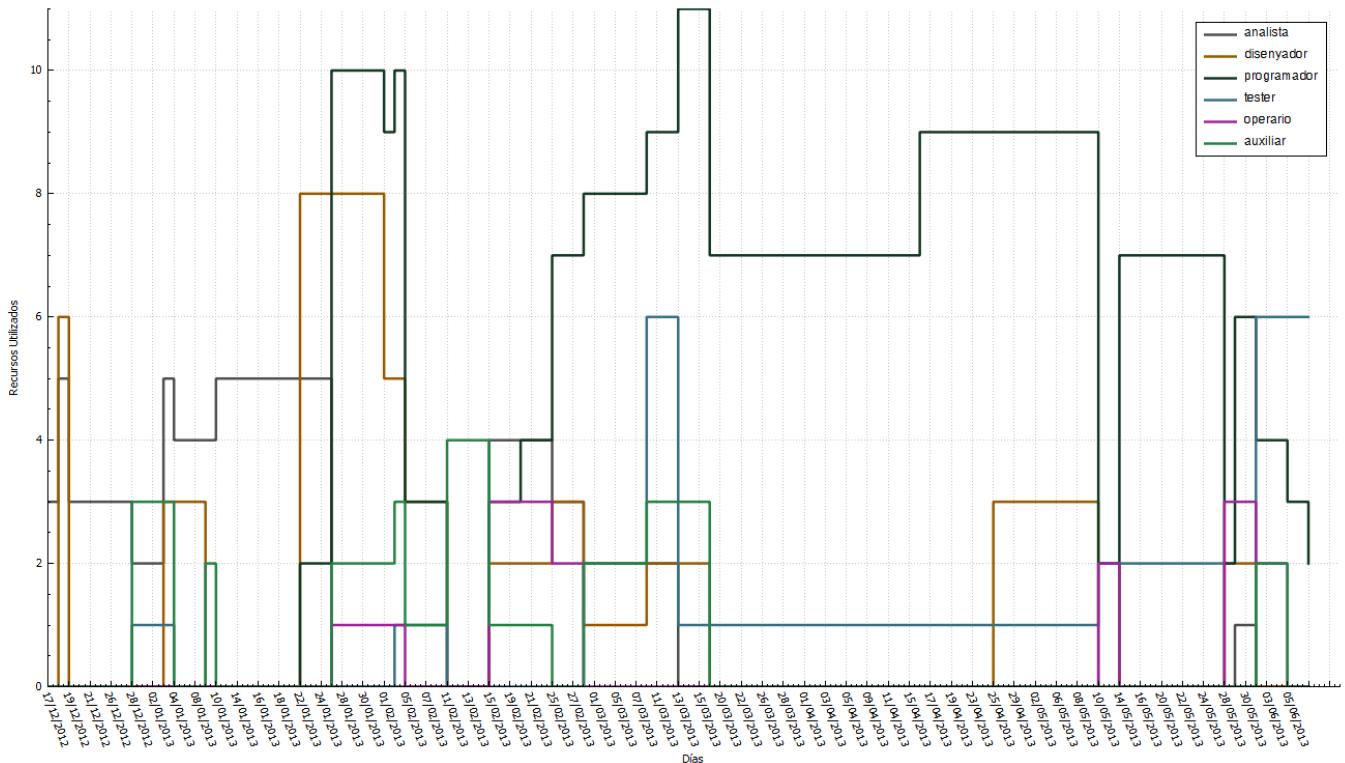
	lun	mar	mié	jue	vie	sáb	dom
18	29	30	1	2	3	4	5
19	6	7	8	9	10	11	12
20	13	14	15	16	17	18	19
21	20	21	22	23	24	25	26
22	27	28	29	30	31	1	2
23	3	4	5	6	7	8	9

17/12/2012 20:33

5.2.5.2 Gantt



5.2.5.3 Histograma de recursos



5.2.5.4 Informe debug

```
***** NIVELACION RECURSOS *****

Orden de las actividades a mover: 522 521 51 47 46 41 42 43 44 37 34 362 361 322 352 321 31 351 26 23
45 24 25 22 21 136 133 135 134 132 131 12 11

CVoriginal = 8.89693

522 TMin Original=121 H.Libre=0:
 522 se realiza en 120
-----
521 TMin Original=119 H.Libre=0:
 521 se realiza en 118
-----
51 TMin Original=116 H.Libre=0:
 51 se realiza en 115
-----
47 TMin Original=113 H.Libre=0:
 47 se realiza en 112
-----
46 TMin Original=103 H.Libre=0:
 46 se realiza en 102
-----
41 TMin Original=101 H.Libre=0:
 41 se realiza en 100
-----
42 TMin Original=61 H.Libre=0:
 42 se realiza en 60
-----
43 TMin Original=61 H.Libre=30:

  Movemos la actividad 43 al instante 61 con un CV=8.89524
  La nueva disposicion mejora la anterior
  Movemos la actividad 43 al instante 62 con un CV=8.89354
  La nueva disposicion mejora la anterior
```

44 TMin Original=61 H.Libre=33:

Movemos la actividad 44 al instante 61 con un CV=8.88498
La nueva disposicion mejora la anterior
Movemos la actividad 44 al instante 62 con un CV=8.88498
La nueva disposicion mejora la anterior
Movemos la actividad 44 al instante 63 con un CV=8.88498
La nueva disposicion mejora la anterior
Movemos la actividad 44 al instante 64 con un CV=8.88498
La nueva disposicion mejora la anterior
Movemos la actividad 44 al instante 65 con un CV=8.88498
La nueva disposicion mejora la anterior
Movemos la actividad 44 al instante 66 con un CV=8.88498
La nueva disposicion mejora la anterior
Movemos la actividad 44 al instante 67 con un CV=8.88498
La nueva disposicion mejora la anterior
Movemos la actividad 44 al instante 68 con un CV=8.88498
La nueva disposicion mejora la anterior
Movemos la actividad 44 al instante 69 con un CV=8.88498
La nueva disposicion mejora la anterior
Movemos la actividad 44 al instante 70 con un CV=8.88498
La nueva disposicion mejora la anterior
Movemos la actividad 44 al instante 71 con un CV=8.88498
La nueva disposicion mejora la anterior
Movemos la actividad 44 al instante 72 con un CV=8.88498
La nueva disposicion mejora la anterior
Movemos la actividad 44 al instante 73 con un CV=8.88498
La nueva disposicion mejora la anterior
Movemos la actividad 44 al instante 74 con un CV=8.88498
La nueva disposicion mejora la anterior

```

Movemos la actividad 44 al instante 75 con un CV=8.88498
La nueva disposicion mejora la anterior
Movemos la actividad 44 al instante 76 con un CV=8.88498
La nueva disposicion mejora la anterior
Movemos la actividad 44 al instante 77 con un CV=8.88498
La nueva disposicion mejora la anterior
Movemos la actividad 44 al instante 78 con un CV=8.88498
La nueva disposicion mejora la anterior
Movemos la actividad 44 al instante 79 con un CV=8.88498
La nueva disposicion mejora la anterior
Movemos la actividad 44 al instante 80 con un CV=8.88498
La nueva disposicion mejora la anterior
Movemos la actividad 44 al instante 81 con un CV=8.88498
La nueva disposicion mejora la anterior
Movemos la actividad 44 al instante 82 con un CV=8.88498
La nueva disposicion mejora la anterior
Movemos la actividad 44 al instante 83 con un CV=8.88498
La nueva disposicion mejora la anterior
Movemos la actividad 44 al instante 84 con un CV=8.8867
Movemos la actividad 44 al instante 85 con un CV=8.88842
Movemos la actividad 44 al instante 86 con un CV=8.89013
Movemos la actividad 44 al instante 87 con un CV=8.89183
Movemos la actividad 44 al instante 88 con un CV=8.89354
Movemos la actividad 44 al instante 89 con un CV=8.89524
Movemos la actividad 44 al instante 90 con un CV=8.89693
Movemos la actividad 44 al instante 91 con un CV=8.89693
Movemos la actividad 44 al instante 92 con un CV=8.89693
Movemos la actividad 44 al instante 93 con un CV=8.89693
44 se realiza en 83
-----
37  TMin Original=58  H.Libre=0:
    37 se realiza en 57
-----
34  TMin Original=52  H.Libre=0:
    34 se realiza en 51
-----
362 TMin Original=52  H.Libre=68:
    Movemos la actividad 362 al instante 52 con un CV=8.88498
    La nueva disposicion mejora la anterior
    Movemos la actividad 362 al instante 53 con un CV=8.88498
    La nueva disposicion mejora la anterior
    Movemos la actividad 362 al instante 54 con un CV=8.88498
    La nueva disposicion mejora la anterior
    Movemos la actividad 362 al instante 55 con un CV=8.88498
    La nueva disposicion mejora la anterior
    Movemos la actividad 362 al instante 56 con un CV=8.8798
    La nueva disposicion mejora la anterior
    Movemos la actividad 362 al instante 57 con un CV=8.87457
    La nueva disposicion mejora la anterior
    Movemos la actividad 362 al instante 58 con un CV=8.87457
    La nueva disposicion mejora la anterior
    Movemos la actividad 362 al instante 59 con un CV=8.87173
    La nueva disposicion mejora la anterior
    Movemos la actividad 362 al instante 60 con un CV=8.86884
    La nueva disposicion mejora la anterior
    Movemos la actividad 362 al instante 61 con un CV=8.86884
    La nueva disposicion mejora la anterior
    Movemos la actividad 362 al instante 62 con un CV=8.86884
    La nueva disposicion mejora la anterior
    Movemos la actividad 362 al instante 63 con un CV=8.86884
    La nueva disposicion mejora la anterior
    Movemos la actividad 362 al instante 64 con un CV=8.86884
    La nueva disposicion mejora la anterior
    Movemos la actividad 362 al instante 65 con un CV=8.86884
    La nueva disposicion mejora la anterior
    Movemos la actividad 362 al instante 66 con un CV=8.86884
    La nueva disposicion mejora la anterior
    Movemos la actividad 362 al instante 67 con un CV=8.86884
    La nueva disposicion mejora la anterior
    Movemos la actividad 362 al instante 68 con un CV=8.86884
    La nueva disposicion mejora la anterior
    Movemos la actividad 362 al instante 69 con un CV=8.86884
    La nueva disposicion mejora la anterior
    Movemos la actividad 362 al instante 70 con un CV=8.86884
    La nueva disposicion mejora la anterior
    Movemos la actividad 362 al instante 71 con un CV=8.86884
    La nueva disposicion mejora la anterior
    Movemos la actividad 362 al instante 72 con un CV=8.86884
    La nueva disposicion mejora la anterior
    Movemos la actividad 362 al instante 73 con un CV=8.86884
    La nueva disposicion mejora la anterior
    Movemos la actividad 362 al instante 74 con un CV=8.86884
    La nueva disposicion mejora la anterior
    Movemos la actividad 362 al instante 75 con un CV=8.86884
    La nueva disposicion mejora la anterior

```

```

Movemos la actividad 362 al instante 76 con un CV=8.86884
La nueva disposicion mejora la anterior
Movemos la actividad 362 al instante 77 con un CV=8.86884
La nueva disposicion mejora la anterior
Movemos la actividad 362 al instante 78 con un CV=8.86884
La nueva disposicion mejora la anterior
Movemos la actividad 362 al instante 79 con un CV=8.86884
La nueva disposicion mejora la anterior
Movemos la actividad 362 al instante 80 con un CV=8.86884
La nueva disposicion mejora la anterior
Movemos la actividad 362 al instante 81 con un CV=8.86884
La nueva disposicion mejora la anterior
Movemos la actividad 362 al instante 82 con un CV=8.8723
Movemos la actividad 362 al instante 83 con un CV=8.87573
Movemos la actividad 362 al instante 84 con un CV=8.87573
Movemos la actividad 362 al instante 85 con un CV=8.87573
Movemos la actividad 362 al instante 86 con un CV=8.87573
Movemos la actividad 362 al instante 87 con un CV=8.87573
Movemos la actividad 362 al instante 88 con un CV=8.87573
Movemos la actividad 362 al instante 89 con un CV=8.87573
Movemos la actividad 362 al instante 90 con un CV=8.87573
Movemos la actividad 362 al instante 91 con un CV=8.87573
Movemos la actividad 362 al instante 92 con un CV=8.87573
Movemos la actividad 362 al instante 93 con un CV=8.87573
Movemos la actividad 362 al instante 94 con un CV=8.87573
Movemos la actividad 362 al instante 95 con un CV=8.87573
Movemos la actividad 362 al instante 96 con un CV=8.87573
Movemos la actividad 362 al instante 97 con un CV=8.87573
Movemos la actividad 362 al instante 98 con un CV=8.87573
Movemos la actividad 362 al instante 99 con un CV=8.86999
Movemos la actividad 362 al instante 100 con un CV=8.864
La nueva disposicion mejora la anterior
Movemos la actividad 362 al instante 101 con un CV=8.86649
Movemos la actividad 362 al instante 102 con un CV=8.86884
Movemos la actividad 362 al instante 103 con un CV=8.86884
Movemos la actividad 362 al instante 104 con un CV=8.86884
Movemos la actividad 362 al instante 105 con un CV=8.86884
Movemos la actividad 362 al instante 106 con un CV=8.86884
Movemos la actividad 362 al instante 107 con un CV=8.86884
Movemos la actividad 362 al instante 108 con un CV=8.86884
Movemos la actividad 362 al instante 109 con un CV=8.86884
Movemos la actividad 362 al instante 110 con un CV=8.86884
Movemos la actividad 362 al instante 111 con un CV=8.86014
La nueva disposicion mejora la anterior
Movemos la actividad 362 al instante 112 con un CV=8.85131
La nueva disposicion mejora la anterior
Movemos la actividad 362 al instante 113 con un CV=8.85131
La nueva disposicion mejora la anterior
Movemos la actividad 362 al instante 114 con un CV=8.85486
Movemos la actividad 362 al instante 115 con un CV=8.85838
Movemos la actividad 362 al instante 116 con un CV=8.85838
Movemos la actividad 362 al instante 117 con un CV=8.85662
Movemos la actividad 362 al instante 118 con un CV=8.85486
Movemos la actividad 362 al instante 119 con un CV=8.85309
362 se realiza en 113
-----
361 TMin Original=46 H.Libre=0:
    361 se realiza en 45
-----
322 TMin Original=49 H.Libre=0:
    322 se realiza en 48
-----
352 TMin Original=46 H.Libre=3:
    Movemos la actividad 352 al instante 46 con un CV=8.84226
    La nueva disposicion mejora la anterior
    Movemos la actividad 352 al instante 47 con un CV=8.83316
    La nueva disposicion mejora la anterior
    Movemos la actividad 352 al instante 48 con un CV=8.82403
    La nueva disposicion mejora la anterior
    352 se realiza en 48
-----
321 TMin Original=43 H.Libre=0:
    321 se realiza en 42
-----
31 TMin Original=43 H.Libre=73:
    Movemos la actividad 31 al instante 43 con un CV=8.81066
    La nueva disposicion mejora la anterior
    Movemos la actividad 31 al instante 44 con un CV=8.79682
    La nueva disposicion mejora la anterior
    Movemos la actividad 31 al instante 45 con un CV=8.7825
    La nueva disposicion mejora la anterior
    Movemos la actividad 31 al instante 46 con un CV=8.81076
    Movemos la actividad 31 al instante 47 con un CV=8.83853

```

```

Movemos la actividad 31 al instante 48 con un CV=8.86585
Movemos la actividad 31 al instante 49 con un CV=8.90746
Movemos la actividad 31 al instante 50 con un CV=8.94768
Movemos la actividad 31 al instante 51 con un CV=8.9866
Movemos la actividad 31 al instante 52 con un CV=8.92971
Movemos la actividad 31 al instante 53 con un CV=8.87153
Movemos la actividad 31 al instante 54 con un CV=8.81197
Movemos la actividad 31 al instante 55 con un CV=8.75435
La nueva disposicion mejora la anterior
Movemos la actividad 31 al instante 56 con un CV=8.69514
La nueva disposicion mejora la anterior
Movemos la actividad 31 al instante 57 con un CV=8.63416
La nueva disposicion mejora la anterior
Movemos la actividad 31 al instante 58 con un CV=8.63761
Movemos la actividad 31 al instante 59 con un CV=8.64104
Movemos la actividad 31 al instante 60 con un CV=8.64446
Movemos la actividad 31 al instante 61 con un CV=8.64446
Movemos la actividad 31 al instante 62 con un CV=8.64446
Movemos la actividad 31 al instante 63 con un CV=8.64446
Movemos la actividad 31 al instante 64 con un CV=8.64446
Movemos la actividad 31 al instante 65 con un CV=8.64446
Movemos la actividad 31 al instante 66 con un CV=8.64446
Movemos la actividad 31 al instante 67 con un CV=8.64446
Movemos la actividad 31 al instante 68 con un CV=8.64446
Movemos la actividad 31 al instante 69 con un CV=8.64446
Movemos la actividad 31 al instante 70 con un CV=8.64446
Movemos la actividad 31 al instante 71 con un CV=8.64446
Movemos la actividad 31 al instante 72 con un CV=8.64446
Movemos la actividad 31 al instante 73 con un CV=8.64446
Movemos la actividad 31 al instante 74 con un CV=8.64446
Movemos la actividad 31 al instante 75 con un CV=8.64446
Movemos la actividad 31 al instante 76 con un CV=8.64446
Movemos la actividad 31 al instante 77 con un CV=8.64446
Movemos la actividad 31 al instante 78 con un CV=8.64785
Movemos la actividad 31 al instante 79 con un CV=8.65123
Movemos la actividad 31 al instante 80 con un CV=8.6546
Movemos la actividad 31 al instante 81 con un CV=8.65795
Movemos la actividad 31 al instante 82 con un CV=8.66128
Movemos la actividad 31 al instante 83 con un CV=8.6646
Movemos la actividad 31 al instante 84 con un CV=8.6646
Movemos la actividad 31 al instante 85 con un CV=8.67945
Movemos la actividad 31 al instante 86 con un CV=8.69414
Movemos la actividad 31 al instante 87 con un CV=8.7087
Movemos la actividad 31 al instante 88 con un CV=8.72311
Movemos la actividad 31 al instante 89 con un CV=8.73738
Movemos la actividad 31 al instante 90 con un CV=8.75153
Movemos la actividad 31 al instante 91 con un CV=8.75153
Movemos la actividad 31 al instante 92 con un CV=8.75153
Movemos la actividad 31 al instante 93 con un CV=8.75153
Movemos la actividad 31 al instante 94 con un CV=8.75153
Movemos la actividad 31 al instante 95 con un CV=8.73515
Movemos la actividad 31 al instante 96 con un CV=8.71855
Movemos la actividad 31 al instante 97 con un CV=8.70089
Movemos la actividad 31 al instante 98 con un CV=8.68308
Movemos la actividad 31 al instante 99 con un CV=8.6651
Movemos la actividad 31 al instante 100 con un CV=8.64697
Movemos la actividad 31 al instante 101 con un CV=8.6458
Movemos la actividad 31 al instante 102 con un CV=8.64446
Movemos la actividad 31 al instante 103 con un CV=8.64446
Movemos la actividad 31 al instante 104 con un CV=8.64446
Movemos la actividad 31 al instante 105 con un CV=8.64446
Movemos la actividad 31 al instante 106 con un CV=8.64446
Movemos la actividad 31 al instante 107 con un CV=8.6458
Movemos la actividad 31 al instante 108 con un CV=8.65392
Movemos la actividad 31 al instante 109 con un CV=8.66197
Movemos la actividad 31 al instante 110 con un CV=8.71104
Movemos la actividad 31 al instante 111 con un CV=8.75833
Movemos la actividad 31 al instante 112 con un CV=8.80399
Movemos la actividad 31 al instante 113 con un CV=8.79598
Movemos la actividad 31 al instante 114 con un CV=8.78081
Movemos la actividad 31 al instante 115 con un CV=8.76372
31 se realiza en 57

-----
351 TMin Original=43 H.Libre=0:
351 se realiza en 42
-----
26 TMin Original=39 H.Libre=0:
26 se realiza en 38
-----
23 TMin Original=33 H.Libre=0:
23 se realiza en 32
-----
45 TMin Original=33 H.Libre=63:
Movemos la actividad 45 al instante 33 con un CV=8.61262

```

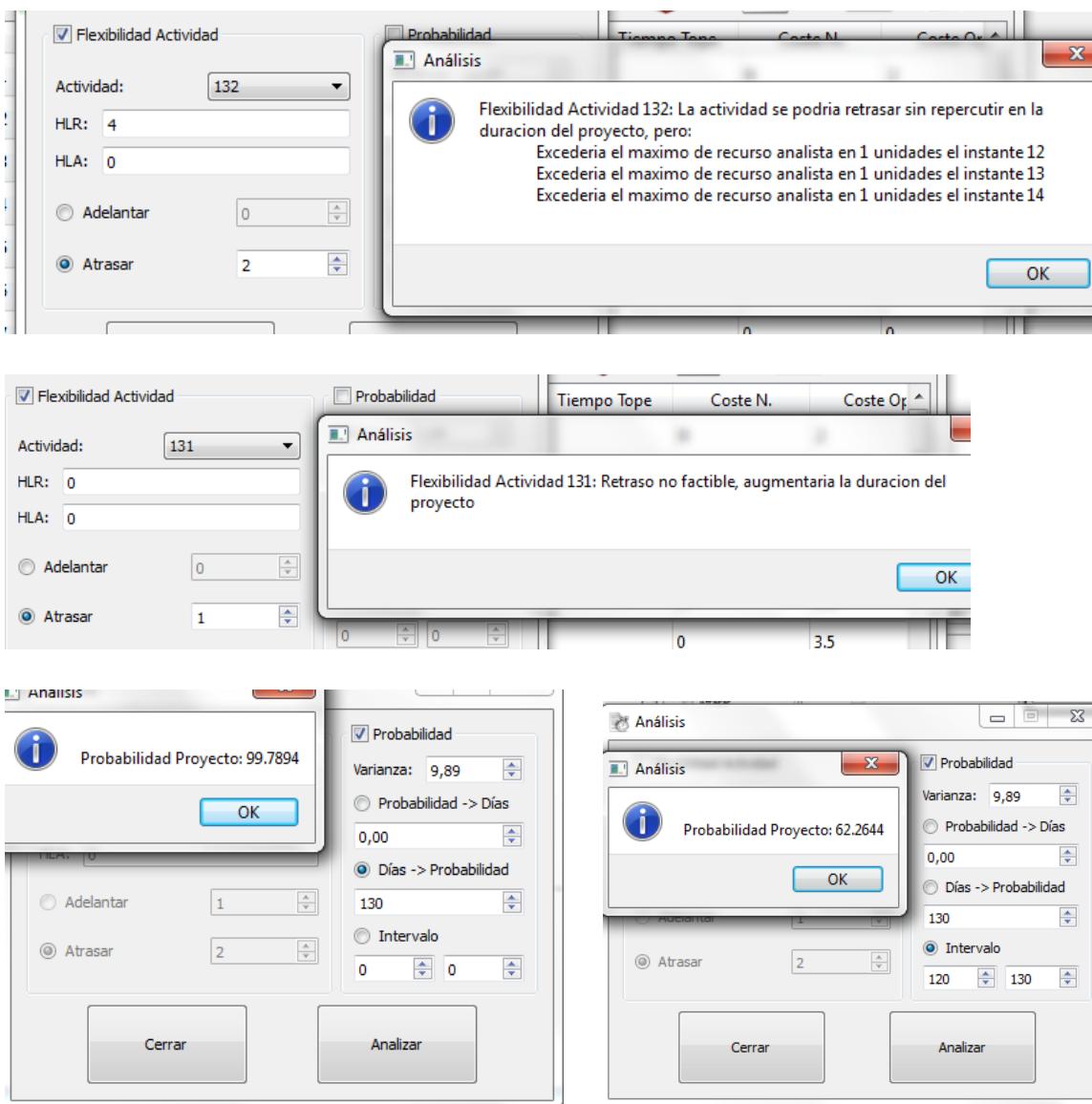
```

La nueva disposicion mejora la anterior
Movemos la actividad 45 al instante 34 con un CV=8.62721
Movemos la actividad 45 al instante 35 con un CV=8.66273
Movemos la actividad 45 al instante 36 con un CV=8.69745
Movemos la actividad 45 al instante 37 con un CV=8.73142
Movemos la actividad 45 al instante 38 con un CV=8.74046
Movemos la actividad 45 al instante 39 con un CV=8.71584
Movemos la actividad 45 al instante 40 con un CV=8.69081
Movemos la actividad 45 al instante 41 con un CV=8.66578
Movemos la actividad 45 al instante 42 con un CV=8.64031
Movemos la actividad 45 al instante 43 con un CV=8.64074
Movemos la actividad 45 al instante 44 con un CV=8.6333
Movemos la actividad 45 al instante 45 con un CV=8.6258
Movemos la actividad 45 al instante 46 con un CV=8.61781
Movemos la actividad 45 al instante 47 con un CV=8.63815
Movemos la actividad 45 al instante 48 con un CV=8.65837
Movemos la actividad 45 al instante 49 con un CV=8.6863
Movemos la actividad 45 al instante 50 con un CV=8.71397
Movemos la actividad 45 al instante 51 con un CV=8.74138
Movemos la actividad 45 al instante 52 con un CV=8.74138
    Se supera la limitacion de recursos
Movemos la actividad 45 al instante 53 con un CV=1e+012
    Se supera la limitacion de recursos
Movemos la actividad 45 al instante 54 con un CV=1e+012
    Se supera la limitacion de recursos
Movemos la actividad 45 al instante 55 con un CV=1e+012
    Se supera la limitacion de recursos
Movemos la actividad 45 al instante 56 con un CV=1e+012
    Se supera la limitacion de recursos
Movemos la actividad 45 al instante 57 con un CV=1e+012
    Se supera la limitacion de recursos
Movemos la actividad 45 al instante 58 con un CV=1e+012
    Se supera la limitacion de recursos
Movemos la actividad 45 al instante 59 con un CV=1e+012
Movemos la actividad 45 al instante 60 con un CV=8.71262
Movemos la actividad 45 al instante 61 con un CV=8.68385
Movemos la actividad 45 al instante 62 con un CV=8.65461
Movemos la actividad 45 al instante 63 con un CV=8.62488
Movemos la actividad 45 al instante 64 con un CV=8.62488
Movemos la actividad 45 al instante 65 con un CV=8.62488
Movemos la actividad 45 al instante 66 con un CV=8.62488
Movemos la actividad 45 al instante 67 con un CV=8.62488
Movemos la actividad 45 al instante 68 con un CV=8.62488
Movemos la actividad 45 al instante 69 con un CV=8.62488
Movemos la actividad 45 al instante 70 con un CV=8.62488
Movemos la actividad 45 al instante 71 con un CV=8.62488
Movemos la actividad 45 al instante 72 con un CV=8.62488
Movemos la actividad 45 al instante 73 con un CV=8.62488
Movemos la actividad 45 al instante 74 con un CV=8.62488
Movemos la actividad 45 al instante 75 con un CV=8.62488
Movemos la actividad 45 al instante 76 con un CV=8.62488
Movemos la actividad 45 al instante 77 con un CV=8.62488
Movemos la actividad 45 al instante 78 con un CV=8.62488
Movemos la actividad 45 al instante 79 con un CV=8.62574
Movemos la actividad 45 al instante 80 con un CV=8.62659
Movemos la actividad 45 al instante 81 con un CV=8.62745
Movemos la actividad 45 al instante 82 con un CV=8.6283
Movemos la actividad 45 al instante 83 con un CV=8.62916
Movemos la actividad 45 al instante 84 con un CV=8.62916
Movemos la actividad 45 al instante 85 con un CV=8.62916
Movemos la actividad 45 al instante 86 con un CV=8.62916
Movemos la actividad 45 al instante 87 con un CV=8.62916
Movemos la actividad 45 al instante 88 con un CV=8.62916
Movemos la actividad 45 al instante 89 con un CV=8.62916
Movemos la actividad 45 al instante 90 con un CV=8.62916
Movemos la actividad 45 al instante 91 con un CV=8.62916
Movemos la actividad 45 al instante 92 con un CV=8.62916
Movemos la actividad 45 al instante 93 con un CV=8.62916
Movemos la actividad 45 al instante 94 con un CV=8.62916
Movemos la actividad 45 al instante 95 con un CV=8.62916
45 se realiza en 33
-----
24 TMin Original=28 H.Libre=4:
    Movemos la actividad 24 al instante 28 con un CV=8.61613
    Movemos la actividad 24 al instante 29 con un CV=8.61927
    Movemos la actividad 24 al instante 30 con un CV=8.62205
    Movemos la actividad 24 al instante 31 con un CV=8.62445
    24 se realiza en 27
-----
25 TMin Original=28 H.Libre=0:
    25 se realiza en 27
-----
22 TMin Original=28 H.Libre=0:
    22 se realiza en 27

```

```
21  TMin Original=25  H.Libre=0:  
    21 se realiza en 24  
-----  
136  TMin Original=17  H.Libre=0:  
    136 se realiza en 16  
-----  
133  TMin Original=13  H.Libre=0:  
    133 se realiza en 12  
-----  
135  TMin Original=13  H.Libre=3:  
    Movemos la actividad 135 al instante 13 con un CV=8.61262  
    La nueva disposicion mejora la anterior  
    Movemos la actividad 135 al instante 14 con un CV=8.61262  
    La nueva disposicion mejora la anterior  
    Movemos la actividad 135 al instante 15 con un CV=8.61262  
    La nueva disposicion mejora la anterior  
    135 se realiza en 15  
-----  
134  TMin Original=9  H.Libre=4:  
    Movemos la actividad 134 al instante 9 con un CV=8.60289  
    La nueva disposicion mejora la anterior  
    Movemos la actividad 134 al instante 10 con un CV=8.5931  
    La nueva disposicion mejora la anterior  
    Movemos la actividad 134 al instante 11 con un CV=8.58323  
    La nueva disposicion mejora la anterior  
        Se supera la limitacion de recursos  
    Movemos la actividad 134 al instante 12 con un CV=1e+012  
    134 se realiza en 11  
-----  
132  TMin Original=9  H.Libre=4:  
    Se supera la limitacion de recursos  
    Movemos la actividad 132 al instante 9 con un CV=1e+012  
        Se supera la limitacion de recursos  
    Movemos la actividad 132 al instante 10 con un CV=1e+012  
        Se supera la limitacion de recursos  
    Movemos la actividad 132 al instante 11 con un CV=1e+012  
        Se supera la limitacion de recursos  
    Movemos la actividad 132 al instante 12 con un CV=1e+012  
    132 se realiza en 8  
-----  
131  TMin Original=3  H.Libre=0:  
    131 se realiza en 2  
-----  
12  TMin Original=2  H.Libre=0:  
    12 se realiza en 1  
-----  
11  TMin Original=1  H.Libre=0:  
    11 se realiza en 0
```

5.2.6 Análisis



6 Conclusiones Finales

Gracias a este trabajo hemos podido apreciar de forma directa la importancia de una buena gestión de proyectos. Desde el primer momento hemos querido abordar el trabajo como un proyecto real con el objetivo de crear una herramienta eficaz, ligera y manejable enfocada a usuarios reales. Por eso hicimos nuestro propio plan de gestión de proyecto, intentando cumplir las metas de entrega que nos proponíamos y trabajando juntos ante los problemas de desarrollo.

Sin duda alguna ambos coincidimos en que ha sido uno de los mayores proyectos que hemos desarrollado a lo largo de nuestra estancia en la Universidad. En este proyecto se han visto aplicados conocimientos de muchas áreas de la informática, gestión de la información, ingeniería del software, ingeniería de requisitos, gestión de proyectos informáticos, metodología y técnicas de la programación entre otras. Por todo esto concluimos felices por un buen trabajo realizado y que esperamos cumpla con su objetivo.

7 Anexos

A continuación hemos añadido tres anexos que creemos pueden ser de interés. El primero y el segundo son los códigos fuentes originales y **completos**, el último es un manual de usuario al que se puede acceder desde el menú de ayuda de la aplicación.

Informamos también, que pese a haber incluido todos los códigos fuente utilizados para el desarrollo de este programa, recomendamos encarecidamente su lectura directamente desde un entorno de desarrollo (Visual Studio, Notepad++) con los códigos que se encuentran en la carpeta *Source Files* del CD que adjuntamos.

7.1 Código Interfaz

7.1.1 ActivityView.h

```
#ifndef ACTIVITYVIEW_H
#define ACTIVITYVIEW_H

#include <QWidget>
#include <qevent.h>
#include <QTableWidget>
#include <QMMessageBox>
#include <QDialog>
#include <QCheckBox>
#include <QLineEdit>
#include <QLabel>
#include <QSpinBox>
#include <QGroupBox>
#include <QLayout>
#include "Project.h"
#include "Exceptions.h"

namespace Ui {
class ActivityView;
}

class ActivityView : public QWidget
{
    Q_OBJECT

public:
    explicit ActivityView(Project *project, QTableWidget *tabla, QWidget *parent = 0);
    explicit ActivityView(Project *project, Activity *activ,QTableWidget *tabla, int rowtoupdate,
    QWidget *parent = 0);
    QTableWidget *table;
    Project *pro;
    Activity *act;
    int row2upd;

    ~ActivityView();

private slots:
    void SaveDataClose();
    void UpdateDataClose();
    void ShowPred();
    void ShowRes();
    void SaveRes();
    void SavePred();
    //void SavePred();

protected:
    void closeEvent(QCloseEvent * event){
        if(ResViewVisible) asigResView->close();
        if(PredViewVisible) asigPredView->close();
        this->close();
    }

private:
    Ui::ActivityView *ui;

    void SetValues();
    void ClearData();
    void connectAll();
    void createResWindow();
    void createPredWindow();

    QWidget *asigResView, *asigPredView;
    bool ResViewVisible, PredViewVisible, edit;
    bool firstRes, firstPred;
    int nSize, resSize;
    QPushButton *resClose, *resSave, *predClose, *predSave;
    QVBoxLayout *toShow;
    QVector<QSpinBox*> spiners; QVector<QLabel*> labels;
    QVector<QCheckBox*> checks;
};

#endif // ACTIVITYVIEW_H
```

7.1.2 ActivityView.cpp

```
#include "ActivityView.h"
#include "ui_activityview.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <QVector>
ActivityView::ActivityView(Project *project, QTableWidget *tabla, QWidget *parent) :
    QWidget(parent),
    ui(new Ui::ActivityView)
{
    this->table=tabla; this->pro=project;
    this->setWindowIcon(QIcon("./icons/management.png")); this->setWindowTitle(QString(tr("Project Manager")));
    nSize = pro->getActivities().size(); resSize = pro->getResources()->size();
    toShow = new QVBoxLayout();
    ResViewVisible=PredViewVisible=edit=firstRes=firstPred=false;
    ui->setupUi(this); connectAll();
    connect(ui->save_Button,SIGNAL(clicked()),this,SLOT(SaveDataClose()));
}
ActivityView::ActivityView(Project *project,Activity *activ, QTableWidget *tabla,int rowtoupdate, QWidget *parent) :
    QWidget(parent),
    ui(new Ui::ActivityView)
{
    this->act=activ; this->row2upd=rowtoupdate; this->table=tabla;
    this->pro=project; ui->setupUi(this); nSize = pro->getActivities().size();
    resSize = pro->getResources()->size(); ResViewVisible=PredViewVisible=firstRes=firstPred=false;
    edit=true; SetValues(); connectAll();
    connect(ui->save_Button,SIGNAL(clicked()),this,SLOT(UpdateDataClose()));
}
ActivityView::~ActivityView()
{
    delete ui;
}
//Building window
void ActivityView::connectAll()
{
    connect(ui->PredBut,SIGNAL(clicked()),this,SLOT>ShowPred());
    connect(ui->AsignarRec_Button,SIGNAL(clicked()),this,SLOT>ShowRes());
    connect(ui->canc_Button,SIGNAL(clicked()),this,SLOT(close()));
}
void ActivityView::createPredWindow()
{
    asigPredView = new QWidget();

    nSize = pro->getActivities().size();
    for (int i=0;i<nSize;i++){ //Inicialización Vector;
        QCheckBox *check = new QCheckBox();
        std::vector<Activity*> vecActs = pro->getActivities();
        //std::string text = pro->activities.at(i)->getName();
        std::string text = vecActs.at(i)->getName();
        if(edit){
            if(QString::fromStdString(text)!=ui->NomAct->text()){
                check->setText(QString::fromStdString(vecActs.at(i)->getName()));
                if(act->getPositionRelationPredecesor(check->text().toStdString())>=0)
                    check->setChecked(true);
                checks.insert(i,check);
            }else{
                check->setText(QString::fromStdString(vecActs.at(i)->getName()));
                checks.insert(i,check); checks[i]->setEnabled(false);
            }
        }else{
            check->setText(QString::fromStdString(vecActs.at(i)->getName())); checks.insert(i,check);
        }
    }
    int j = 0; int it=0;
    if(nSize>=20) j=5; //mayor de 20 -> de 5 en 5
    else if(nSize>=10) j=4; //mayor de 10 -> de 4 en 4
    else if(nSize>6) j=3; //mayor de 5-> de 3 en 3.
    else j=2; //resto de 2-> de 2 en 2.
    QGridLayout *grid = new QGridLayout();
    for (int i=0;i<nSize;i++){
        QGroupBox *first = new QGroupBox(); //Solo para agrupar
        QHBoxLayout *hbox = new QHBoxLayout();
        hbox->addWidget(checks[i]);
        first->setLayout(hbox); // first->setTitle(tr("Predecesores: "));
        grid->addWidget(first,(i/j),it);
        it++;
        if(it==j) it=0;
    }
}
```

```

QGroupBox *group = new QGroupBox(tr("Asignar: "));
group->setLayout(grid);
QHBoxLayout *hbuts = new QHBoxLayout();
predClose = new QPushButton(tr("Cerrar")); predClose->setFixedSize(80,40);
predSave = new QPushButton(tr("Guardar")); predSave->setFixedSize(80,40);
hbuts->addWidget(predClose); hbuts->addWidget(predSave);
connect(predClose,SIGNAL(clicked()),asigPredView,SLOT(close()));
connect(predSave,SIGNAL(clicked()),this,SLOT(SavePred()));

QVBoxLayout *vbox = new QVBoxLayout();
vbox->addWidget(group);
vbox->addLayout(hbuts);
asigPredView->setLayout(vbox); asigPredView->setMinimumWidth(260);
asigPredView->setWindowTitle(tr("Asignar Predecesores"));
asigPredView->show();
asigPredView->setFixedSize(asigPredView->size());
PredViewVisible = true;
}

void ActivityView::createResWindow()
{
    asigResView = new QWidget();
    resSize = pro->getResources()->size();
    std::vector<Resource*> *resVec = pro->getResources();
    for(int i=0;i<resSize;i++)
    {
        QLabel *lab=new QLabel(QString::fromStdString(resVec->at(i)->getName()));
        labels.insert(i,lab);
        QSpinBox *spin = new QSpinBox();
        spin->setMaximum(resVec->at(i)->getUnitsMax());
        if(edit){
            if(act->getResource(lab->text().toStdString())){
                int n = act->getResource(lab->text().toStdString())->units_asig;
                spin->setValue(n);
            }
        }
        spinners.insert(i,spin);
    }

    QGridLayout *grid = new QGridLayout();
    int j = 0; int it=0;
    if(resSize>=20) j=5; //mayor de 20 -> de 5 en 5
    else if(resSize>=10) j=4; //mayor de 10 -> de 4 en 4
    else if(resSize>6) j=3; //mayor de 5-> de 3 en 3.
    else j=2; //resto de 2-> de 2 en 2.
    for (int i=0;i<resSize;i++){
        QGroupBox *first = new QGroupBox(); //Solo para agrupar
        spinners[i]->setMinimum(0);
        spinners[i]->setMaximumWidth(50);
        QHBoxLayout *hbox = new QHBoxLayout();
        hbox->addWidget(labels[i]);
        hbox->addWidget(spinners[i]);
        first->setLayout(hbox);
        first->setTitle(tr("Recursos: "));

        grid->addWidget(first,(i/j),it);
        it++;
        if(it==j) it=0;
    }
    QGroupBox *group = new QGroupBox(tr("Asignar: "));
    group->setLayout(grid);
    QHBoxLayout *hbuts = new QHBoxLayout();
    resClose = new QPushButton(tr("Cancelar"));
    resSave = new QPushButton(tr("Guardar"));
    hbuts->addWidget(resClose); resClose->setFixedSize(80,40);
    hbuts->addWidget(resSave); resSave->setFixedSize(80,40);
    connect(resClose,SIGNAL(clicked()),asigResView,SLOT(close()));
    connect(resSave,SIGNAL(clicked()),this,SLOT(SaveRes()));

    QVBoxLayout *vbox = new QVBoxLayout();
    vbox->addWidget(group); vbox->addLayout(hbuts);

    asigResView->setLayout(vbox); asigResView->setMinimumWidth(260);
    asigResView->setWindowTitle(tr("Asignar Recursos"));
    asigResView->show();
    asigResView->setFixedSize(asigResView->size());
    ResViewVisible=true;
}

void ActivityView::ShowPred()
{
}

```

```

if(nSize>0 && !PredViewVisible){    createPredWindow(); firstPred=true;}
else if(PredViewVisible){  asigPredView->close(); PredViewVisible=false;}
else{
    QMessageBox mes;
    mes.information(0,QString(tr("Aviso")),QString(tr("No hay predecesores a asignar")));
}

}
void ActivityView::ShowRes()
{
    if(resSize>0 && !ResViewVisible){    createResWindow(); firstRes=true;}
    else if(ResViewVisible){  asigResView->close(); ResViewVisible=false;}
    else{
        QMessageBox mes;
        mes.information(0,QString(tr("Aviso")),QString(tr("No hay recursos a asignar")));
    }
}
void ActivityView::ClearData()
{
    ui->NomAct->clear();
    ui->DurAct->setValue(0);  ui->RetardAct->setValue(0);
    ui->PriceSpin->setValue(0.00);  ui->OverRunSpin->setValue(0.00);
}
void ActivityView::SetValues()
{
    std::string auxname = act->getName();
    ui->NomAct->setText(QString::fromStdString(auxname));
    ui->NomAct->setDisabled(true);
    ui->DurAct->setValue(act->getTNormal());
    ui->RetardAct->setValue(act->getTNormal()-act->getTTope());
    ui->PriceSpin->setValue(act->getCostNormal());
    ui->OverRunSpin->setValue(act->getOportunityCost());

}
//Saving Data
void ActivityView::SavePred()
{
    for(int i=0;i<nSize;i++)
    {
        checks[i]->setChecked(checks[i]->isChecked());
    }

    PredViewVisible=false;
    asigPredView->close();
}
void ActivityView::SaveRes()
{
    for(int i=0;i<resSize;i++)
    {
        spiners[i]->setValue(spiners[i]->text().toInt());
    }
    asigResView->close();
    ResViewVisible=false;
}
void ActivityView::UpdateDataClose()
{
    std::string acname = table->item(row2upd,0)->text().toStdString();
    int tn = ui->DurAct->value();
    int tp= tn - ui->RetardAct->value();
    if(tp<0){
        QMessageBox mes;
        mes.information(this,QString(tr("Aviso")),QString(tr("El retraso no puede ser mayor que la duración.")));
        mes.show();
    }else{
        //Table->setItem(row2upd,0,new QTableWidgetItem(ui->NomAct->text()));
        pro->modifyActivity(acname,tn,tp,ui->PriceSpin->value(),ui->OverRunSpin->value());
        table->setItem(row2upd,2,new QTableWidgetItem(ui->DurAct->text()));
        table->setItem(row2upd,4,new QTableWidgetItem(QString::number(tp)));
        table->setItem(row2upd,5,new QTableWidgetItem(ui->PriceSpin->text()));
        table->setItem(row2upd,6,new QTableWidgetItem(ui->OverRunSpin->text()));

        //Resources
        QString *textRes = new QString("");
        bool any=false;
        if(firstRes){
            for(int i=0;i<resSize;i++){
                if(!act->getResource(labels[i]->text().toStdString())){
                    if(spiners[i]->value()>0){
                        //If we add by first time this resource to the activity it must be created.
                        string nameRes = labels[i]->text().toStdString();  int units = spiners[i]->value();
                        pro->allocateResourceActivity(nameRes,act->getName(),units);}
                }
            }
        }
    }
}

```

```

}else{
    int newValue = (spinners[i]->value()) - (act->getResource(labels[i]->text().toStdString())->units_asig);
    if(newValue>0) pro->incrementUnitsResourceActivity(labels[i]->text().toStdString(),act->getName(),newValue);
    if(newValue<0) pro->decrementUnitsResourceActivity(labels[i]->text().toStdString(),act->getName(),-newValue);
}
if(spinners[i]->value()>0){
    if(any) textRes->append(" ");
    textRes->append(labels[i]->text().append("("));
    textRes->append(spinners[i]->text().append(")"));
    any=true;
}
}
table->setItem(row2upd,1,new QTableWidgetItem(*textRes));

}
table->setItem(row2upd,2,new QTableWidgetItem(ui->DurAct->text()));

//Predecesores
any=false;
QString *textPred = new QString("");
if(firstPred){
    for(int i=0;i<nSize;i++){
        if(checks[i]->isChecked()){
            if(act->getPositionRelationPredecesor(checks[i]->text().toStdString())<0) {
                pro->addRelation(checks[i]->text().toStdString(),act->getName());
            }
            if(any) textPred->append("-");
            any=true;
            textPred->append(checks[i]->text());
        }else{
            if(act->getPositionRelationPredecesor(checks[i]->text().toStdString())>=0) {
                pro->deleteRelation(checks[i]->text().toStdString(),act->getName());
            }
        }
    }
    table->setItem(row2upd,3,new QTableWidgetItem(*textPred));
}
ClearData();
this->close();
}
}
void ActivityView::SaveDataClose()
{
if(!(ui->NomAct->text().isEmpty())){
    if(!!(pro->getActivity(ui->NomAct->text().toStdString()))){
        int tn = ui->DurAct->value();
        int tp= tn - ui->RetardAct->value();
        if(tp<0){
            QMessageBox mes;
            mes.information(this,QString(tr("Aviso")),QString(tr("El retraso no puede ser mayor que la duración.")));
            mes.show();
        }else{
            int aux = table->rowCount();
            table->setRowCount(aux+1);
            table->setItem(aux,0,new QTableWidgetItem(ui->NomAct->text()));
            //tn = duracio int ; tp = duracio - retrasoMax; cnormal = cost activitat cuoport=sobrecost
            pro->addActivity((ui->NomAct->text()).toStdString(),tn,tp,ui->PriceSpin->value(),ui->OverRunSpin->value());
            table->setItem(aux,4,new QTableWidgetItem(QString::number(tp)));
            table->setItem(aux,5,new QTableWidgetItem(ui->PriceSpin->text()));
            table->setItem(aux,6,new QTableWidgetItem(ui->OverRunSpin->text()));
        }
        //Resources
        QString *textRes = new QString("");
        bool any=false;
        if(firstRes){
            for(int i=0;i<resSize;i++){
                if(spinners[i]->value()>0){
                    if(any) textRes->append(" ");
                    textRes->append(labels[i]->text().append("("));
                    textRes->append(spinners[i]->text().append(")"));
                    string nameRes = labels[i]->text().toStdString();
                    string nameAct = ui->NomAct->text().toStdString();
                    int units = spinners[i]->value();
                    pro->allocateResourceActivity(nameRes,nameAct,units);
                    any=true;
                }
            }
        }
        table->setItem(aux,1,new QTableWidgetItem(*textRes));           table->setItem(aux,2,new QTableWidgetItem(ui->DurAct->text()));
        //Predecesores
        any=false;
        QString *textPred = new QString("");
    }
}

```

```

if(firstPred){
    for(int i=0;i<nSize;i++){
        if(checks[i]->isChecked()){
            if(any) textPred->append("-");
            any=true;
            textPred->append(chcks[i]->text());
            pro->addRelation(chcks[i]->text().toStdString(),ui->NomAct->text().toStdString());
        }
    }
    table->setItem(aux,3,new QTableWidgetItem(*textPred));      ClearData();      this->close();
}
}else{      showErrorInfo(E_ACTIVITY_ALREADY_EXIST,"");      }
}else{
    QMessageBox *help = new QMessageBox();
    help->information(0,QString("Ayuda"),QString("Dale nombre a la actividad"));
}
}
}

```

7.1.3 Calendar.h

```

#ifndef CALENDAR_H
#define CALENDAR_H

#include <QWidget>
#include <QCalendarWidget>
#include <QTextCharFormat>
#include <QPainter>
#include <QPen>
#include <QBrush>
#include <QString>
#include <QDate>
#include <QVector>
#include <qevent.h>
#include <InsertDate.h>

namespace Ui {
class calendar;
}

class calendar : public QWidget
{
    Q_OBJECT

public:
    explicit calendar(QVector<QDate> *fest, QVector<QDate> *labo,QString *weekEnd, QWidget *parent = 0);
    ~calendar();
    void getLabo();
    void getFest();

private slots:
    void LoadCal();  void UpdateCal();
    void AddExtra(); void DeleteExtra(); void AddFestivo(); void DeleteFestivo();
    void ChangeFesRow(int row, int col); void ChangeLabRow(int row, int col);
    void PaintCal(QDate day,bool LabFes);
    void aboutWeekends();

protected:
    void closeEvent(QCloseEvent * event){
        if(!addWin) insertWindow->close();
        this->close();
    }

private:
    QVector<QDate> *festivos, *laborables;
    int tam_festivos, tam_laborables;
    QString *weekEnd;
    QTextCharFormat *original1, *original2;
    bool addWin;
    InsertDate *insertWindow;
    QDate ex;
    QTextCharFormat format;
    QDate m_currentDate; QPen m_outlinePen; QBrush m_transparentBrush;
    QBrush brus; QPainter painter; QRect rect;
};

#endif // CALENDAR_H

```

7.1.4 Calendar.cpp

```
#include "calendar.h"
#include "ui_calendar.h"
#include "stdio.h"

calendar::calendar(QVector<QDate> *fest, QVector<QDate> *labo,QString *weekEnd, QWidget *parent) :
    QWidget(parent),
    ui(new Ui::calendar)
{
    ui->setupUi(this);
    tam_festivos=0; tam_laborables=0;
    addWin = true;
    this->festivos=fest;
    this->laborables=labo;
    this->weekEnd=weekEnd;
    this->setWindowIcon(QIcon("./icons/management.png"));
    this->setWindowTitle(QString(tr("Project Manager")));

    ui->FesTable->setEditTriggers(QTableWidget::NoEditTriggers);
    ui->LabTable->setEditTriggers(QTableWidget::NoEditTriggers);

    LoadCal();

    //ICONS
    ui->addFes_Button->setIcon(QPixmap("./icons/add.png")); ui->addFes_Button->setIconSize(QSize(40,40));
    ui->addLab_Button->setIcon(QPixmap("./icons/add.png")); ui->addLab_Button->setIconSize(QSize(40,40));
    ui->delLab_Button->setIcon(QPixmap("./icons/minus.png")); ui->delLab_Button->setIconSize(QSize(40,40));
    ui->delFes_Button->setIcon(QPixmap("./icons/minus.png")); ui->delFes_Button->setIconSize(QSize(40,40));
    ui->updLab_Button->setIcon(QPixmap("./icons/update.png")); ui->updLab_Button->setIconSize(QSize(40,40));
    ui->updFes_Button->setIcon(QPixmap("./icons/update.png")); ui->updFes_Button->setIconSize(QSize(40,40));
    //Tooltips
    ui->addFes_Button->setToolTip(QString("Añadir Festivo")); ui->addLab_Button->setToolTip(QString("Añadir Laborable"));
    ui->updFes_Button->setToolTip(QString("Mostrar en calendario")); ui->updLab_Button->setToolTip(QString("Mostrar en calendario"));
    ui->delFes_Button->setToolTip(QString("Eliminar Festivo")); ui->delFes_Button->setToolTip(QString("Eliminar Laborable"));
    //Connecting
    connect(ui->addFes_Button,SIGNAL(clicked()),this,SLOT(AddFestivo()));
    connect(ui->delFes_Button,SIGNAL(clicked()),this,SLOT>DeleteFestivo());
    connect(ui->addLab_Button,SIGNAL(clicked()),this,SLOT(AddExtra()));
    connect(ui->delLab_Button,SIGNAL(clicked()),this,SLOT>DeleteExtra());
    connect(ui->updFes_Button,SIGNAL(clicked()),this,SLOT(UpdateCal()));
    connect(ui->updLab_Button,SIGNAL(clicked()),this,SLOT(UpdateCal()));
    connect(ui->Sab_Labo,SIGNAL(clicked()),this,SLOT(aboutWeekends()));
    connect(ui->Dom_Labo,SIGNAL(clicked()),this,SLOT(aboutWeekends()));
    //Tables
    connect(ui->FesTable,SIGNAL(cellClicked(int,int)),this,SLOT(ChangeFesRow(int,int)));
    connect(ui->LabTable,SIGNAL(cellClicked(int,int)),this,SLOT(ChangeLabRow(int,int)));
}

calendar::~calendar()
{
    delete ui;
}

void calendar:: aboutWeekends()
{
    weekEndC->clear();
    if((!ui->Dom_Labo->isChecked())&&(!ui->Sab_Labo->isChecked())) weekEndC->append(QString("NONE"));
    if(ui->Dom_Labo->isChecked()) weekEndC->append(QString("SUN"));
    if(ui->Sab_Labo->isChecked()){
        weekEndC->clear();
        if(ui->Dom_Labo->isChecked()) weekEndC->append(QString("BOTH"));
        else weekEndC->append(QString("SAT"));
    }
}

void calendar:: ChangeFesRow(int row, int col)
{
    ui->FesTable->setCurrentCell(row,col);
    ui->FesTable->selectRow(row);
}

void calendar:: ChangeLabRow(int row, int col)
{
    ui->LabTable->setCurrentCell(row,col);
    ui->LabTable->selectRow(row);
}

void calendar:: LoadCal()
{
    ui->MyCal->setGridVisible(true);
}
```

```

ui->MyCal->setFirstDayOfWeek(Qt::Monday);
getLabo();
getFest();

}

void calendar:: UpdateCal()
{
    ui->MyCal->update();
 //UpdateFestivos
 for(int i=0;i<festivos->size();i++)  PaintCal(festivos->value(i),false);
 //UpdateLaborables
 for(int j=0;j<laborables->size();j++)  PaintCal(laborables->value(j),true);
}
void calendar:: PaintCal(QDate day,bool LabFes) //Rep un día i un bool que determina si el día es Laborable o festivo
{
    if(LabFes){ //Es laborable
        brus.setColor(Qt::black);
        format.setForeground(brus);    format.setBackground(Qt::white);
        ui->MyCal->setDateTextFormat(day,format);
    }else{ //Es festivo
        brus.setColor(Qt::white);
        format.setBackground(Qt::red);    format.setForeground(brus);
        ui->MyCal->setDateTextFormat(day,format);
    }
}
void calendar:: getFest()
{
    /* National holidays      Gener: 1, 6      Març: 19      Maig: 1;     Agost: 15;      Octubre: 12      Novembre: 1      Decembre: 6, 8, 25*/
    ui->FesTable->setRowCount(10);
    ui->FesTable->setItem(0,0,new QTableWidgetItem(QString("01/01"))); ui->FesTable->setItem(0,1,new QTableWidgetItem(QString("Año Nuevo")));
    ui->FesTable->setItem(1,0,new QTableWidgetItem(QString("06/01"))); ui->FesTable->setItem(1,1,new QTableWidgetItem(QString("Día de Reyes")));
    ui->FesTable->setItem(2,0,new QTableWidgetItem(QString("19/03"))); ui->FesTable->setItem(2,1,new QTableWidgetItem(QString("San José")));
    ui->FesTable->setItem(3,0,new QTableWidgetItem(QString("01/05"))); ui->FesTable->setItem(3,1,new QTableWidgetItem(QString("Fiesta del Trabajo")));
    ui->FesTable->setItem(4,0,new QTableWidgetItem(QString("15/08"))); ui->FesTable->setItem(4,1,new QTableWidgetItem(QString("La Asunción")));
    ui->FesTable->setItem(5,0,new QTableWidgetItem(QString("12/10"))); ui->FesTable->setItem(5,1,new QTableWidgetItem(QString("Fiesta Nacional de España")));
    ui->FesTable->setItem(6,0,new QTableWidgetItem(QString("01/11"))); ui->FesTable->setItem(6,1,new QTableWidgetItem(QString("Todos los Santos")));
    ui->FesTable->setItem(7,0,new QTableWidgetItem(QString("06/12"))); ui->FesTable->setItem(7,1,new QTableWidgetItem(QString("Día de la Constitución")));
    ui->FesTable->setItem(8,0,new QTableWidgetItem(QString("08/12"))); ui->FesTable->setItem(8,1,new QTableWidgetItem(QString("Día de la Concepción")));
    ui->FesTable->setItem(9,0,new QTableWidgetItem(QString("25/12"))); ui->FesTable->setItem(9,1,new QTableWidgetItem(QString("Navidad")));
}
void calendar:: AddFestivo(){
    if(addWin){
        insertWindow = new InsertDate(ui->FesTable,festivos);
        insertWindow->show();
        addWin = false;
    }else{
        if(!insertWindow->isVisible()){
            insertWindow = new InsertDate(ui->FesTable,festivos);
            insertWindow->show();
        }
    }
}
void calendar:: DeleteFestivo()
{
    int row2delete = ui->FesTable->currentRow();  int festivosNacionales =10;
    if(row2delete>=0){
        if(row2delete<festivosNacionales){
            QMessageBox::information(this,QString("Aviso"),QString("No se pueden eliminar los Festivos nacionales"
                "\nno debe de introducir un día Laboral EXTRAS en su"
                "\n\nlugar."));
        }else{
            ui->FesTable->removeRow(row2delete);
            if(festivos->value(row2delete-festivosNacionales).dayOfWeek()==Qt::Saturday || festivos->value(row2delete-festivosNacionales).dayOfWeek()==Qt::Sunday)
            {
                ui->MyCal->setDateTextFormat(festivos->value(row2delete-festivosNacionales),ui->MyCal->dateTextFormat(QDate(2012,10,26)));
            }else    ui->MyCal->setDateTextFormat(festivos->value(row2delete-festivosNacionales), ui->MyCal->dateTextFormat(QDate(2012,10,27)));
            festivos->remove(row2delete-festivosNacionales);
            ui->FesTable->setCurrentCell(-1,-1);
        }
    }else{
        QMessageBox mes;    mes.information(0,QString(tr("Aviso")),QString(tr("Debe seleccionar una fila\n de la tabla")));
    }
}

```

```

void calendar:: AddExtra(){
    if(addWin){
        insertWindow = new InsertDate(ui->LabTable,laborables);
        insertWindow->show();
        addWin = false;
    }else{
        if(!insertWindow->isVisible()){
            insertWindow = new InsertDate(ui->LabTable,laborables);
            insertWindow->show();
        }
    }
}
void calendar:: DeleteExtra(){
    int row2delete = ui->LabTable->currentRow();
    if(row2delete>=0){
        ui->LabTable->removeRow(row2delete);
        if(festivos->value(row2delete).dayOfWeek()==Qt::Saturday || festivos->value(row2delete).dayOfWeek()==Qt::Sunday)
        {
            ui->MyCal->setDateTextFormat(festivos->value(row2delete),ui->MyCal->dateTextFormat(QDate(2012,10,26)));
        }else ui->MyCal->setDateTextFormat(festivos->value(row2delete), ui->MyCal->dateTextFormat(QDate(2012,10,27)));
        laborables->remove(row2delete);
        ui->LabTable->setCurrentCell(-1,-1);
    }else{
        QMessageBox mes;
        mes.information(0,QString(tr("Aviso")),QString(tr("Debe seleccionar una fila\n de la tabla")));
    }
}

```

7.1.5 GenerateAnalysis.h

```

#ifndef GENERATEANALYSIS_H
#define GENERATEANALYSIS_H

#include <QWidget>
#include <QGroupBox>
#include <QGridLayout>
#include <QRadioButton>
#include <QCheckBox>
#include <QLineEdit>
#include <QLabel>
#include <QSpinBox>
#include <QDoubleSpinBox>
#include <QPushButton>
#include <QDate>
#include <QComboBox>
#include <QMessageBox>
#include "Project.h"
class QGroupBox;
class GenerateAnalysis : public QWidget
{
    Q_OBJECT
public:
    explicit GenerateAnalysis(Project *proyec,QVector<QDate> *workDates, QWidget *parent=0);
    ~GenerateAnalysis();
private slots:
    void changeOptions();
    void refreshFlex(int index);
    void submit();

private:
    //Functions
    void buildFlex(); void buildVar(); void buildProb(); void joinGroups();
    Project *pro; Activity *act;
    QVector<QDate> *datesVec;

    QHBoxLayout *hGroups; QGroupBox *groupFlex,*groupVar, *groupProb; QPushButton *submitAnalysis,*cancelAnalysis;
    //Flex
    QComboBox *actCombo; QLineEdit *hlrLine, *hlaLine; QRadioButton *forRadio, *backRadio; QSpinBox *forSpinner, *backSpinner;
    //Var
    QDoubleSpinBox *realCost,*varian; QSpinBox *realDur;
    //Prob
    QGroupBox *prob1, *prob2, *prob3; QRadioButton *radioProb1, *radioProb2, *radioProb3; QDoubleSpinBox *probProb;
    QSpinBox *durProb, *int1Prob, *int2Prob;
};

#endif // GENERATEANALYSIS_H

```

7.1.6 GenerateAnalysis.cpp

```
#include "GenerateAnalysis.h"

GenerateAnalysis::GenerateAnalysis(Project *proyec,QVector<QDate> *workDates, QWidget *parent)
:QWidget(parent)
{
    //Linking
    this->pro=proyec; this->setWindowTitle(QString("Análisis"));
    this->setWindowIcon(QIcon("./icons/management.png")); this->setWindowIconText(QString(tr("Project Manager")));
    this->datesVec=workDates;
    buildFlex(); buildProb();
    hGroups = new QBoxLayout();
    hGroups->addWidget(groupFlex); hGroups->addWidget(groupProb);
    joinGroups();
}
void GenerateAnalysis:: joinGroups(){
    //BUTTONS
    submitAnalysis = new QPushButton(QString("Analizar"));
    submitAnalysis->setFixedSize(120,60);
    cancelAnalysis = new QPushButton(QString("Cerrar"));
    cancelAnalysis->setFixedSize(120,60);
    connect(submitAnalysis,SIGNAL(clicked()),this,SLOT(submit()));
    connect(cancelAnalysis,SIGNAL(clicked()),this,SLOT(close()));
    QBoxLayout *hbut = new QBoxLayout();
    hbut->addWidget(cancelAnalysis); hbut->addWidget(submitAnalysis);
    QVBoxLayout *vbox = new QVBoxLayout();
    vbox->addLayout(hGroups); vbox->addLayout(hbut);
    this->setLayout(vbox);
}
GenerateAnalysis::~GenerateAnalysis()
{
    delete this;
}
//Build WINDOWS
void GenerateAnalysis:: refreshFlex(int index)
{
    act=pro->getActivities().at(index); hlrLine->setText(QString::number(act->getHLR())); hlaLine->setText(QString::number(act->getHLA()));
}
void GenerateAnalysis:: buildFlex()
{
    groupFlex = new QGroupBox(QString("Flexibilidad Actividad")); groupFlex->setCheckable(true); groupFlex->setChecked(false);
    actCombo = new QComboBox();
    for(int i=0;i<pro->sizeActivities();i++)
        actCombo->addItem(QString::fromStdString(pro->getActivities().at(i)->getName()));
    connect(actCombo,SIGNAL(currentIndexChanged(int)),this,SLOT(refreshFlex(int)));
    QBoxLayout *hCombo = new QBoxLayout();
    QLabel *actLabel = new QLabel(QString("Actividad: ")); hCombo->addWidget(actLabel); hCombo->addWidget(actCombo);
    act=pro->getActivities().at(0);
    QLabel *hlrlab = new QLabel(QString("HLR: "));
    hlrLine = new QLineEdit();
    hlrLine->setText(QString::number(act->getHLR())); hlrLine->setReadOnly(true);
    QBoxLayout *hlrlBox = new QBoxLayout();
    hlrlBox->addWidget(hlrlab); hlrlBox->addWidget(hlrLine);
    QLabel *hlaLab = new QLabel(QString("HLA: "));
    hlaLine = new QLineEdit();
    hlaLine->setText(QString::number(act->getHLA())); hlaLine->setReadOnly(true);
    QBoxLayout *hlaBox = new QBoxLayout();
    hlaBox->addWidget(hlaLab); hlaBox->addWidget(hlaLine);
    backRadio = new QRadioButton();
    backRadio->setText(QString("Atrasar")); backRadio->setFixedWidth(120);
    forRadio = new QRadioButton();
    forRadio->setText(QString("Adelantar")); forRadio->setFixedWidth(120);
    connect(backRadio,SIGNAL(clicked()),this,SLOT(changeOptions())); connect(forRadio,SIGNAL(clicked()),this,SLOT(changeOptions()));
    forRadio->setChecked(true);
    backSpinner = new QSpinBox();
    backSpinner->setFixedWidth(80); backSpinner->setAlignment(Qt::AlignLeft); backSpinner->setMinimum(0);
    backSpinner->setSingleStep(1); backSpinner->setMaximum(999); backSpinner->setEnabled(false);
    forSpinner = new QSpinBox();
    forSpinner->setFixedWidth(80); forSpinner->setAlignment(Qt::AlignLeft); forSpinner->setMinimum(0); forSpinner->setSingleStep(1);
    forSpinner->setMaximum(999); forSpinner->setEnabled(true);
}
//TOP
QVBoxLayout *top = new QVBoxLayout();
top->addLayout(hCombo); top->addLayout(hlrlBox); top->addLayout(hlaBox);
//BOTTOM
QHBoxLayout *hbox1 = new QHBoxLayout();
hbox1->addWidget(forRadio); hbox1->addWidget(forSpinner);
QHBoxLayout *hbox2 = new QHBoxLayout();
hbox2->addWidget(backRadio); hbox2->addWidget(backSpinner);
QVBoxLayout *vbox = new QVBoxLayout();
vbox->addLayout(top); vbox->addLayout(hbox1); vbox->addLayout(hbox2); groupFlex->setLayout(vbox); groupFlex->resize(200,200);
}
```

```

void GenerateAnalysis::buildVar(){
    groupVar = new QGroupBox(QString("Variación"));
    groupVar->setCheckable(true); groupVar->setChecked(false);
    realDur = new QSpinBox();
    realDur->setMinimum(0); realDur->setSingleStep(1);    realDur->setMaximum(99999999); realDur->setFixedWidth(80);
    realCost = new QDoubleSpinBox();
    realCost->setMinimum(0.00); realCost->setSingleStep(0.05);    realCost->setMaximum(99999999); realCost->setFixedWidth(80);
    QHBoxLayout *hdur = new QHBoxLayout;
    QLabel *durlab = new QLabel("Duración Real");
    hdur->addWidget(durlab);    hdur->addWidget(realDur);
    QHBoxLayout *hcost = new QHBoxLayout;
    QLabel *costLab = new QLabel("Coste Real");
    hcost->addWidget(costLab);    hcost->addWidget(realCost);
    QVBoxLayout *vbox = new QVBoxLayout;
    vbox->addLayout(hdur);    vbox->addLayout(hcost);
    groupVar->setLayout(vbox);
}
void GenerateAnalysis::buildProb(){

    groupProb = new QGroupBox(QString("Probabilidad"));  groupProb->setCheckable(true); groupProb->setChecked(false);
    radioProb1 = new QRadioButton();  radioProb1->setText(QString("Probabilidad -> Días"));
    radioProb2 = new QRadioButton();  radioProb2->setText(QString("Días -> Probabilidad"));
    radioProb3 = new QRadioButton();  radioProb3->setText(QString("Intervalo"));    radioProb1->setChecked(true);
    varian = new QDoubleSpinBox();
    QLabel *varLab = new QLabel(QString("Varianza: "));
    varian->setMaximum(999999); varian->setMinimum(0.00);    varian->setDecimals(2); varian->setSingleStep(0.5);
    QHBoxLayout *hVar = new QHBoxLayout;
    hVar->addWidget(varLab);    hVar->addWidget(varian);
    probProb = new QDoubleSpinBox(); probProb->setMaximum(99.5);
    probProb->setMinimum(0.00); probProb->setDecimals(2);
    probProb->setSingleStep(0.5);
    QVBoxLayout *vbox1 = new QVBoxLayout;
    vbox1->addWidget(radioProb1);    vbox1->addWidget(probProb);
    durProb = new QSpinBox();  durProb->setMaximum(9999);
    durProb->setMinimum(0); durProb->setSingleStep(1);
    QVBoxLayout *vbox2 = new QVBoxLayout;
    vbox2->addWidget(radioProb2);    vbox2->addWidget(durProb);

    int1Prob = new QSpinBox(); int1Prob->setMaximum(9999); int1Prob->setMinimum(0); int1Prob->setSingleStep(1);
    int2Prob = new QSpinBox(); int2Prob->setMaximum(9999); int2Prob->setMinimum(0); int2Prob->setSingleStep(1);
    //int1Prob->setPrefix(QString(" "));int2Prob->setPrefix(QString(" ")); int2Prob->setSuffix(QString(" "));
    QHBoxLayout *hbox3 = new QHBoxLayout;
    hbox3->addWidget(int1Prob);    hbox3->addWidget(int2Prob);
    QVBoxLayout *vbox3 = new QVBoxLayout;
    vbox3->addWidget(radioProb3);    vbox3->addLayout(hbox3);

    QVBoxLayout *vbox = new QVBoxLayout;
    vbox->addLayout(hVar);    vbox->addLayout(vbox1);
    vbox->addLayout(vbox2);    vbox->addLayout(vbox3);
    groupProb->setLayout(vbox);
}

//PRIVATE SLOTS
void GenerateAnalysis::changeOptions(){
    if(forRadio->isChecked()){
        forSpinner->setEnabled(true);    backSpinner->setEnabled(false);
    }else{
        backSpinner->setEnabled(true);    forSpinner->setEnabled(false);
    }
}
void GenerateAnalysis::submit(){
    QMessageBox *mes = new QMessageBox();
    mes->setWindowTitle(QString("Análisis"));  QString analisisMes;
    if(!groupFlex->isChecked() && !groupProb->isChecked()){
        analisisMes.append(QString("No ha sido marcaado ningún tipo de análisis"));
    }

    if(groupFlex->isChecked()){
        //act = pro->getActivity(actCombo->currentText().toStdString());
        QString flexMes("Flexibilidad Actividad "); flexMes.append(QString::fromStdString(act->getName()));
        flexMes.append(QString(" "));
        if(forRadio->isChecked()){
            flexMes.append(QString::fromStdString(pro->analizeAnticipateTMin(forSpinner->value(),act->getName())));
        }else{
            flexMes.append(QString::fromStdString(pro->analizeDelayTMin(backSpinner->value(),act->getName())));
        }
        analisisMes+=flexMes;
    }
    int num=0;
    if(groupProb->isChecked()){

}

```

```

        if(radioProb1->isChecked()){
            QString probMes("\nDuracion Proyecto:");
            num=pro->getDurationByProbability(probProb->value(),varian->value());
            QString proDate("");
            proDate.append(QString::number(num));
            if(pro->getEnd()->getTMax()>=num){
                proDate.append(" Fecha Fin:");
                proDate.append(datesVec->at(num-1).toString("dd/MM/yyyy"));
            }
            probMes.append(proDate);
            analisisMes+=probMes;
        }
        if(radioProb2->isChecked()){
            QString probMes("\nProbabilidad Proyecto:");
            probMes.append(QString::number(pro->getProbabDurationLessThan(durProb->value(),varian->value())));
            analisisMes+=probMes;
        }
        if(radioProb3->isChecked()){
            QString probMes("\nProbabilidad Proyecto:");
            probMes.append(QString::number(pro->getProbabDurationBetween(int1Prob->value(),int2Prob->value(),varian->value())));
            analisisMes+=probMes;
        }
    }
    mes->information(0,QString("Análisis"),analisisMes);
}

```

7.1.7 GenerateReport.h

```

#ifndef WINDOW_H
#define WINDOW_H
#include <QWidget>
#include <QGroupBox>
#include <QGridLayout>
#include <QRadioButton>
#include <QCheckBox>
#include <QLineEdit>
#include <QLabel>
#include <QSpinBox>
#include <QDoubleSpinBox>
#include <QPushButton>
#include <QDate>
#include <QComboBox>
#include <QMessageBox>
#include "Project.h"
#include "Rules.h"
#include <time.h>
#include <Report.h>
#include <QDir>
#include <QFileDialog>

class QGroupBox;
class Window : public QWidget
{
    Q_OBJECT
public:
    Window(Project *proyecto, QVector<QDate> *festivo, QVector<QDate> *laborable, QString *weekEndC, QWidget *parent);
    ~Window();
    void updateProject(Project *proyecto){ this->pro=proyecto; }
    void updateDates(QVector<QDate> *festivo, QVector<QDate> *laborable){
        this->fest=festivo;    this->labo=laborable;
    }
    void clear(){
        this->workDays->clear();    workDays = new QVector<QDate>(0);
    }

    bool OptionsSaved;
    QVector<QDate> *workDays;  QPushButton *save, *submit, *cancel;
protected:
    void closeEvent(QCloseEvent * event){
        if(repGen) rep->close();
        this->close();
    }
public slots:
    void generate();
private slots:
    void disableOptions();
    void getDebugDir();
}

```

```

private:
    void createStartDay();    void createMaxOverrun();   void createResourceMan();   void createNivRec();   void createBackForward();
    void getWorkDays();      bool dayValid(QDate date);  void createAndConnect();

    QVector<QDate> *fest;  QVector<QDate> *labo;   QDate *startDate;   QString *weekEnd;
    Report *rep;   Project *pro;
    bool repGen;
    //Main window
    QGridLayout *grid;  QCheckBox *modeDebug;  QString debPath;   QPushButton *chooseDebDir; QLineEdit *linePath;
    //Groups and stuff
    QGroupBox *StartDay;  QGroupBox *CalcMinCos;  QGroupBox *LimAtAd;  QGroupBox *Group1, *Group2, *Group3, *Group4;
    QGroupBox *RetraAdel;  QGroupBox *NivRec;
    //Date
    QLabel *date_iniLa;  QHBoxLayout *hboxDate;  QSpinBox *date_dd;  QSpinBox *date_mm;  QSpinBox *date_yyyy;
    //Minimo Coste
    QDoubleSpinBox *maxOverrun;
    //Serie
    QRadioButton *ser;   QRadioButton *par;   QRadioButton *multi;  QComboBox *priorityRules;
    //Atraso-Adelanto
    QRadioButton *ser1, *ser2;  QRadioButton *par1, *par2;  QComboBox *priorityRules1, *priorityRules2;  QLabel *priorityLab1, *priorityLab2;
    //Nivelacion
    bool mesOnce;
    QDoubleSpinBox *incAllowed;  QGroupBox *radioRes;  QRadioButton *allRes, *chooseRes;  QGroupBox *listGroup;  QListWidget *resList;
};

#endif // WINDOW_H

```

7.1.8 GenerateReport.cpp

```

#include "GenerateReport.h"
#include <stdio.h>
Window::Window(Project *proyecto, QVector<QDate> *festivo, QVector<QDate> *laborable,QString *weekEndC, QWidget *parent)
:QWidget(parent)
{
    //Linking
    this->pro=proyecto;  this->fest=festivo;  this->labo=laborable;  this->weekEnd=weekEndC;
    this->setWindowIcon(QIcon("./icons/management.png"));  this->setWindowTitle(QString(tr("Project Manager")));
    mesOnce=false;  repGen=false;
    createAndConnect();
}
Window::~Window()
{
    delete this;
}
//Building the window
void Window::createAndConnect()
{
    //Generate windows and stuff
    workDays = new QVector<QDate>(0);  grid = new QGridLayout;

    modeDebug = new QCheckBox(QString("Modo Debug"));
    modeDebug->setChecked(false);
    chooseDebDir = new QPushButton(QString("Elegir directorio"));
    chooseDebDir->setIcon(QPixmap("./icons/open_file.png")); //chooseDebDir->setIconSize(QSize(15,15));
    linePath = new QLineEdit();
    linePath->setReadOnly(true);
    submit = new QPushButton();
    submit->setFixedSize(100,108); submit->setFlat(true);
    submit->setIcon(QPixmap("./icons/generate.png"));  submit->setIconSize(QSize(80,80));

    save = new QPushButton();
    save->setFixedSize(100,50); save->setFlat(true);  save->setIcon(QPixmap("./icons/save_changes.png")); save->setIconSize(QSize(50,50));
    cancel = new QPushButton();
    cancel->setFixedSize(100,50);  cancel->setFlat(true);
    cancel->setIcon(QPixmap("./icons/cancel_changes.png")); cancel->setIconSize(QSize(50,50));
    QHBoxLayout *buts1 = new QHBoxLayout;
    buts1->addWidget(cancel);  buts1->addWidget(save);  buts1->addWidget(submit);
    QVBoxLayout *vDeb = new QVBoxLayout;
    QHBoxLayout *hDeb = new QHBoxLayout;
    hDeb->addWidget(modeDebug);  hDeb->addWidget(chooseDebDir);  vDeb->addLayout(hDeb);  vDeb->addWidget(linePath);
    QVBoxLayout *buts2 = new QVBoxLayout;
    buts2->addLayout(vDeb);  buts2->addLayout(buts1);

    Group1 = new QGroupBox();  Group2 = new QGroupBox();  Group3 = new QGroupBox();  Group4 = new QGroupBox();
    radioRes = new QGroupBox(); listGroup = new QGroupBox();

    //Creating window groups
    createStartDay();  createMaxOverrun();  createResourceMan();  createNivRec();  createBackForward();
    //Composing the window
    grid->addWidget(StartDay,0,0);  grid->addWidget(CalcMinCos,0,1);  grid->addWidget(LimAtAd,1,0);  grid->addWidget(RetraAdel,1,1);
    grid->addWidget(NivRec,2,0);  grid->addLayout(buts2,2,1);  setLayout(grid);  setWindowTitle(tr("Generar Informe"));  resize(450,200);
}

```

```

connect(submit,SIGNAL(clicked()),this,SLOT(generate())); connect(save,SIGNAL(clicked()),this,SLOT(close()));
connect(cancel,SIGNAL(clicked()),this,SLOT(close())); connect(chooseDebDir,SIGNAL(clicked()),this,SLOT(getDebugDir()));
}
void Window::getDebugDir()
{
    QDir directory;
    debPath = QFileDialog::getExistingDirectory(this,tr("Directorio"),directory.path());
    if(!debPath.isNull()) linePath->setText(debPath);
}

void Window::createStartDay()
{
    StartDay = new QGroupBox(tr("Dia deseado inicio"));

    date_iniLa = new QLabel(tr("Fecha Inicio:"));
    date_dd = new QSpinBox;
    date_dd->setMaximum(31);date_dd->setMinimum(1);    date_dd->setMaximumWidth(50);
    date_mm = new QSpinBox;
    date_mm->setMaximum(12);date_mm->setMinimum(1);    date_mm->setMaximumWidth(50);
    date_yyyy = new QSpinBox;
    date_yyyy->setMinimum(2012); date_yyyy->setMaximumWidth(110);    date_yyyy->setMaximum(2100);
//Let's place the predefined date to the actual day just by doing the following:
    date_dd->setValue(QDate::currentDate().day());    date_mm->setValue(QDate::currentDate().month());
    date_yyyy->setValue(QDate::currentDate().year());

    hboxDate = new QHBoxLayout;    hboxDate->addWidget(date_iniLa);
    hboxDate->addWidget(date_dd); hboxDate->addWidget(date_mm);    hboxDate->addWidget(date_yyyy);
    hboxDate->setAlignment(Qt::AlignLeft);    StartDay->setLayout(hboxDate);
}

void Window::createMaxOverrun()
{
    CalcMinCos = new QGroupBox(tr("Calcular Minimo Coste ").append(QChar(8364)));
    CalcMinCos->setCheckable(true);    CalcMinCos->setChecked(false);
    connect(CalcMinCos,SIGNAL(clicked()),this,SLOT(disableOptions()));
    maxOverrun = new QDoubleSpinBox(); maxOverrun->setMaximumWidth(120);
    maxOverrun->setMinimum(0.00); maxOverrun->setSingleStep(0.05);
    maxOverrun->setMaximum(99999999.99);// maxOverrun->setSuffix(QChar(8364)); //QChar(8364) = €
    QHBoxLayout *hbox = new QHBoxLayout;
    QLabel *lab = new QLabel("Max. Sobrecoste: ");
    hbox->addWidget(lab);    hbox->addWidget(maxOverrun);    hbox->setAlignment(Qt::AlignLeft);
    CalcMinCos->setLayout(hbox);
}

void Window::createNivRec()
{
    NivRec = new QGroupBox(tr("Nivelacion Recursos"));    NivRec->setCheckable(true);    NivRec->setChecked(false);

    incAllowed = new QDoubleSpinBox();
    incAllowed->setMaximum(100);    incAllowed->setMinimum(0);    incAllowed->setSingleStep(0.05);    incAllowed->setFixedWidth(100);
    //incAllowed->setSuffix(QString("%"));

    allRes = new QRadioButton(tr("Todos"));
    chooseRes = new QRadioButton(tr("Elige"));
    allRes->setChecked(true);
    connect(chooseRes,SIGNAL(toggled(bool)),this,SLOT(disableOptions()));

    resList = new QListWidget();
    resList->setEnabled(false); vector<Resource*> *vecRes = pro->getResources();
    int nvecRes = vecRes->size();
    for(int i=0;i<nvecRes;i++){
        string name = vecRes->at(i)->getName();
        QListWidgetItem *checkbox1 = new QListWidgetItem(QString::fromStdString(name));
        resList->addItem(checkbox1);
        resList->item(i)->setFlags(resList->item(i)->flags() | Qt::ItemIsUserCheckable);
        resList->item(i)->setCheckState(Qt::Unchecked);
    }
    resList->setFixedSize(200,120);

    QHBoxLayout *hbox1 = new QHBoxLayout;
    hbox1->addWidget(allRes);hbox1->addWidget(chooseRes);    hbox1->setAlignment(Qt::AlignLeft);
    QVBoxLayout *vbox1 = new QVBoxLayout;
    vbox1->addLayout(hbox1);vbox1->addWidget(resList);    vbox1->setAlignment(Qt::AlignLeft);
    radioRes->setLayout(vbox1);    radioRes->setTitle("Aplicar a:");
    QVBoxLayout *vbox = new QVBoxLayout;
    vbox->addWidget(radioRes);    NivRec->setLayout(vbox);
}

void Window::createBackForward()
{
    RetraAdel = new QGroupBox(tr("Retraso-Adelanto"));
    RetraAdel->setCheckable(true);    RetraAdel->setChecked(false);    RetraAdel->setEnabled(false);
    connect(RetraAdel,SIGNAL(clicked()),this,SLOT(disableOptions()));
    ser1 = new QRadioButton(tr("En serie"));//tr("En serie")
}

```

```

par1 = new QRadioButton(tr("En paralelo"));//tr("En paralelo")
ser2 = new QRadioButton(tr("En serie")); par2 = new QRadioButton(tr("En paralelo"));
ser1->setChecked(true); ser2->setChecked(true);
QVBoxLayout *vbox1 = new QVBoxLayout;
vbox1->addWidget(ser1); vbox1->addWidget(par1);
QVBoxLayout *vbox2 = new QVBoxLayout;
vbox2->addWidget(ser2); vbox2->addWidget(par2);
QHBoxLayout *hbox1 = new QHBoxLayout;
Group3->setLayout(vbox1); Group3->setTitle(tr("Adelanto")); Group4->setLayout(vbox2); Group4->setTitle(tr("Atraso"));
hbox1->addWidget(Group3); hbox1->addWidget(Group4);
RetraAdel->setLayout(hbox1);
}
void Window::createResourceMan()
{
    LimAtAd = new QGroupBox(tr("Limitar Recursos")); LimAtAd->setCheckable(true); LimAtAd->setChecked(false);
    connect(LimAtAd,SIGNAL(clicked()),this,SLOT(disableOptions()));
    ser = new QRadioButton(tr("En serie")); par = new QRadioButton(tr("En paralelo")); multi= new QRadioButton(tr("Multipasada"));
    connect(multi,SIGNAL(toggled(bool)),this,SLOT(disableOptions()));
    ser->setChecked(true);
    priorityRules = new QComboBox;
    priorityRules->addItem("LFT"); priorityRules->addItem("EFT"); priorityRules->addItem("EST");
    priorityRules->addItem("LST"); priorityRules->addItem("HT"); priorityRules->addItem("HL");
    priorityRules->addItem("DUR"); priorityRules->addItem("NSUC"); priorityRules->addItem("DEM");

    QVBoxLayout *vbox1 = new QVBoxLayout;
    vbox1->addWidget(ser); vbox1->addWidget(par); vbox1->addWidget(multi);
    QVBoxLayout *vbox2 = new QVBoxLayout;
    vbox2->addWidget(priorityRules);
    QHBoxLayout *hbox1 = new QHBoxLayout;
    Group1->setLayout(vbox1); Group1->setTitle(tr("Esquema")); Group2->setLayout(vbox2); Group2->setTitle(tr("Regla prioridad"));
    hbox1->addWidget(Group1); hbox1->addWidget(Group2);
    LimAtAd->setLayout(hbox1);
}
//Options & Generate
void Window::disableOptions(){
    if(CalcMinCos->isChecked()&&(!mesOnce)){
        QMessageBox mes;
        mes.information(0,QString(tr("Aviso")),QString(tr("Marcando esta opcion la duracion de\n"
            "las actividades puede verse modificada")));
        mes.show(); mesOnce=true;
    }
    if(LimAtAd->isChecked()){
        RetraAdel->setEnabled(true);
    }else{
        RetraAdel->setEnabled(false); RetraAdel->setChecked(false);
    }
    if(multi->isChecked()) Group2->setEnabled(false);
    else if(LimAtAd->isChecked()) Group2->setEnabled(true);
    if(chooseRes->isChecked()) resList->setEnabled(true);
    else resList->setEnabled(false);
}
bool Window::dayValid(QDate date)
{
    //If it's a work day just return TRUE
    for(int i=0;i<labo->size();i++) if(date.toString("dd/MM/yyyy")==labo->value(i).toString("dd/MM/yyyy")) return true;
    //Check in case it's a weekend.
    if(weekEnd!=QString("NONE")){
        if(date.dayOfWeek()==6){
            if(weekEnd==QString("BOTH") || weekEnd==QString("SAT")) return true;
            else return false;
        }
        if(date.dayOfWeek()==7){
            if(weekEnd==QString("BOTH") || weekEnd==QString("SUN")) return true;
            else return false;
        }
    }else{
        if(date.dayOfWeek()==6 || date.dayOfWeek()==7) return false;
    }
    //Finally check if it's a holiday when there's no work.
    //Check Nacionales.
    int yyyy=date.year();
    for(int i=0;j<2;i++)//calcula para 2 años
        if(date.toString("dd/MM/yyyy")==QString("01/01/").append(QString::number(yyyy))) return false;
        if(date.toString("dd/MM/yyyy")==QString("06/01/").append(QString::number(yyyy))) return false;
        if(date.toString("dd/MM/yyyy")==QString("19/03/").append(QString::number(yyyy))) return false;
        if(date.toString("dd/MM/yyyy")==QString("01/05/").append(QString::number(yyyy))) return false;
        if(date.toString("dd/MM/yyyy")==QString("15/08/").append(QString::number(yyyy))) return false;
        if(date.toString("dd/MM/yyyy")==QString("12/10/").append(QString::number(yyyy))) return false;
        if(date.toString("dd/MM/yyyy")==QString("01/11/").append(QString::number(yyyy))) return false;
        if(date.toString("dd/MM/yyyy")==QString("06/12/").append(QString::number(yyyy))) return false;
        if(date.toString("dd/MM/yyyy")==QString("08/12/").append(QString::number(yyyy))) return false;
}

```

```

if(date.toString("dd/MM/yyyy") == QString("25/12/").append(QString::number(yyyy))) return false;
    yyyy+=1;
}
//Check additional Holidays.
for(int i=0;i<fest->size();i++) if(date==fest->value(i)) return false;
//and so, if there's no other choice we return true because it's a normal boring day...
return true;
}
void Window::getWorkDays(){
    int auxDura = pro->getEnd()->getTMin(); int i=0; workDays->clear();
    while(!dayValid(startingDate->addDays(i))){
        i++;
    }
    workDays->append(startingDate->addDays(i)); auxDura=auxDura-1; int j=1;
    for(int i=0;i<auxDura;i++){
        j=1;
        while(!dayValid(workDays->at(i).addDays(j))){
            j++;
        }
        workDays->append(workDays->at(i).addDays(j));
    }
}
void Window::generate(){
    bool valido; mesOnce=false; QMessageBox mes; bool showIt=false;
    clear(); valido = QDate::isValid(date_yyyy->text().toInt(),date_mm->text().toInt(),date_dd->text().toInt());
    if(!valido){
        mes.warning(0,QString(tr("Error")),QString(tr("Fecha no valida")));
    }else if(modeDebug->isChecked() && debPath.isNull()){
        mes.warning(0,QString(tr("Error")),QString(tr("No se ha seleccionado ningn directorio\nDesmarque la casilla o seleccione uno")));
    }else{
        startingDate = new QDate(date_yyyy->text().toInt(),date_mm->text().toInt(),date_dd->text().toInt());
        pro->setReportDebug(debPath.toStdString().append("\\"));
        if(modeDebug->isChecked()) pro->enableDebug();
        //Camino crítico
        pro->startDebug("criticalPath");
        pro->calculCriticalPath(); //Realiza el algoritmo del camino crítico
        pro->finishDebug(); getWorkDays(); rep = new Report(workDays,pro,0);
        rep->setWindowTitle(QString("Informe Camino Crítico")); repGen=true; rep->show();
        //Mínimo Coste
        if(CalcMinCos->isChecked()){
            showIt=false; pro->startDebug("minCostMinDuration");
            if(maxOverrun->value(>0){ pro->calculPathMinTimeMinCost(maxOverrun->value());
            }else{ pro->calculPathMinTimeMinCost(); }
            pro->finishDebug(); getWorkDays();
            rep = new Report(workDays,pro,0); rep->setWindowTitle(QString("Informe Mínimo Coste"));
            rep->show();
        }
        //Limitación
        if(LimAtAd->isChecked()){
            showIt=false;
            int esquema=0;
            if(ser->isChecked()) esquema=_SERIE;
            if(par->isChecked()) esquema=_PARALEL;
            if(multi->isChecked()) esquema=_MULTIPASADA;

            string limName = priorityRules->currentText().toStdString();
            int reglaPrio=0;
            if(limName=="LFT") reglaPrio=MIN_LFT_FIFO;
            if(limName=="EFT") reglaPrio=MIN_EFT_FIFO;
            if(limName=="EST") reglaPrio=MIN_EST_FIFO;
            if(limName=="LST") reglaPrio=MIN_LST_FIFO;
            if(limName=="HT") reglaPrio=MIN_HT_FIFO;
            if(limName=="HL") reglaPrio=MIN_HL_FIFO;
            if(limName=="DUR") reglaPrio=MIN_DUR_FIFO;
            if(limName=="NSUC") reglaPrio=MAX_NSUC_FIFO;
            if(limName=="DEM") reglaPrio=MAX_DEM_FIFO;

            pro->startDebug("limitation");
            if(pro->limitarResources(reglaPrio,esquema)) showIt=true;
            pro->finishDebug();
            if(showIt){
                getWorkDays(); rep = new Report(workDays,pro,0); rep->setWindowTitle(QString("Informe Limitación de Recursos"));
                rep->show();
            }
        }
        //Atraso-Adelanto
        if(RetraAdel->isChecked()&& showIt){
            showIt=false; int esquema1=0; int esquema2=0;
            if(ser1->isChecked()) esquema1=_SERIE;
            if(par1->isChecked()) esquema1=_PARALEL;
            if(ser2->isChecked()) esquema2=_SERIE;
            if(par2->isChecked()) esquema2=_PARALEL;
        }
    }
}

```

```

pro->startDebug("compaction");
if(pro->retrasoAdelantoActivities(esquema2,esquema1))           showIt=true;
pro->finishDebug();
if(showIt){
    getWorkDays(); rep = new Report(workDays,pro,0); rep->setWindowTitle(QString("Informe Atraso-Adelanto")); rep->show();
}
}

//Nivelacion Recursos
if(NivRec->isChecked()&& showIt){
    showIt=false; pro->startDebug("leveling");
    if(allRes->isChecked()){
        if(pro->levelAllResources()) showIt=true;
    }else{
        vector<string> vecResNiv;      int sizeList = resList->height();
        for(int i=0;i<sizeList;i++){
            if(resList->item(i)->checkState()>0) vecResNiv.push_back(resList->item(i)->text().toStdString());
        }
        if(pro->levelResources(vecResNiv)) showIt=true;
    }
    pro->finishDebug();
    if(showIt){
        getWorkDays(); rep = new Report(workDays,pro,0); rep->setWindowTitle(QString("Informe Nivelación de Recursos"));
        rep->show();
    }
    //finestra
}
if(modeDebug->isChecked())   pro->disableDebug();   this->hide();
}
}

```

7.1.9 InsertDate.h

```

#ifndef INSERTDATE_H
#define INSERTDATE_H

#include <QWidget>
#include <QLabel>
#include <QDate>
#include <QLayout>
#include <QSpinBox>
#include <QTextEdit>
#include <QPushButton>
#include <QTableWidget>
#include <QMessageBox>

class InsertDate : public QWidget
{
    Q_OBJECT
public:
    explicit InsertDate(QTableWidget *table, QVector<QDate> *vec_dates, QWidget *parent = 0);
    ~InsertDate();
    void BuildWindow();
public slots:
    void SaveDate();
    bool CheckRepeated(QDate *Date);

private:
    QVector<QDate> *dates;
    QTableWidget *tabla;

    QHBoxLayout *HBox;
    QSpinBox *dd; QSpinBox *mm; QSpinBox *yyyy;
    QPushButton *Guardar; QPushButton *Cancelar;
};

#endif // INSERTDATE_H

```

7.1.10 InsertDate.cpp

```
#include "InsertDate.h"
#include <stdio.h>

InsertDate::InsertDate(QTableWidget *table, QVector<QDate> *vec_dates, QWidget *parent)
:QWidget(parent)
{
    this->dates=vec_dates;  this->tabla=table;
    this->setWindowIcon(QIcon("./icons/management.png"));  this->setWindowIconText(QString(tr("Project Manager")));

    BuildWindow();
    this->setWindowTitle(tr("Nueva Fecha"));
    connect(Guardar,SIGNAL(clicked()),this,SLOT(SaveDate()));  connect(Cancelar,SIGNAL(clicked()),this,SLOT(close()));

}
InsertDate::~InsertDate()
{
    delete this;
}
bool InsertDate::CheckRepeated(QDate *date)
{
    for(int i=0;i<dates->size();i++)
        if(dates->value(i)==*date)  return true;
    return false;
}

void InsertDate::SaveDate(){
    int aux = tabla->rowCount();  QMessageBox mes;
    bool valido = QDate::isValid(yyyy->value(),mm->value(),dd->value());
    if(!valido){
        mes.warning(0,QString("Error"),QString("La fecha no es valida"));
    }else{
        QDate *dateis = new QDate(yyyy->value(),mm->value(),dd->value());
        if(!CheckRepeated(dateis)){
            tabla->setRowCount(aux+1);      dates->append(*dateis);
            tabla->setItem(aux,0,new QTableWidgetItem(dates->last().toString("dd/MM/yyyy")));
            this->close();
        }else  mes.warning(0,QString("Error"),QString("La fecha ya se ha introducido aqui"));
    }
}
void InsertDate::BuildWindow(){
    Guardar = new QPushButton(QString(tr("Guardar")));  Cancelar = new QPushButton(QString(tr("Cancelar")));
    HBox = new QHBoxLayout;
    dd = new QSpinBox;      dd->setMaximum(31);dd->setMinimum(1);      dd->setMaximumWidth(50);
    mm = new QSpinBox;
    mm->setMaximum(12);mm->setMinimum(1);      mm->setMaximumWidth(50);      yyyy = new QSpinBox;
    yyyy->setMinimum(2012);yyyy->setMaximumWidth(110);      yyyy->setMaximum(2100);
    QLabel *text = new QLabel(QString("Fecha: "));
    HBox->addWidget(text);  HBox->addWidget(dd);  HBox->addWidget(mm);  HBox->addWidget(yyyy);
    QHBoxLayout *butaux = new QHBoxLayout;
    butaux->addWidget(Cancelar);  butaux->addWidget(Guardar);
    QVBoxLayout *wind = new QVBoxLayout;
    wind->addItem(HBox);      wind->addItem(butaux);  this->setLayout(wind);
}
```

7.1.11 Report.h

```
#ifndef REPORT_H
#define REPORT_H

#include <QWidget>
#include <QGroupBox>
#include <QGridLayout>
#include <QLabel>
#include <QPushButton>
#include <QDate>
#include <QCalendarWidget>
#include <QListWidget>
#include <QTableWidget>
#include <QRadioButton>
#include <QPainter>
#include <QPen>
#include <QBrush>
#include <QTextFormat>
#include <Project.h>
#include <time.h>
#include <QMessageBox>
#include <qcustomplot.h>
#include <QComboBox>

class QGroupBox;
class Report : public QWidget
{
    Q_OBJECT

public:
    Report(QVector<QDate> *datesVec, Project *proyecto, QWidget *parent);
    ~Report();
    void clear(){
        this->workDays->clear();
    }
    QVector<QDate> *workDays;

protected:
    void closeEvent(QCloseEvent * event){
        if(ganttOpened) ganttWindow->close();
        if(resOpened) customPlot->close();
        this->close();
    }
private slots:
    void showGantt(); //Muestra el gráfico de Gantt
    void showResources(); //Muestra el gráfico de gestión de Recursos
    void zoomGanttIn(){
        if(maxZoomGantt==3) return;
        int h = ganttTable->rowHeight(0);
        for(int i=0;i<ganttTable->rowCount();i++) ganttTable->setRowHeight(i,h*2);
        int w =ganttTable->columnWidth(0);
        for(int i=0;i<ganttTable->columnCount();i++) ganttTable->setColumnWidth(i,w*2);
        maxZoomGantt++;
    }
    void zoomGanttOut(){
        if(maxZoomGantt==0) return;
        int h = ganttTable->rowHeight(0);
        for(int i=0;i<ganttTable->rowCount();i++) ganttTable->setRowHeight(i,h/2);
        int w =ganttTable->columnWidth(0);
        for(int i=0;i<ganttTable->columnCount();i++) ganttTable->setColumnWidth(i,w/2);
        maxZoomGantt--;
    }
    void showCritPath(){ //Muestra el diálogo con distintos caminos críticos.
        QMessageBox *mes = new QMessageBox();
        QString CamCritic(""); int camCrits = pro->getCriticalPaths()->size();
        for(int i=0;i<camCrits;i++){
            CamCritic.append(QString("\nC. Crítico ").append(QString::number(i+1).append(QString(": "))));
            CamCritic.append(QString::fromStdString(pro->getCriticalPaths()->at(i)->toString()));
        }
        mes->setMinimumWidth(200); mes->information(0,QString("C. Crítico"),CamCritic);
    }
    void refreshCCrit(int op){
        int nAct = pro->sizeActivities(); vector<Activity*> vecAct = pro->getActivities();
        for(int i=0;i<nAct;i++){
            if(pro->getCriticalPaths()->at(op)->includesActivity(vecAct.at(i))) paintActivity(i,vecAct.at(i)->getTMin(),vecAct.at(i)->getTNormal(),true);
            else paintActivity(i,vecAct.at(i)->getTMin(),vecAct.at(i)->getTNormal(),false);
        }
        ganttTable->update();
    }
};

#endif
```

```

private:
    void paintActivity(int row, int from, int to, bool crit);
    void saveReport();
    void genCalendar(); //Genera el calendario con las fechas del proyecto marcadas
    void genTableActiv(); //Genera la lista de actividades con sus fechas inicio - fin
    void genGantt(); //Genera el diagrama de Gantt
    void genResources(); //Genera el diagrama de asignación de recursos.
private:
    Project *pro;
    QGridLayout *grid; QPushButton *print; bool ganttOpened, resOpened; QPushButton *showCrit; QComboBox *comboCrit;
    QGroupBox *GroupCal; QCalendarWidget *Cal; QListWidget *callList; QTextCharFormat format; QBrush m_transparentBrush; QBrush brus;
    QPainter painter; QRect rect;
    QGroupBox *GroupList; QTableWidget *actTable; QVector<QLabel*> headerNames;
    QPushButton *OpenGant; QPushButton *OpenRes; //Si se apretan debe cambiar a cerrar ventana.
/*GANTT*/
int maxZoomGantt; QWidget *ganttWindow; QTableWidget *ganttTable; QPushButton *zoomIn, *zoomOut;
/*Resources*/
int maxZoomRes; QCustomPlot *customPlot; QPushButton *zoom2In, *zoom2Out; QGroupBox *GroupResources;
};

#endif // REPORT_H

```

7.1.12 Report.cpp

```

#include "Report.h"
#include <stdio.h>
Report::Report(QVector<QDate> *datesVec, Project *proyecto, QWidget *parent)
: QWidget(parent)
{
    //Linking
    this->workDays=datesVec; this->pro = proyecto;
    this->setWindowIcon(QIcon("./icons/management.png")); this->setWindowIconText(QString(tr("Project Manager")));
    //Creating Windows & stuff
    maxZoomGantt=maxZoomRes=2; //this means it's in state 2, but its value can be among 0 and 4.
    grid = new QGridLayout; print = new QPushButton(tr("Imprimir Informe"));
    GroupCal = new QGroupBox(); GroupList = new QGroupBox(); GroupResources = new QGroupBox();

    genCalendar(); genTableActiv(); genGantt(); genResources();
    resOpened=ganttOpened = false;
    QVBoxLayout *vbox = new QVBoxLayout; vbox->addWidget(GroupList); vbox->addWidget(GroupCal);
    setLayout(vbox); this->setMinimumWidth(550);
    //Listeners
    connect(OpenGant,SIGNAL(clicked()),this,SLOT(showGantt())); connect(OpenRes,SIGNAL(clicked()),this,SLOT(showResources()));
}
Report::~Report(){
    delete this;
}

//PRIVATE SLOTS
void Report::showGantt()
{
    if(!ganttOpened){ ganttOpened=true; ganttWindow->show(); }
    else{ ganttWindow->show(); }
}
void Report::showResources()
{
    QMessageBox mes;
    if(!pro->sizeResources()){

        //QMessageBox mes;
        mes.information(0,QString("No hay recursos"),QString("No puede generarse este informe porque no hay recursos."));
        return;
    }
    else{
        if(!resOpened){ customPlot->show(); resOpened=true; }
        else{ customPlot->show(); }
    }
}
//PUBLIC FUNCTIONS
void Report::genCalendar()
{
    GroupCal = new QGroupBox(tr("Calendario Proyecto"));
    OpenGant = new QPushButton();
    OpenGant->setToolTip(QString("Muestra Diagrama\n de Gantt"));
    OpenGant->setIcon(QPixmap("./icons/gantt.png")); OpenGant->setIconSize(QSize(60,60));
    OpenGant->setFlat(true);
    OpenRes = new QPushButton();
    OpenRes->setToolTip(QString("Muestra Asignacion\n de Recursos"));
    OpenRes->setIcon(QPixmap("./icons/plot_res.png")); OpenRes->setIconSize(QSize(60,60));
    OpenRes->setFlat(true);

    Cal = new QCalendarWidget(); //w:340 h:200
    Cal->setFirstDayOfWeek(Qt::Monday); Cal->setGridVisible(true);
}

```

```

callList = new QListWidget();
int i=-1;
do{
    i++; brus.setColor(Qt::black); format.setForeground(brus); format.setBackground(QColor(25,125,35,127));
    Cal->setTextFormat(workDays->at(i),format); callList->addItem(new QListWidgetItem(workDays->at(i).toString("dd/MM/yyyy")));
}while(workDays->at(i)!=workDays->last());

QHBoxLayout *hbox2 = new QHBoxLayout;
hbox2->addWidget(OpenGant); hbox2->addWidget(OpenRes);
QVBoxLayout *vbox = new QVBoxLayout;
vbox->addWidget(Cal); vbox->addLayout(hbox2);
GroupCal->setLayout(vbox);
}

void Report::genTableActiv()
{
    GroupList = new QGroupBox(tr("Informe Actividades")); actTable = new QTableWidget();

    actTable->setColumnCount(8); actTable->setRowCount(pro->getActivities().size()+2);

    QStringList listNames; listNames<<"Nombre"<<"F. Inicio Temprano"<<"F. Inicio Tardío";
    listNames<<"F. Fin Temprano"<<"F. Fin Tardío"<<"HT"<<"HL"<<"Sucesoras";
    actTable->setHorizontalHeaderLabels(listNames);
    //Análisis básico
    showCrit = new QPushButton(QString("Muestra caminos crícos")); showCrit->setFixedSize(200,40);
    connect(showCrit,SIGNAL(clicked()),this,SLOT(showCritPath()));

    int camCrits = pro->getCriticalPaths()->size(); QString CamCritic("");
    if(camCrits==1) CamCritic.append(QString::fromStdString(pro->getCriticalPaths()->at(0)->toString()));

    QString fechalni("Fecha Inicio: "); fechalni.append(workDays->first().toString("dd/MM/yyyy"));
    QString fechaFin(" Fecha fin: "); fechaFin.append(workDays->last().toString("dd/MM/yyyy"));
    QString fechalniFin; fechalniFin = fechalni+fechaFin;
    QString dura("Duración: ");

    dura.append(QString::number(pro->getEnd()->getTMax()).append(" días laborables"));

    int nAct = pro->sizeActivities(); vector<Activity*> vecAct = pro->getActivities();
    for(int i=0;i<nAct;i++){
        actTable->setItem(i,0,new QTableWidgetItem(QString::fromStdString(vecAct.at(i)->getName())));
        actTable->setItem(i,1,new QTableWidgetItem(workDays->at(vecAct.at(i)->getEarlyStart()).toString("dd/MM/yyyy")));
        actTable->setItem(i,2,new QTableWidgetItem(workDays->at(vecAct.at(i)->getLateStart()).toString("dd/MM/yyyy")));
        actTable->setItem(i,3,new QTableWidgetItem(workDays->at(vecAct.at(i)->getEarlyEnd()-1).toString("dd/MM/yyyy")));
        actTable->setItem(i,4,new QTableWidgetItem(workDays->at(vecAct.at(i)->getLateEnd()-1).toString("dd/MM/yyyy")));
        actTable->setItem(i,5,new QTableWidgetItem(QString::number(vecAct.at(i)->getHTotal())));
        actTable->setItem(i,6,new QTableWidgetItem(QString::number(vecAct.at(i)->getHFree())));
        actTable->setItem(i,7,new QTableWidgetItem(QString::fromStdString(vecAct.at(i)->succesorsToString())));
    }

    QVBoxLayout *vbox1 = new QVBoxLayout;
    if(camCrits==1){ vbox1->addWidget(new QLabel(CamCritic));
    }else{ vbox1->addWidget(showCrit);
        vbox1->addWidget(new QLabel(fechalniFin)); vbox1->addWidget(new QLabel(dura)); QString total("");
        if(pro->getOverrun()>0){ total.append("Sobrecoste:").append(QString::number(pro->getOverrun())).append(" euros ");
        }
        if(pro->getTotalCost()>pro->getOverrun()){
            total.append("Coste Total:").append(QString::number(pro->getTotalCost())).append(" euros");
        }
        if(pro->getOverrun()>0 || pro->getTotalCost()>pro->getOverrun()) vbox1->addWidget(new QLabel(total));
        vbox1->addWidget(actTable);
    }
    GroupList->setLayout(vbox1);
}

void Report::paintActivity(int row, int from, int to, bool crit){
    for(int i=from;i<from+to;i++){
        QTableWidgetItem *item = new QTableWidgetItem;
        ganttTable->setItem(row,i,item);
        if(crit) ganttTable->item(row,i)->setBackgroundColor(QColor(220,0,0,175));
        else ganttTable->item(row,i)->setBackgroundColor(QColor(100,200,215,175));
    }
}

void Report::genGantt()
{
    ganttWindow = new QWidget();
    zoomIn = new QPushButton();
    zoomIn->setIcon(QPixmap("./icons/zoom_in")); zoomIn->setFixedSize(40,40);
    zoomIn->setIconSize(QSize(40,40)); zoomIn->setFlat(true);
    zoomOut = new QPushButton();
    zoomOut->setIcon(QPixmap("./icons/zoom_out")); zoomOut->setFixedSize(40,40);
    zoomOut->setIconSize(QSize(40,40)); zoomOut->setFlat(true);
    connect(zoomIn,SIGNAL(clicked()),this,SLOT(zoomGantIn())); connect(zoomOut,SIGNAL(clicked()),this,SLOT(zoomGantOut()));
}

```

```

int nAct = pro->sizeActivities(); ganttTable = new QTableWidget();
ganttTable->setRowCount(nAct); ganttTable->setColumnCount(workDays->size());
//Columns
QStringList daysList;
for(int i=0;i<workDays->size();i++)
    daysList.append(workDays->at(i).toString("dd/MM/yyyy"));
ganttTable->setHorizontalHeaderLabels(daysList);
//rows
QStringList actList; vector<Activity*> vecAct = pro->getActivities();
//Creating ComboBox to display the Critical paths.
int camCrits = pro->getCriticalPaths()->size(); comboCrit = new QComboBox();    comboCrit->setFixedWidth(120);
for(int i=0;i<camCrits;i++)comboCrit->addItem(QString("Camino crítico ").append(QString::number(i+1)));
connect(comboCrit,SIGNAL(currentIndexChanged(int)),this,SLOT(refreshCCrit(int)));
for(int i=0;i<nAct;i++){
    actList.append(QString::fromStdString(vecAct.at(i)->getName()));
    if(pro->getCriticalPaths()->at(0)->includesActivity(vecAct.at(i))) paintActivity(i,vecAct.at(i)->getTMin(),vecAct.at(i)->getTNormal(),true);
    else    paintActivity(i,vecAct.at(i)->getTMin(),vecAct.at(i)->getTNormal(),false);
}
ganttTable->setVerticalHeaderLabels(actList);
QHBoxLayout *hbox = new QHBoxLayout();
hbox->addWidget(comboCrit);    hbox->addWidget(zoomIn);    hbox->addWidget(zoomOut);
hbox->setAlignment(zoomIn,Qt::AlignRight);    hbox->setAlignment(zoomOut,Qt::AlignLeft);
QVBoxLayout *vbox = new QVBoxLayout();
vbox->addLayout(hbox);    vbox->addWidget(ganttTable);
ganttWindow->setLayout(vbox);
ganttWindow->setWindowIcon(QIcon("./icons/management.png")); ganttWindow->setWindowIconText(QString(tr("Project Manager")));
ganttWindow->setMinimumSize(700,500); ganttWindow->setWindowTitle(QString("Diagrama Gantt"));
}

void Report::genResources()
{
    int nvecRes = pro->sizeResources(); vector<Resource*> *vecRes = pro->getResources(); customPlot= new QCustomPlot();
    // give the axes some labels:
customPlot->xAxis->setLabel("Días"); customPlot->yAxis->setLabel("Recursos Utilizados");
customPlot->legend->setVisible(true); customPlot->legend->setFont(QFont("Helvetica", 9));
customPlot->legend->setPositionStyle(QCPLegend::psTopRight); QPen pen;    QStringList lineNames;
int xLength = pro->getEnd()->getTMin(); int max1=0;
for(double i=0;i<nvecRes;i++){
    lineNames<< QString::fromStdString(vecRes->at(i)->getName());
    //generate data
QVector<double> yValRes = QVector<double>::fromStdVector(pro->getAllocationResourcePerDay(vecRes->at(i)->getName()));
QVector<double> xVal(xLength);
//paint plot for this resource
customPlot->addGraph();    pen.setColor(QColor(sin(i*2)*80+80, sin(i*3)*80+80, sin(i*5)*80+80));    pen.setWidthF(2.5);
customPlot->graph()->setName(lineNames.at(i));
customPlot->graph()->setPen(pen);    customPlot->graph()->setLineStyle(QCPGraph::lsStepLeft);
customPlot->graph()->setScatterStyle(QCP::ssNone);
for(int j=0;j<xLength;j++){
    if(yValRes.toStdVector().at(j)>max1)      max1=yValRes.toStdVector().at(j);
    xVal[j]=j;
}
customPlot->graph()->setData(xVal,yValRes);    customPlot->graph()->rescaleAxes(true);
}
customPlot->xAxis->setRange(0, xLength+2); customPlot->yAxis->setRange(0, max1);
customPlot->yAxis->scaleRange(1, customPlot->yAxis->range().center());
customPlot->xAxis->scaleRange(1, customPlot->xAxis->range().center());
QString dayLabel;
customPlot->xAxis->setAutoTickStep(false); customPlot->xAxis->setTickStep(2);
customPlot->yAxis->setAutoTickStep(false); customPlot->yAxis->setTickStep(2);

for(int i=0;i<workDays->size();i+=2) dayLabel.append(workDays->at(i).toString("dd/MM/yyyy"));
customPlot->xAxis->setAutoTickLabels(false)
customPlot->xAxis->setTickVectorLabels(dayLabel);
customPlot->xAxis->setTickLabelRotation(70);
customPlot->setWindowTitle(QString("Histograma Uso de Recursos"));
customPlot->setGeometry(this->geometry().x()+100,this->geometry().y()+100,customPlot->width(),customPlot->height());
}

```

7.1.13 ResourcesView.h

```
#ifndef RESOURCES_H
#define RESOURCES_H
#include <QWidget>
#include <QVector>
#include <QErrorMessage>
#include <QLabel>
#include <QSpinBox>
#include <QLineEdit>
#include <QDoubleSpinBox>
#include <QCheckBox>
#include <QTableWidget>
#include "Project.h"
#include <QMessageBox>
#include "Exceptions.h"
namespace Ui {
class Resources;
}
class resourcesView : public QWidget
{
    Q_OBJECT
public:
    explicit resourcesView(Project *proyecto, QWidget *parent = 0);
    ~resourcesView();
    QPushButton *acceptDelete;
private slots:
    void createAddResource(); void addResource(); void editResource();
    void deleteResource(); void cancelNewResource(); void changeCurrentRow(int row,int col);
signals:
    void resDeleted();
private:
    //Add Resource window
    void createAddResourceWindow(); void updateTable(); void setValues(); void clearAll();
    QWidget *newResource; QCheckBox *resUnlimited; QLineEdit *resName; QSpinBox *resAvailable; QPushButton *save,*cancel;
    bool edit; int row2edit;
    //Resources window
    Project *pro; Ui::Resources *ui;
protected:
    void closeEvent(QCloseEvent * event){
        newResource->close();
        this->close();
    }
};
#endif // RESOURCES_H
```

7.1.14 ResourcesView.cpp

```
#include "ResourcesView.h"
#include "ui_ResourcesView.h"
#include <stdio.h>
resourcesView::resourcesView(Project *proyecto, QWidget *parent) :
    QWidget(parent),
    ui(new Ui::Resources){
    //initializing variables
    this->pro=proyecto; this->setWindowIcon(QIcon("./icons/management.png")); this->setWindowTitleText(QString(tr("Project Manager")));
    //Creating windows
    edit = false; resAvailable = new QSpinBox(); resAvailable->setMaximum(99999999);
    resUnlimited = new QCheckBox(); resUnlimited->setText(QString("Recurso Ilimitado"));
    createAddResourceWindow();
    ui->setupUi(this); ui->ResourceTable->clearContents(); ui->ResourceTable->setRowCount(0); updateTable();
    ui->ResourceTable->setEditTriggers(QTableWidget::NoEditTriggers);
    //ICONS
    ui->AddResource_Button->setIcon(QPixmap("./icons/add.png")); ui->AddResource_Button->setIconSize(QSize(40,40));
    ui->EditResource_Button->setIcon(QPixmap("./icons/edit.png")); ui->EditResource_Button->setIconSize(QSize(40,40));
    ui->Exit_Button->setIcon(QPixmap("./icons/exit.png")); ui->Exit_Button->setIconSize(QSize(40,40));
    ui->DeleteResource_Button->setIcon(QPixmap("./icons/minus.png")); ui->DeleteResource_Button->setIconSize(QSize(40,40));
    //Tooltips
    ui->AddResource_Button->setToolTip(QString("Añadir Recurso")); ui->EditResource_Button->setToolTip(QString("Editar recurso"));
    ui->DeleteResource_Button->setToolTip(QString("Eliminar recurso seleccionado")); ui->Exit_Button->setToolTip(QString("Salir"));
    ui->DeleteResource_Button->setEnabled(false); ui->EditResource_Button->setEnabled(false);
    //Connecting
    connect(ui->AddResource_Button,SIGNAL(clicked()),this,SLOT(createAddResource()));
    connect(ui->DeleteResource_Button,SIGNAL(clicked()),this,SLOT(deleteResource()));
    connect(ui->EditResource_Button, SIGNAL(clicked()),this,SLOT(editResource()));
    connect(ui->ResourceTable,SIGNAL(cellClicked(int,int)),this,SLOT(changeCurrentRow(int,int)));
    connect(ui->Exit_Button,SIGNAL(clicked()),this,SLOT(close()));
    connect(ui->ResourceTable,SIGNAL(itemChanged(QTableWidgetItem*)),this,SLOT(update()));
}
```

```

resourcesView::~resourcesView()
{
    delete ui;
}
void resourcesView::changeCurrentRow(int row, int col)
{
    ui->ResourceTable->setCurrentCell(row,col); ui->ResourceTable->selectRow(row);
    ui->EditResource_Button->setEnabled(true); ui->DeleteResource_Button->setEnabled(true);
    if(row<0){
        ui->EditResource_Button->setEnabled(false); ui->DeleteResource_Button->setEnabled(false);
    }
}
void resourcesView::createAddResourceWindow(){
    newResource = new QWidget();
    //items
    save = new QPushButton(tr("Guardar")); cancel = new QPushButton(tr("Cancelar"));
    //layout
    QLabel *resNameLab = new QLabel(tr("Nombre Recurso: ")); resName = new QLineEdit("");
    QHBoxLayout *hname = new QHBoxLayout(); hname->addWidget(resNameLab); hname->addWidget(resName);
    QHBoxLayout *hbox1 = new QHBoxLayout(); QLabel *resNLab = new QLabel(tr("Recursos Maximos: "));
    hbox1->addWidget(resNLab); hbox1->addWidget(resAvailable);
    QVBoxLayout *vbox = new QVBoxLayout(); vbox->addLayout(hbox1); vbox->addWidget(resUnlimited);
    QHBoxLayout *hbottom = new QHBoxLayout(); hbottom->addWidget(cancel); hbottom->addWidget(save);
    QVBoxLayout *vbox1 = new QVBoxLayout(); vbox1->addLayout(hname); vbox1->addLayout(vbox); vbox1->addLayout(hbottom);
    newResource->setWindowTitle(tr("Nuevo Recurso"));
    //connecting
    connect(save,SIGNAL(clicked()),this,SLOT(addResource())); connect(cancel,SIGNAL(clicked()),this,SLOT(cancelNewResource()));
    newResource->setWindowIcon(QIcon("./icons/management.png")); newResource->setWindowIconText(QString(tr("Project Manager")));
    newResource->setLayout(vbox1); newResource->setFixedSize(250,200);
}
void resourcesView::createAddResource(){
    clearAll(); newResource->show();
}

void resourcesView::addResource(){
    if(resName->isEnabled()) //New
    {
        if(!(resName->text().isEmpty())){
            if(!pro->getResource(resName->text().toStdString())){
                int aux = ui->ResourceTable->rowCount(); ui->ResourceTable->setRowCount(aux+1);
                QTableWidgetItem *item = new QTableWidgetItem(resName->text()); item->setFlags(Qt::ItemIsSelectable);
                ui->ResourceTable->setItem(aux,0,item);
                if(resUnlimited->isChecked()){
                    ui->ResourceTable->setItem(aux,1,new QTableWidgetItem(QString("Ilimitado")));
                    pro->addResource(resName->text().toStdString(),99999999);
                }else{
                    ui->ResourceTable->setItem(aux,1,new QTableWidgetItem(resAvailable->text()));
                    pro->addResource(resName->text().toStdString(),resAvailable->text().toInt());
                }
                clearAll(); newResource->close();
            }else{
                showErrorInfo(E_RESOURCE_ALREADY_EXIST,"");
            }else{
                QMessageBox *help = new QMessageBox(); help->information(0,QString("Ayuda"),QString("Dale nombre al Recurso"));
            }
        }else{
            int aux = row2edit; bool canEdit=true; int unAsig=0;
            for(int i=0;i<pro->sizeActivities();i++){
                if(pro->getActivities().at(i)->getResource(resName->text().toStdString())!=NULL){
                    unAsig=pro->getActivities().at(i)->getResource(resName->text().toStdString())->units_asig;
                    if(unAsig>0) canEdit=false;
                }
            }
            if(canEdit){
                QTableWidgetItem *item = new QTableWidgetItem(resName->text());
                item->setFlags(Qt::ItemIsSelectable); ui->ResourceTable->setItem(aux,0,item);
                if(resUnlimited->isChecked()){
                    ui->ResourceTable->setItem(aux,1,new QTableWidgetItem(QString("Ilimitado")));
                    pro->modifyResource(resName->text().toStdString(),99999999);
                }else{
                    ui->ResourceTable->setItem(aux,1,new QTableWidgetItem(resAvailable->text()));
                    pro->modifyResource(resName->text().toStdString(),resAvailable->text().toInt());
                }
            }else{
                showErrorInfo(E_VALUE_RESOURCE, "No se puede reducir el numero de unidades si el recurso "+resName->text().toStdString()+" esta
asignado");
                clearAll(); newResource->close();
            }
        }
    }
    void resourcesView::cancelNewResource(){
        clearAll(); newResource->close();
    }
}

```

```

void resourcesView::setValues(){
    clearAll();
    resName->setText(ui->ResourceTable->item(row2edit,0)->text()); resName->setEnabled(false);
    int nResMax = pro->getResource(resName->text().toStdString())->getUnitsMax();
    if(nResMax>=99999999){
        resUnlimited->setChecked(true); resAvailable->setValue(0);
    }
    else resAvailable->setValue(nResMax);
    newResource->setWindowTitle(tr("Edita Recurso")); edit = true;
}
void resourcesView:: clearAll()
{
    resName->clear(); resName->setEnabled(true); resAvailable->setValue(0); resUnlimited->setChecked(false);
    newResource->setWindowTitle(tr("Nuevo Recurso")); edit = false;
}
void resourcesView:: updateTable(){
vector<Resource*> *vecRes = pro->getResources(); int resSize=pro->sizeResources();
for(int i=0;i<resSize;i++){
    ui->ResourceTable->setRowCount(ui->ResourceTable->rowCount()+1);
    QTableWidgetItem *item = new QTableWidgetItem(QString::fromStdString(vecRes->at(i)->getName()));
    item->setFlags(Qt::ItemIsSelectable); ui->ResourceTable->setItem(i,0,item);
    ui->ResourceTable->setItem(i,1,new QTableWidgetItem(QString::number(vecRes->at(i)->getUnitsMax())));
}
}
void resourcesView:: deleteResource(){
int row2delete = ui->ResourceTable->currentRow();
if(row2delete>=0){
    QMessageBox mes; mes.setText(QString("Eliminar Actividad"));
    mes.setInformativeText(QString(tr("Se borrarán todas las relaciones de esta actividad ¿Esta seguro?\n")));
    QAbstractButton *cancelDelete = mes.addButton(("Cancelar"),QMessageBox::NoRole);
    QAbstractButton *acceptDelete = mes.addButton(("Aceptar"),QMessageBox::YesRole);
    mes.setIcon(QMessageBox::Question); mes.exec();
    if(mes.clickedButton()==acceptDelete){
        pro->deleteResource(ui->ResourceTable->item(row2delete,0)->text().toStdString());
        ui->ResourceTable->removeRow(row2delete);
        ui->ResourceTable->setCurrentCell(-1,-1); ui->DeleteResource_Button->setEnabled(false);
        ui->EditResource_Button->setEnabled(false); emit resDeleted();
    }
    else{
        QMessageBox mes; mes.information(0,QString(tr("Aviso")),QString(tr("Debe seleccionar una fila\n de la tabla")));
    }
}
void resourcesView:: editResource(){
    row2edit = ui->ResourceTable->currentRow();
    if(row2edit>=0){ setValues(); newResource->show();
    }else{
        QMessageBox mes; mes.information(0,QString(tr("Aviso")),QString(tr("Debe seleccionar una fila\n de la tabla")));
    }
}

```

7.1.15 VentanaPrincipal.h

```
#ifndef VENTANAPRINCIPAL_H
#define VENTANAPRINCIPAL_H

#include <iostream>
#include <stdlib.h>
#include <qevent.h>
#include <iomanip>
#include < QMainWindow>
#include < QCalendarWidget>
#include <QStandardItemModel>
#include <QStandardItem>
#include < QVector>
#include <QCheckBox>
#include <QMessageBox>
#include <QFileDialog>
#include <QTextBlock>
#include <QPlainTextEdit>
#include <QShortcut>
#include <QDesktopServices>
#include <QUrl>
#include <fstream>
#include "calendar.h"
#include "ResourcesView.h"
#include "GenerateReport.h"
#include "GenerateAnalysis.h"
#include "Project.h"
#include "Report.h"
#include "ActivityView.h"

namespace Ui {
class MainWindow;
}

class MainWindow : public QMainWindow
{
    Q_OBJECT
public:
    explicit MainWindow(QWidget *parent = 0);
    int NAct; int NCals; bool res_op; bool cal_op;
    Project *project; Activity *act;
    ~MainWindow();
private slots:
    void AddActivity(); void EditActivity(); void DeleteActivity();
    void OpenCalendar(); void OpenResources(); void GenerateReport(); void GenerateLast();
    void changeReport(); void changeReport2(); void ChangeCurrentRow(int row,int column);
    void ShowManual(); void ShowAcercaDe(); void ShowAcercaDeQT(); void LoadProject();
    void SaveProject(); void SaveAs(); void doNewProject(); void windowFull(); void showOptions();
    void windowCustom(); void windowBasic(); void OpenAnalysis(); void chooseAnalysis();
    void resourceDeleted(){ updateMainTable(); };
private:
    void updateProject(); void updateMainTable();
    void save(std::string filename, Project * p);
    Project * loadPro(std::string filename);
    int posWinX, posWinY;
    calendar *cal; QString *weekEnd; resourcesView *res;
    Window *genRep, *genAgain; ActivityView *newAct; ActivityView *editAct; GenerateAnalysis *genAn;
    bool calOp,resOp,gen1Op,gen2Op,genAnOp,act1Op,act2Op,anOp;
    bool changes, options, firstCal, firstAn;
    bool proGenerated; QString fileName;
//Choose Analysis Window
QWidget *optionsAnalysis; QRadioButton *proRadio, *actRadio; QPushButton *contButton, *backButton; QComboBox *actCombo;
//CustomWindow Options.
QWidget *optionsWin; QVector<QCheckBox*> checks; QPushButton *accepOpt;
QVector<QDate> *festivos; QVector<QDate> *laborables; QVector<QDate> *workDates;
Ui::MainWindow *ui;
void connectAll();
protected:
    void closeEvent(QCloseEvent * event){
        if(!calOp) cal->close();
        if(!resOp) res->close();
        if(!gen1Op) genRep->close();
        if(!gen2Op) genAgain->close();
        if(!genAnOp) genAn->close();
        if(!act1Op) newAct->close();
        if(!act2Op) editAct->close();
        this->close();
    }
};
#endif // MAINWINDOW_H
```

7.1.16 VentanaPrincipal.cpp

```
#include "VentanaPrincipal.h"
#include "ui_VentanaPrincipal.h"
#include <QTextCodec>
#include <stdio.h>

//MAINWINDOW FUNCTIONS AND SLOTS.
MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow)
{
    this->setWindowIcon(QIcon("./icons/logo.ico"));
    this->setWindowTitle(QString(tr("Project Manager")));
    QTextCodec *linuxCodec = QTextCodec::codecForName("UTF-8"); QTextCodec::setCodecForTr(linuxCodec);
    QTextCodec::setCodecForCStrings(linuxCodec); ui->setupUi(this);

    //Initialization variables
    project = new Project(); festivos = new QVector<QDate>(0); laborables = new QVector<QDate>(0); workDates = new QVector<QDate>(0);
    weekEnd = new QString("NONE"); firstCal=firstAn=true; proGenerated = false;
    ui->ActivTable->setEditTriggers(QTableWidget::NoEditTriggers); ui->actionGuardar_Proyecto->setEnabled(false);
    ui->acGen_Ultimo->setEnabled(false); //At least we need a Starting day.
    ui->editButton->setEnabled(false); ui->deleteButton->setEnabled(false);
    calOp=resOp=gen1Op=gen2Op=act1Op=act2Op=genAnOp=anOp=true;
    //Creating windows
    cal = new calendar(festivos,laborables,weekEnd); newAct = new ActivityView(project,ui->ActivTable); res = new resourcesView(project);
    options=false;
    posWinX = this->geometry().x(); posWinY = this->geometry().y();
    connectAll();
}

void MainWindow::connectAll()
{
    //ICONS
    ui->addButton->setIcon(QPixmap("./icons/add.png")); ui->analysisButton->setIconSize(QSize(40,40));
    ui->analysisButton->setIcon(QPixmap("./icons/analysis.png")); ui->analysisButton->setIconSize(QSize(40,40));
    ui->calendarButton->setIcon(QPixmap("./icons/calendar.png")); ui->calendarButton->setIconSize(QSize(40,40));
    ui->editButton->setIcon(QPixmap("./icons/edit.png")); ui->editButton->setIconSize(QSize(40,40));
    ui->reportButton->setIcon(QPixmap("./icons/report.png")); ui->reportButton->setIconSize(QSize(40,40));
    ui->deleteButton->setIcon(QPixmap("./icons/minus.png")); ui->deleteButton->setIconSize(QSize(40,40));
    ui->resourceButton->setIcon(QPixmap("./icons/resources.png")); ui->resourceButton->setIconSize(QSize(40,40));
    ui->actionAcerca_de->setIcon(QPixmap("./icons/about.png")); ui->actionAcerca_de->setIconVisibleInMenu(true);
    ui->actionManual->setIcon(QPixmap("./icons/manual.png")); ui->actionManual->setIconVisibleInMenu(true);
    ui->actionGuardar_Como->setIcon(QPixmap("./icons/save_file.png")); ui->actionGuardar_Como->setIconVisibleInMenu(true);
    ui->actionCargar_Proyecto->setIcon(QPixmap("./icons/open_file.png")); ui->actionCargar_Proyecto->setIconVisibleInMenu(true);
    ui->actionSalir->setIcon(QPixmap("./icons/exit.png")); ui->actionSalir->setIconVisibleInMenu(true);
    ui->acGen_Informe->setIcon(QPixmap("./icons/report.png")); ui->acGen_Informe->setIconVisibleInMenu(true);
    ui->actionNuevo_Proyecto->setIcon(QPixmap("./icons/new_file.png")); ui->actionNuevo_Proyecto->setIconVisibleInMenu(true);

    //Toolips
    ui-> addButton->setToolTip(QString("Nueva Actividad"));
    ui-> editButton->setToolTip(QString("Editar Actividad"));
    ui-> deleteButton->setToolTip(QString("Eliminar Actividad"));
    ui-> calendarButton->setToolTip(QString("Mostrar Calendario"));
    ui-> resourceButton->setToolTip(QString("Mostrar Recursos"));
    ui-> reportButton->setToolTip(QString("Generar Informe"));

    //Buttons activation
    connect(ui-> addButton, SIGNAL(clicked()), this, SLOT(AddActivity()));
    connect(ui-> editButton, SIGNAL(clicked()), this, SLOT(EditActivity()));
    connect(ui-> deleteButton, SIGNAL(clicked()), this, SLOT(DeleteActivity()));
    connect(ui-> calendarButton, SIGNAL(clicked()), this, SLOT(OpenCalendar()));
    connect(ui-> resourceButton, SIGNAL(clicked()), this, SLOT(OpenResources()));
    connect(ui-> acGen_Informe, SIGNAL(triggered()), this, SLOT(GenerateReport()));
    connect(ui-> reportButton, SIGNAL(clicked()), this, SLOT(GenerateReport()));
    connect(ui-> acGen_Ultimo, SIGNAL(triggered()), this, SLOT(GenerateLast()));
    connect(ui-> analysisButton, SIGNAL(clicked()), this, SLOT(OpenAnalysis()));

    //Show options
    connect(ui-> actionSalir, SIGNAL(triggered()), this, SLOT(close()));
    connect(ui-> actionManual, SIGNAL(triggered()), this, SLOT>ShowManual());
    connect(ui-> actionAcerca_de, SIGNAL(triggered()), this, SLOT>ShowAcercaDe());
    connect(ui-> actionAcerca_de_QT, SIGNAL(triggered()), this, SLOT>ShowAcercaDeQT());
    connect(ui-> actionGuardar_Proyecto, SIGNAL(triggered()), this, SLOT(SaveProject()));
    connect(ui-> actionGuardar_Como, SIGNAL(triggered()), this, SLOT(SaveAs()));
    connect(ui-> actionCargar_Proyecto, SIGNAL(triggered()), this, SLOT(LoadProject()));
    connect(ui-> actionNuevo_Proyecto, SIGNAL(triggered()), this, SLOT(doNewProject()));
    connect(ui-> actionModo_Completo, SIGNAL(triggered()), this, SLOT(windowFull()));
    connect(ui-> actionModo_Simple, SIGNAL(triggered()), this, SLOT(windowBasic()));
    connect(ui-> actionPersonalizar, SIGNAL(triggered()), this, SLOT(showOptions()));

    //Table
    connect(ui-> ActivTable, SIGNAL(cellClicked(int,int)), this, SLOT(ChangeCurrentRow(int,int)));
    connect(ui-> ActivTable, SIGNAL(itemChanged(QTableWidgetItem*)), this, SLOT(update()));
}
```

```

//ShortCuts
QShortcut *shortcut1 = new QShortcut(QKeySequence("Ctrl+N"), this); QShortcut *shortcut2 = new QShortcut(QKeySequence("Ctrl+E"), this);
QShortcut *shortcut3 = new QShortcut(QKeySequence("Ctrl+X"), this); QShortcut *shortcut4 = new QShortcut(QKeySequence("Ctrl+C"), this);
QShortcut *shortcut5 = new QShortcut(QKeySequence("Ctrl+R"), this); QShortcut *shortcut6 = new QShortcut(QKeySequence("Ctrl+O"), this);
QShortcut *shortcut7 = new QShortcut(QKeySequence("Ctrl+G"), this); //Generate with last report config
QShortcut *shortcut8 = new QShortcut(QKeySequence("Ctrl+S"), this); QShortcut *shortcut9 = new QShortcut(QKeySequence("Ctrl+L"), this);
QShortcut *shortcut10 = new QShortcut(QKeySequence("Ctrl+Shift+S"), this);
QShortcut *shortcut11 = new QShortcut(QKeySequence("Ctrl+A"), this);

QObject::connect(shortcut1, SIGNAL(activated()), this, SLOT(AddActivity()));
QObject::connect(shortcut2, SIGNAL(activated()), this, SLOT(EditActivity()));
QObject::connect(shortcut3, SIGNAL(activated()), this, SLOT(DeleteActivity()));
QObject::connect(shortcut4, SIGNAL(activated()), this, SLOT(OpenCalendar()));
QObject::connect(shortcut5, SIGNAL(activated()), this, SLOT(OpenResources()));
QObject::connect(shortcut6, SIGNAL(activated()), this, SLOT(GenerateReport()));
QObject::connect(shortcut7, SIGNAL(activated()), this, SLOT(GenerateLast())); //Generate with last report config
QObject::connect(shortcut8, SIGNAL(activated()), this, SLOT(SaveProject()));
QObject::connect(shortcut9, SIGNAL(activated()), this, SLOT(LoadProject()));
QObject::connect(shortcut10, SIGNAL(activated()), this, SLOT(SaveAs()));
QObject::connect(shortcut11, SIGNAL(activated()), this, SLOT(OpenAnalysis()));

}

MainWindow::~MainWindow(){
    delete ui;
}

// LOAD & SAVE
Project* MainWindow::loadPro(std::string filename)
{
    int n_activities; string name, name2; string date_project;
    int t_normal; int t_topo; int units_max; float cost_normal; float cost_oportunity; int n_resources;
    ifstream infile; string line;
    /*Devuelve el proyecto o NULL si no se ha podido cargar*/
    int pos=filename.find_last_of(".gpi");
    if(pos<0) { showErrorInfo(E_FORMAT_FILE, filename); return NULL; }
    if(pos+1!=filename.size()) { showErrorInfo(E_FORMAT_FILE, filename); return NULL; }
    infile.open(filename.c_str(), std::ios_base::binary);
    if(!infile.is_open()) { showErrorInfo(E_OPEN_FILE, filename); return NULL; }
    Project * p= new Project();
    do { infile>>line; } while(line.compare("ACTIVITIES")!=0);
    infile>>n_activities;
    for(int i=0; i<n_activities;++) {
        infile>>name>>t_normal>>cost_normal>>t_topo>>cost_oportunity;
        p->addActivity(name,t_normal,t_topo,cost_normal,cost_oportunity);
    }
    do { infile>>line; } while(line.compare("RELATIONS")!=0);
    do {
        infile>>name>>name2;
        if(name.compare("RESOURCES")==0)
            break;
        p->addRelation(name, name2);
    } while(!name.empty());
    if(name.compare("RESOURCES")!=0) {
        do { infile>>line; } while(line.compare("RESOURCES")!=0);
        infile>>n_resources;
    }else n_resources=atoi(name2.c_str());
    for(int i=0; i<n_resources; i++) { infile>>name>>units_max; p->addResource(name, units_max); }
    infile>>name>>name2>>units_max;
    while(!infile.eof()&&name.compare("")!=0) {
        p->allocateResourceActivity(name, name2, units_max); infile>>name>>name2>>units_max;
    }
    infile.close(); return p;
}
void MainWindow:: LoadProject()
{
    QString fileName = QFileDialog::getOpenFileName(this,tr("Open Project"),"","Project (*.gpi);All Files (*)");
    if(fileName.isEmpty()) return;
    else{
        QFile file(fileName);
        if (!file.open(QIODevice::ReadOnly)){
            QMessageBox::information(this,tr("No se puede abrir el fichero"),file.errorString());
            return;
        }
        delete (project); project = loadPro(fileName.toStdString());
        if(project==NULL){
            QMessageBox mes; mes.warning(0,QString("Aviso"),QString(tr("No se ha podido cargar el fichero"))); mes.show();
        }else{
            updateProject();
            QMessageBox mes; mes.information(0,QString("Aviso"),QString(tr("Se ha cargado correctamente el fichero")));
            mes.show();
        }
    }
}

```

```

void MainWindow::save(std::string filename, Project * p){
    ofstream outfile; outfile.open(filename.c_str(), std::ios_base::binary);
    if(!outfile.is_open())    showErrorInfo(E_OPEN_FILE, filename);
    else {
        outfile<<"ACTIVITIES"<<endl;
        outfile<<p->getActivities().size()<<endl;
        for(int i=0; i<p->getActivities().size();i++) {
            outfile<<p->getActivities().at(i)->getName()<<" ";      outfile<<p->getActivities().at(i)->getTNormal()<<" ";
            outfile<<p->getActivities().at(i)->getCostNormal()<<" ";      outfile<<p->getActivities().at(i)->getTTope()<<" ";
            outfile<<p->getActivities().at(i)->getOportunityCost()<<endl;
        }
        outfile<<"RELATIONS"<<endl;
        for(int i=0; i<p->getActivities().size();i++)
            if(!p->getActivities().at(i)->sucesorIsEnd())
                for(int j=0; j<p->getActivities().at(i)->getList_Activities_suc()->size();j++)
                    outfile<<p->getActivities().at(i)->getName()<<" "<<p->getActivities().at(i)->getList_Activities_suc()->at(j)->activity->getName()<<endl;

        outfile<<"RESOURCES"<<endl;    outfile<<p->getResources()->size()<<endl;
        for(int i=0; i<p->getResources()->size();i++)
            outfile<<p->getResources().at(i)->getName()<<" "<<p->getResources().at(i)->getUnitsMax()<<endl;
        for(int i=0; i<p->getActivities().size();i++)
            for(int j=0; j<p->getActivities().at(i)->getResources().size();j++)
                {
                    outfile<<p->getActivities().at(i)->getResources().at(j)->resource_asig->getName()<<" ";
                    outfile<<p->getActivities().at(i)->getName()<<" ";      outfile<<p->getActivities().at(i)->getResources().at(j)->units_asig<<endl;
                }
        }
        outfile.close();
    }
}

void MainWindow::SaveAs(){
    fileName = QFileDialog::getSaveFileName(this,tr("Save Project"), "", tr("GPI files (*.gpi);;All Files (*)")); QFile file(fileName);
    if(file.open(QIODevice::WriteOnly|QIODevice::Text)){
        QFileInfo info(file);
        if(info.suffix()!=QString("gpi")){
            fileName.append(".gpi"); QMessageBox::information(this,tr("Aviso"),tr("Se añade la extension .gpi"));
        }
        save(fileName.toStdString(),project); ui->actionGuardar_Proyecto->setEnabled(true);
    }else{ return; }
}

void MainWindow:: SaveProject(){
    if(fileName.isEmpty()) SaveAs();
    else{
        QFile file(fileName);
        if(file.open(QIODevice::WriteOnly|QIODevice::Text)){
            save(fileName.toStdString(),project); ui->actionGuardar_Proyecto->setEnabled(true);
        }else{
            QMessageBox::information(this,tr("Unable to open file"),file.errorString());
            return;
        }
    }
}

// UPDATE & CLEAN
void MainWindow:: doNewProject(){
    //deleting & closing
    delete(project); delete(festivos); delete(laborables);
    ui->ActivTable->clearContents(); ui->ActivTable->setRowCount(0);
    if(!calOp) cal->close();
    if(!resOp) res->close();
    if(!gen1Op) genRep->close();
    if(!gen2Op) genAgain->close();
    if(!genAnOp) genAn->close();
    if(!act1Op) newAct->close();
    if(!act2Op){ editAct->close(); delete(editAct);}
    delete(cal); delete(newAct); delete(res);
    //initializing
    festivos = new QVector<QDate>(0); laborables = new QVector<QDate>(0); workDates = new QVector<QDate>(0);
    ui->ActivTable->setEditTriggers(QTableWidget::NoEditTriggers);
    ui->actionGuardar_Proyecto->setEnabled(false); weekEnd->clear();
    ui->acGen_Ultimo->setEnabled(false); //At least we need a Starting day.
    ui->editButton->setEnabled(false); ui->deleteButton->setEnabled(false);
    calOp=resOp=gen1Op=gen2Op=act1Op=act2Op=genAnOp=true;
    options=false; project = new Project(); res=new resourcesView(project);
    cal = new calendar(festivos,laborables,weekEnd); newAct = new ActivityView(project,ui->ActivTable);
}

void MainWindow:: updateProject(){
    //Cleaning project
    delete(festivos); delete(laborables);
    ui->ActivTable->clearContents(); ui->ActivTable->setRowCount(0);
    if(!calOp) cal->close();
    if(!resOp) res->close();
    if(!gen1Op) genRep->close();
    if(!gen2Op) genAgain->close();
}

```

```

if(!genAnOp) genAn->close();
if(!act1Op) newAct->close();
if(!act2Op){ editAct->close(); delete(editAct);}
delete(cal); delete(newAct); delete(res);
ui->ActivTable->setEditTriggers(QTableWidget::NoEditTriggers);
ui->actionGuardar_Proyecto->setEnabled(false);
ui->acGen_Ultimo->setEnabled(false); //At least we need a Starting day.
ui->editButton->setEnabled(false);
ui->deleteButton->setEnabled(false);
calOp=resOp=gen1Op=gen2Op=act1Op=act2Op=genAnOp=true;
options=false; weekEnd->clear();
//UpdateMainTable
updateMainTable();
//Update Widnows:
festivos = new QVector<QDate>(0); laborables = new QVector<QDate>(0); workDates = new QVector<QDate>(0);
cal = new calendar(festivos,laborables,weekEnd);
newAct = new ActivityView(project,ui->ActivTable);
res = new resourcesView(project);
}

void MainWindow:: updateMainTable(){
//Loading project
vector<Activity*> vecAct = project->getActivities(); vector<Resource*> *vecRes = project->getResources();
int actSize = project->sizeActivities(); int resSize = project->sizeResources();
//Update Ventana Principal
for(int i=0;i<actSize;i++) {
    ui->ActivTable->setRowCount(i+1); ui->ActivTable->setItem(i,0,new QTableWidgetItem(QString::fromStdString(vecAct.at(i)->getName())));
    int tn = vecAct.at(i)->getTNormal(); int tp = vecAct.at(i)->getTTope();
    float cn = vecAct.at(i)->getCostNormal(); float co = vecAct.at(i)->getOportunityCost();
    ui->ActivTable->setItem(i,2,new QTableWidgetItem(QString::number(tn)));
    ui->ActivTable->setItem(i,4,new QTableWidgetItem(QString::number(tp)));
    ui->ActivTable->setItem(i,5,new QTableWidgetItem(QString::number(cn)));
    ui->ActivTable->setItem(i,6,new QTableWidgetItem(QString::number(co)));
    //Resources
    QString *textRes = new QString(" "); bool any=false; Activity *actAux = vecAct.at(i);
    for(int j=0;j<actAux->getResources().size();j++){
        int auxN = actAux->getResources().at(j)->units_asig;
        if(any) textRes->append(", ");
        textRes->append(QString::fromStdString(actAux->getResources().at(j)->resource_asig->getName()));
        textRes->append(");textRes->append(QString::number(auxN)); textRes->append(")");
        any=true;
    }
    ui->ActivTable->setItem(i,1,new QTableWidgetItem(*textRes));
    //Predecesores
    QString *textPred = new QString(""); any=false; int nSize=actSize;
    for(int j=0;j<nSize;j++){
        if(vecAct.at(i)->getPositionRelationPredecesor(vecAct.at(j)->getName())>=0){
            if(any) textPred->append(",");
            any=true; textPred->append(QString::fromStdString(vecAct.at(j)->getName()));
        }
    }
    ui->ActivTable->setItem(i,3,new QTableWidgetItem(*textPred));
}
}

// MAIN WINDOW OPTIONS
void MainWindow:: ShowManual(){
    QString path="/Docs/Manual.pdf"; QString appath=qApp->applicationDirPath(); QString fullpath=appath.prepend("file:///")+path;
    QDesktopServices::openUrl(QUrl(fullpath, QUrl::TolerantMode));
}

void MainWindow:: ShowAcercaDe(){
    QMessageBox aboutBox(this);
    aboutBox.setText(tr("Curso 2012/2013 - ETSINF - UPV.\n" "GPI - Intensificacion Sistemas de Informacion"));
    aboutBox.setInformativeText(tr("Este Software ha sido desarrollado con motivo academico" "para facilitar la gestion de proyectos informaticos segun"
        "los conceptos e ideas impartidas durante el curso en la" "asignatura Gestion de Proyectos Informaticos.\n\n"
        "Autores: \nPaula Navarro Alfonso\nCarles S. Soriano Perez\n"));
    aboutBox.setWindowTitle(tr("Acerca de")); aboutBox.setIconPixmap(QPixmap("./icons/upv.png")); aboutBox.exec();
}

void MainWindow:: ShowAceraDeQT(){
    QMessageBox aboutMes; aboutMes.aboutQt(0,QString("Acerca de QT"));
}

void MainWindow:: showOptions(){
    ui->actionPersonalizar->setEnabled(false); optionsWin = new QWidget(); checks.clear();
    accepOpt = new QPushButton(QString("Aceptar")); connect(accepOpt,SIGNAL(clicked()),this,SLOT(windowCustom()));
    for(int i=0;i<6;i++){
        QCheckBox *check = new QCheckBox();
        if(ui->ActivTable->isColumnHidden(i+1)) check->setChecked(false);
        else check->setChecked(true);
        checks.insert(i,check);
    }
    checks[0]->setText(QString("Recursos")); checks[1]->setText(QString("Duración")); checks[2]->setText(QString("Predecesores"));
    checks[3]->setText(QString("T. Tope")); checks[4]->setText(QString("Coste Normal")); checks[5]->setText(QString("Coste Oport."));
    QGroupBox *basicOp = new QGroupBox(QString("Básicas"));
}

```

```

QHBoxLayout *hbox1 = new QHBoxLayout();
    hbox1->addWidget(checks[0]);    hbox1->addWidget(checks[1]);    hbox1->addWidget(checks[2]);    basicOp->setLayout(hbox1);
QGroupBox *advancedOp = new QGroupBox(QString("Avanzadas"));
QHBoxLayout *hbox2 = new QHBoxLayout();
    hbox2->addWidget(checks[3]);    hbox2->addWidget(checks[4]);    hbox2->addWidget(checks[5]);    advancedOp->setLayout(hbox2);
QHBoxLayout *hbox3 = new QHBoxLayout();
    hbox3->addWidget(aceptOpt);
QVBoxLayout *vbox = new QVBoxLayout();
    vbox->addWidget(basicOp);    vbox->addWidget(advancedOp);    vbox->addLayout(hbox3);
optionsWin->setLayout(vbox);    optionsWin->setWindowTitle(QString("Personalizar Vista"));    optionsWin->show();
}
void MainWindow::windowCustom(){
optionsWin->close();
for(int i=0;i<6;i++){
    if(!checks[i]->isChecked())
        ui->ActivTable->setColumnHidden(i+1,true);
    else
        ui->ActivTable->setColumnHidden(i+1,false);
}
ui->actionModo_Simple->setEnabled(true);    ui->actionModo_Completo->setEnabled(true);    ui->actionPersonalizar->setEnabled(false);
}
void MainWindow::windowFull(){
for(int i=0;i<7;i++)
    ui->ActivTable->setColumnHidden(i,false);
ui->actionModo_Simple->setEnabled(true);    ui->actionModo_Completo->setEnabled(false);    ui->actionPersonalizar->setEnabled(true);
}
void MainWindow::windowBasic(){
for(int i=0;i<7;i++){
    if(i<4) ui->ActivTable->setColumnHidden(i,false);
    else
        ui->ActivTable->setColumnHidden(i,true);
}
ui->actionModo_Simple->setEnabled(false);    ui->actionModo_Completo->setEnabled(true);    ui->actionPersonalizar->setEnabled(true);
}
void MainWindow::ChangeCurrentRow(int row,int col){
ui->ActivTable->setCurrentCell(row,col);    ui->ActivTable->selectRow(row);
ui->editButton->setEnabled(true);    ui->deleteButton->setEnabled(true);    ui->analysisButton->setEnabled(true);
if(row<0){
    ui->editButton->setEnabled(false);    ui->deleteButton->setEnabled(false);    ui->analysisButton->setEnabled(false);
}
}
//ACTIVITY
void MainWindow::AddActivity(){
if(!newAct->isVisible()){
    newAct = new ActivityView(project,ui->ActivTable);    newAct->show();    newAct->setFixedSize(newAct->size());
    act1Op = false;
}
}
void MainWindow::EditActivity(){
int row2edit = ui->ActivTable->currentRow();
if(row2edit>=0){
    act = new Activity();    act = project->getActivity(ui->ActivTable->item(row2edit,0)->text().toStdString());
    if(act2Op){
        editAct=new ActivityView(project,act,ui->ActivTable,row2edit);
        editAct->show(); editAct->setFixedSize(editAct->size());    act2Op = false;
    }else{
        if(!editAct->isVisible()){
            editAct = new ActivityView(project,act,ui->ActivTable,row2edit);
            editAct->show(); editAct->setFixedSize(editAct->size());
        }
    }
    ui->deleteButton->setEnabled(false);    ui->editButton->setEnabled(false);
}else{
    QMessageBox mes;    mes.information(0,QString(tr("Aviso")),QString(tr("Debe seleccionar una fila\n de la tabla")));
}
}
void MainWindow::DeleteActivity(){
int row2delete = ui->ActivTable->currentRow();
if(row2delete>=0){
    QMessageBox mes;    mes.setText(QString("Eliminar Actividad"));
    mes.setInformativeText(QString("Se borrarán todas las relaciones de esta actividad ¿Esta seguro?\n"));
    QAbstractButton *cancelDelete = mes.addButton(("Cancelar"),QMessageBox::NoRole);
    QAbstractButton *acceptDelete = mes.addButton(("Aceptar"),QMessageBox::YesRole);
    mes.setIcon(QMessageBox::Question);    mes.exec();
    if(mes.clickedButton()==acceptDelete){
        std::string aux = ui->ActivTable->item(row2delete,0)->text().toStdString();
        project->deleteActivity(aux);    ui->ActivTable->removeRow(row2delete);
        ui->ActivTable->setCurrentCell(-1,-1);    ui->deleteButton->setEnabled(false);    ui->editButton->setEnabled(false);
        updateMainTable();
    }
}else{    QMessageBox mes;    mes.information(0,QString(tr("Aviso")),QString(tr("Debe seleccionar una fila\n de la tabla")));
}
}

```

```

// NEW WINDOWS
void MainWindow::chooseAnalysis(){
    bool cont = true;
    if(!genAnOp){
        genAnOp=true;
        if(genAn->isVisible()) genAn->close();
    }
    if(!anOp && optionsAnalysis->isVisible()) return;
    optionsAnalysis = new QWidget(); proRadio = new QRadioButton(QString("Analizar Proyecto")); proRadio->setChecked(true);
    actRadio = new QRadioButton(QString("Actividad: ")); actCombo = new QComboBox();
    for(int i=0;i<project->sizeActivities();i++) actCombo->addItem(ui->ActivTable->item(i,0)->text());
    contButton = new QPushButton(QString("Continuar")); backButton = new QPushButton(QString("Cancelar"));
    connect(contButton,SIGNAL(clicked()),this,SLOT(OpenAnalysis())); connect(backButton,SIGNAL(clicked()),optionsAnalysis,SLOT(close()));
    QHBoxLayout *hact = new QHBoxLayout(); hact->addWidget(actRadio); hact->addWidget(actCombo);
    QVBoxLayout *vbox1 = new QVBoxLayout(); vbox1->addWidget(proRadio); vbox1->addLayout(hact);
    QHBoxLayout *hbot = new QHBoxLayout(); hbot->addWidget(backButton); hbot->addWidget(contButton);
    QVBoxLayout *vbox2 = new QVBoxLayout(); vbox2->addLayout(vbox1); vbox2->addLayout(hbot);
    optionsAnalysis->setLayout(vbox2); anOp=false;

    optionsAnalysis->setWindowIcon(QIcon("./icons/management.png")); optionsAnalysis->setWindowIconText(QString(tr("Project Manager")));
    optionsAnalysis->setWindowTitle(QString("Elige Tipo")); optionsAnalysis->show();
}

void MainWindow::OpenAnalysis(){
    if(project->getEnd()->getTMax()<0){/*
        QMessageBox::information(this,QString("Información"),QString("Debe de generar el proyecto antes."));
    */else{
        if(!genAnOp){ genAnOp=true; genAn->close(); }
        if(firstAn) QMessageBox::information(this,QString("Información"),QString("Los análisis corresponden al último informe generado."));
        genAn = new GenerateAnalysis(project,workDates);
        genAn->show(); genAnOp=firstAn=false; genAn->setFixedSize(genAn->size());
    }
}
//CALENDAR
void MainWindow::OpenCalendar(){
    if(firstCal) QMessageBox::information(this,QString("Información"),QString("Se han introducido los festivos Nacionales de España"));
    firstCal=false;
    if(!calOp){
        if(cal->isVisible()){ cal->close(); calOp=true;
        }else{ cal->show(); calOp=false; }
    }else{ cal->show(); cal->setFixedSize(cal->size()); calOp=false; }
}
//RESOURCES
void MainWindow::OpenResources(){
    res->show();
    if(resOp) connect(res,SIGNAL(resDeleted()),this,SLOT(resourceDeleted()));
    resOp=false;
}
//REPORT
void MainWindow::GenerateReport(){
    if(project->sizeActivities()==0){
        QMessageBox mes; mes.information(0,QString("Información"), QString("El proyecto debe tener actividades."));
        return;
    }
    if(!gen1Op){
        if(!genRep->isVisible()){
            genRep->close(); gen1Op=true;
        }else{
            genRep->show(); gen1Op=false;
        }
    }else{
        genRep = new Window(project,festivos,laborables,weekEnd,0); genRep->show();
        ui->acGen_Ultimo->setEnabled(false); connect(genRep->save,SIGNAL(clicked()),this,SLOT(changeReport()));
        connect(genRep->submit,SIGNAL(clicked()),this,SLOT(changeReport()));
        connect(genRep->cancel,SIGNAL(clicked()),this,SLOT(changeReport2()));
    }
}
void MainWindow::GenerateLast(){
    if(!gen2Op){ genAgain->generate(); updateMainTable(); workDates=genRep->workDays; }
}
void MainWindow::changeReport(){
    //Means we've saved the options
    ui->acGen_Informe->setEnabled(true); ui->acGen_Ultimo->setEnabled(true);
    gen1Op=true; gen2Op=false; this->genAgain=genRep;
    //Patch here.
    workDates=genRep->workDays; updateMainTable();
}
void MainWindow::changeReport2(){
    if(!ui->acGen_Informe->isEnabled()){
        ui->acGen_Informe->setEnabled(true);
    }
}

```

7.2 Código Completo Algoritmos

7.2.1 Activity.h

```
#include <vector>
#include <string>
#include <set>
#include "Resource.h"

using namespace std;

#ifndef ACTIVIDAD_H_
#define ACTIVIDAD_H_

class Activity {
    string name;
    /*Instantes*/
    int t_normal; // tiempo normal
    int t_tope; // tiempo maximo al que se puede reducir el tiempo normal
    int t_min; // instante de inicio minimo
    int t_max; // instante de inicio maximo
    int dif_t_normal_tope; // tiempo normal - tiempo tope , se usa en el algoritmo ackoff
    int t_secuenciacion; // parametro auxiliar , instante de secuenciacion en la limitacion de recursos

    /*Costes*/
    float cost_normal; //coste normal
    float cost_oportunity; //coste si la actividad se realizase en el tiempo tope
    /*Holguras*/
    int h_total; // holgura total
    int h_free; // holgura libre y holgura libre de retraso

public:
    /*Recurso*/
    typedef struct resource // estructura que representa la asignacion de un recurso
    {
        Resource* resource_asig; // recurso asignado
        int units_asig; // unidades del recurso asignadas a la actividad
        resource(Resource *resource_asig, int units);
    };

    std::vector<resource*> resources; // lista de los recursos asignados

    /*Relaciones*/
    typedef struct relation // estructura que representa la relacion de la actividad con otras actividades
    {
        Activity * activity; // actividad con la que se establece la relacion
        relation(Activity * activity);
    };
    vector<relation*> List_Activities_pred; // lista de actividades predecesoras
    vector<relation*> List_Activities_suc; // lista de actividades sucesoras

    /*Metodos*/
    Activity(string name, int t_normal, int t_tope, float cost_normal, float cost_oportunity);
    Activity(string name, int t_normal, float cost_normal);
    Activity(); ~Activity(); void clear(); // resetea la actividad
    void modify(int t_normal, int t_tope, float cost_normal, float cost_oportunity); //modifica la actividad
    bool isContained(vector<Activity*> list); //comprueba si la actividad esta incluida dentro del vector
    string getName() { return name; }
    void setTNormal(int time) { t_normal=time; }
    int getTNormal() { return t_normal; }
    int getTTope() { return t_tope; }

    /*Instantes*/
    void setTMin(int t_min){ this->t_min=t_min; } int getTMin(){ return t_min; }
    void setTMax(int t_max){ this->t_max=t_max; } int getTMax(){ return t_max; }
    int getEarlyStart(){ return t_min; } // Inicio temprano
    int getLateStart(){ return t_max; } // Inicio tardio
    int getEarlyEnd(){ return t_min+t_normal; } // Fin temprano
    int getLateEnd(){ return t_max+t_normal; } // Fin tardio
    int getTSecuenciacion(){ return t_secuenciacion; } void setTSecuenciacion(int t){ this->t_secuenciacion=t; }
    void setDifTNormalTope(){ this->dif_t_normal_tope=t_normal-t_tope;} //Calcula el tiempo que se puede reducir
    void setDifTNormalTope( int dif){ this->dif_t_normal_tope=dif; } int getDifTNormalTope(){ return dif_t_normal_tope; }

    /*Holguras*/
    void setHTotal(int h_total){ this->h_total=h_total; } int getHTotal(){ return h_total; }
    void setHFree(int h_free){ this->h_free=h_free; } int getHFree(){ return h_free; }
    void calculHoguraFree(int time ); // calcula la holgura libre
    int getHLR(){ return h_free; } // holgura libre de retraso
    int getHLA(); // holgura libre de adelanto

    /*Costes*/
    float getCostNormal(){ return cost_normal; } float getOportunityCost(){ return this->cost_oportunity; }

    /*Predecesora*/
    vector<relation *> *getList_Activities_pred(){return &List_Activities_pred;}; // crea una relacion desde una actividad a esta
    void addPredecesor(Activity * pred); //devuelve la posicion en el vector predecesoras donde esta la relacion con la actividad name
    int getPositionRelationPredecesor(string name); // calcula el tiempo minimo de la actividad
    int calculMinTime(); // comprueba si es una actividad de inicio
    bool predecesorIsInitial(); // comprueba si es una actividad de fin

    /*Sucesora*/
    vector<relation *> *getList_Activities_suc(){return &List_Activities_suc;}; // crea una relacion hace una actividad desde esta
    void addSucesor(Activity * succ); //devuelve la posicion en el vector sucesoras donde esta la relacion con la actividad de nombre
    int getPositionRelationSucesor(string name); // calcula el tiempo maximo de la actividad
    int calculMaxTime(); // comprueba si es una actividad de fin
    string sucresorsToString(); // obtiene la lista de nombre de las sucesoras
    bool validateDependencies(); // comprueba si se cumplen las relaciones de precedencias a la hora de calcular los instantes
};

#endif
```

```

/*Recursos*/
void incrementResource(int value, string resource);      // incrementa las unidades asignadas del recurso si este ha sido asignado previamente
void decrementResource(int value, string resource);       // decrementa las unidades asignadas del recurso si este ha sido asignado previamente
void allocateResource(Resource * resource, int value);    // asigna a la actividad unidades de un recurso, reseteando las unidades si ya estaba asignado
void deleteResource(Resource *resource);                 // elimina la asignacion que tiene la actividad del recurso resource
int getPositionResource(string resource);                // obtiene la posicion en el vector resources donde esta el recurso
Activity::resource* getResource(string name_resource);   // obtiene la asignacion del recurso a la actividad, si no esta asignado devulev null
vector<Activity::resource> getResources(){return resources;}
bool isEnoughResource(set<disp_instant, CompareDisponibilidades::iterator disponibilidad>; //comprueba si hay suficientes recursos para que pueda empezar la actividad
bool isEnoughResource(disp_instant disponibilidad);
};

/* ACTIVIDAD PROCESANDOSE EN LIMITACION DE RECURSOS*/

typedef struct
{
    int instant;                                         // estructura para representar las actividades que se estan procesando en la limitacion de recursos
    Activity * act;                                       // instante que termina de procesarse la actividad
}procesing_activity;                                    // actividad procesandose

class CompareActProcesandose{                           // clase auxiliar para ordenar de las actividades que estan procesandose por instante de fin de proceso
public:
bool operator()(procesing_activity d1, procesing_activity d2){ if (d1.instant < d2.instant) return true;  return false; }
};

string num_to_str(float n);
#endif;

```

7.2.2 Activity.cc

```

#include <iostream>
#include "Resource.h"
#include "Exceptions.h"
#include "Activity.h"
#include <set>
using namespace std;
//*********************************************************************
/*
    CONSTRUCTORES
*/
//*********************************************************************
Activity::Activity(string name,int tn, int tp, float cnormal, float cuoport){
    this->name=name;          t_normal=tn; t_top=tp;    cost_normal=cnormal;    cost_oportunity=cuoport;
    h_total=0;    h_free=0;    t_min=-1;    t_max=-1;
}
Activity::Activity(std::string name,int tn, float cnormal)
{
    this->name=name;          t_normal=tn; t_top=tn;
    cost_normal=cnormal;    cost_oportunity=cnormal; h_total=0;
    h_free=0;    t_min=-1;    t_max=-1;
}
Activity::Activity(){}
/*
    DESTRUCTOR
*/
//*********************************************************************
Activity::~Activity(){
    /*Destruir las relaciones*/
    for(int i=0;<getList_Activities_pred()->size();i++)
        delete(getList_Activities_pred()->at(i));
    for(int i=0;<getList_Activities_suc()->size();i++)
        delete(getList_Activities_suc()->at(i));
    /*Destruir las asignaciones de recurso*/
    for(int i=0;<resources.size();i++)
        delete(resources.at(i));
}
//*********************************************************************
/*
    METODOS
*/
//*********************************************************************
void Activity::clear(){
    /*Iniciaiza la actividad*/
    h_total=0;    h_free=0;    t_min=-1;    t_max=-1;
}

void Activity::modify(int t_normal, int t_top, float cost_normal, float cost_oportunity){
    /*Modifica los parametros de la actividad*/
    this->t_normal=t_normal; this->t_top=t_top;    this->cost_normal=cost_normal;    this->cost_oportunity=cost_oportunity;
}
bool Activity::predecesorInitial(){
    /*Comprueba si su predecesora es la actividad ficticia inicio*/
    if(getList_Activities_pred()->size()==1&&getList_Activities_pred()->at(0)->activity->getName()=="Inicio")
        return true;
    return false;
}
bool Activity::succesorsEnd(){
    /*Comprueba si su sucesora es la actividad ficticia fin*/
    if(getList_Activities_suc()->size()==1&&getList_Activities_suc()->at(0)->activity->getName()=="Fin")
        return true;
    return false;
}
//*********************************************************************
/*
    RELACIONES
*/
//*********************************************************************
Activity::relation::relation(Activity * act){ this->activity=act;}
void Activity::addSucesor(Activity * succ){
    /*Añadir una sucesora*/
    relation *relation_succ=new relation(succ);this->List_Activities_suc.push_back(relation_succ);
}

void Activity::addPredecesor(Activity * pred){

```

```

    /*Añadir una predecesora*/
    relation *relation_pred=new relation(pred);           this->List_Activities_pred.push_back(relation_pred);
}
int Activity::getPositionRelationPredecesor(string name_pred){
    /*Obtener la posicion dentro de las relaciones de precedencia, donde se encuentra la relacion con la actividad dada*/
    for(int i=0; i<this->getList_Activities_pred()->size();i++) {
        if(getList_Activities_pred()->at(i)->activity->name.compare(name_pred)==0)      return i;
    }
    return -1;// no es predecesora
}
int Activity::getPositionRelationSucesor(std::string name){
    /*Obtener la posicion dentro de las relaciones de sucesoras, donde se encuentra la relacion con la actividad dada*/
    for(int i=0; i<this->getList_Activities_suc()->size();i++) {
        if(getList_Activities_suc()->at(i)->activity->name.compare(name)==0)      return i;
    }
    return -1; // no es sucesora
}
bool Activity::isContained(std::vector<Activity*> List){
    /*Comprueba si la actividad esta dentro del vector dado*/
    for(int j=0; j<List.size();j++)  if(List.at(j)->getName()==getName()) return true;
    return false;
}
bool Activity::validateDependences(){
    bool result=true;
    /*Comprueba si las dependencias se cumplen*/
    for(int i=0; i<this->List_Activities_suc.size();i++){
        /*La sucesoras no pueden empezar antes de que terminen las predecesoras*/
        if(t_min+t_normal>List_Activities_suc.at(i)->activity->t_min) {
            showErrorInfo(E_DEPENDENCES, name+" TN="+num_to_str(t_min+t_normal)+" y "+List_Activities_suc.at(i)->activity->name+""
TN=num_to_str(List_Activities_suc.at(i)->activity->t_min));
            result=false;
        }
    }
    return result;
}
std::string Activity::sucresorsToString(){
    /*Obtener las actividades sucesoras en formato string*/
    std::string path_text="";
    /*No indicamos como sucesora la actividad ficticia fin*/
    if(this->sucresorsEnd())          return path_text;
    /*String de nombres de las sucesoras separadas por guion*/
    for(int i=0; i<this->List_Activities_suc.size(); i++) {
        if(i!=0) path_text.append("-");
        path_text.append(List_Activities_suc.at(i)->activity->getName());
    }
    return path_text;
}
***** CAMINO CRITICO *****
int Activity::calculMinTime(){
    int max=-1;  Activity * act;
    if(List_Activities_pred.empty())      return 0;
    /*Examinar lista de predecesoras*/
    for(int i=0; i<List_Activities_pred.size();i++) {
        act=List_Activities_pred.at(i)->activity;
        /*Si la sucesora no tiene tiempo minimo, calcularlo*/
        if(act->t_min==1)      act->t_min=act->calculMinTime();
        /*Maximo instante de finalizacion temprano entre las predecesoras*/
        if(max<act->t_min+act->t_normal)      max=act->t_min+act->t_normal;
    }
    return max;
}
int Activity::calculMaxTime(){
    int min=1000000000000000; // infinito
    Activity * act;
    if(List_Activities_suc.empty())      return t_min;
    /*Examina las sucesoras*/
    for(int i=0; i<List_Activities_suc.size();i++) {
        act=List_Activities_suc.at(i)->activity;
        /*Si la sucesora no tiene calculado el TMAX, calcularlo*/
        if(act->t_max==-1){
            act->t_max=act->calculMaxTime();
            /*Holgura total*/
            act->h_total=act->t_max-act->t_min;
        }
        /*Minimo instante de comienzo mas tardio de las sucesoras*/
        if(min>act->t_max-t_normal)      min=act->t_max-t_normal;
    }
    return min;
}
void Activity::calculHolguraFree( int time){
    h_free=time; // como maximo usamos la propia duracion del proyecto
    /*Es la actividad de fin*/
    if(List_Activities_suc.empty()) h_free=time-(t_normal+t_min);
    else{
        /*Explora sus sucesoras*/
        for(int i=0; i<this->List_Activities_suc.size();i++){
            /*Minima diferencia entre el TMIN de una sucesora y el instante de finalizacion mas temprano de la actividad*/
            if(h_free>List_Activities_suc.at(i)->activity->t_min-(t_normal+t_min))
                h_free=List_Activities_suc.at(i)->activity->t_min-(t_normal+t_min);

            /*Calculamos la holgura libre para la sucesora*/
            List_Activities_suc.at(i)->activity->calculHolguraFree(time);
        }
    }
}

```

```

int Activity::getHLA() //holgura libre de retraso
{
    Activity * act; int max=0;
    if(List_Activities_pred.empty())
        return 0;
    /*Examinar lista de predecesoras*/
    for(int i=0; i<List_Activities_pred.size();i++) {
        act=List_Activities_pred.at(i)->activity;
        /*Maximo intante de comienzo mas tarde entre las predecesoras*/
        if(max<act->t_min+act->t_normal)      max=act->t_min+act->t_normal;
    }
    /*Tiempo desde que podria empezar la actividad hasta que realmente empieza*/
    return t_min-max;
}
//*********************************************************************
/*
 RECURSOS
 *****/
Activity::resource(Resource *resource_asig, int units){
    this->resource_asig=resource_asig;      this->units_asig=units;
}
int Activity::getPositionResource(string name_rec){
    /*Posicion dentro de las asignaciones, donde esta la asignacion del recurso dado*/
    for(int i=0; i<this->resources.size();i++)    if(resources.at(i)->resource_asig->getName()==name_rec)      return i;
    return -1; // no ha sido asignado
}
Activity::resource* Activity::getResource(string name_rec){
    /*Obtener la asignacion del recurso dado*/
    for(int i=0; i<resources.size();i++)
        if(resources.at(i)->resource_asig->getName()==name_rec) return resources.at(i);
    return NULL; // no ha sido asignado
}
void Activity::incrementResource(int value, string name){
    int pos=getPositionResource(name);
    if(pos== -1) showErrInfo(E_RESOURCE_NOT_ALLOCATED, name+" no esta asignado a la actividad "+this->name);
    /*El incremento de la asignacion superara al maximo de unidades del recurso?*/
    else if(resources.at(pos)->units_asig+value>resources.at(pos)->resource_asig->getUnitsMax())
        showErrInfo(E_RESOURCE_MAX_EXCEEDED, name);
    else this->resources.at(pos)->units_asig+=value;
}
void Activity::decrementResource(int value, string name_resource){
    int pos=getPositionResource(name_resource);
    if(pos== -1) showErrInfo(E_RESOURCE_NOT_ALLOCATED, name_resource+" no esta asignado a la actividad "+this->name);
    /*No se puede quitar lo que no se tiene asignado*/
    else if(resources.at(pos)->units_asig<value)
        showErrInfo(E_VALUE_RESOURCE, "Se esta intentando asignar un value negativo al resource "+name_resource);
    else{
        resources.at(pos)->units_asig-=value;
        /*Si no hay unidades asignadas eliminamos el la asignacion*/
        if(resources.at(pos)->units_asig==0)
            {
                showAvisoInfo(A_UNITS_RESOURCE_0, name+" tiene 0 unidades asignadas del recurso "+name_resource );
                resources.erase(resources.begin()+pos);
            }
    }
}
void Activity::allocateResource(Resource *resource_asign, int value){
    int pos=getPositionResource(resource_asign->getName());           Activity::resource * resource;
    /*El recurso ya esta añadido, cambiamos su valor*/
    if(pos!= -1)
        showAvisoInfo(A_RESOURCE_ALREADY_ADDED, name+" ya tiene asignado "+resource_asign->getName());
    this->getResource(resource_asign->getName())->units_asig+=value;
    else /*Lo asignamos creandolo*/
    {
        resource=new Activity::resource(resource_asign,value);   this->resources.push_back(resource);
    }
}
void Activity::deleteResource(Resource *resource){
    int pos=getPositionResource(resource->getName());
    if(pos== -1) showErrInfo(E_RESOURCE_NOT_ALLOCATED, resource->getName()+" no esta asignado a la actividad "+name );
    else //borra la asignacion del recurso
        resources.erase(resources.begin()+pos);
}
bool Activity::isEnoughResource(std::set<disp_instant, CompareDisponibilities>::iterator disponibilidad){
    /*Comprobar que para cada disponibilidad de un recurso, no se consume mas alla de esa disponibilidad*/
    for(int i=0; i<disponibilidad->resources.size();i++)
    {
        resource * res=getResource(disponibilidad->resources.at(i).resource->getName());
        if(res!=NULL&&res->units_asig>disponibilidad->resources.at(i).units_disp)
            return false;
    }
    return true;
}
bool Activity::isEnoughResource(disp_instant disponibility){
    for(int i=0; i<disponibility.resources.size();i++)
    {
        resource * res=getResource(disponibility.resources.at(i).resource->getName());
        if(res!=NULL&&res->units_asig>disponibility.resources.at(i).units_disp)
            return false;
    }
    return true;
}

```

7.2.3 Exceptions.h

```
#include <string>
#include <QMessageBox>
/*Códigos de errores*/

#define OK 1
#define E_ACTIVITY_NOT_EXIST 2
#define E_RELATION_NOT_EXIST 3
#define E_CYCLE 4
#define E_RELATION_ALREADY_EXIST 5
#define E_ACTIVITY_ALREADY_EXIST 6
#define E_VALUE_RESOURCE 7
#define E_RESOURCE_NOT_ALLOCATED 8
#define A_RESOURCE_ALREADY_ADDED 9
#define E_RESOURCE_NOT_EXIST 10
#define E_RESOURCE_ALREADY_EXIST 11
#define A_UNITS_RESOURCE_0 12
#define E_ALGORITHM_NOT_EXIST_RESOURCES 13
#define E_OPEN_FILE 14
#define E_RESOURCE_MAX_EXCEEDED 15
#define E_DEPENDENCES 16
#define E_FORMAT_FILE 17
#define E_UPDATE_PATH 18

void showError(int code);
void showAvisoInfo(int code, std::string information);
void showErrorInfo(int code, std::string information);
```

7.2.4 Exceptions.cc

```
#include <iostream>
#include "Exceptions.h"

using namespace std;
void showAvisoInfo(int code, std::string information){
    QMessageBox mes;
    switch(code){
        case A_RESOURCE_ALREADY_ADDED:
            mes.warning(0,QString("Aviso"),QString("La actividad ").append(QString::fromStdString(information)).append(QString(" por lo que se modifican las unidades asignadas")));
            break;
        case A_UNITS_RESOURCE_0:
            mes.warning(0,QString("Aviso"),QString("La actividad ").append(QString::fromStdString(information)).append(QString(" por lo que se elimina la asignacion del recurso")));
            break;
        default:
            std::cout<<information<<endl;
            break;
    }
    mes.show();
}
void showErrorInfo(int code, std::string information){
    QMessageBox mes;
    switch(code){

        case E_ACTIVITY_NOT_EXIST:
            mes.information(0,QString("Aviso"),QString("La actividad ").append(QString::fromStdString(information)).append(QString(" no existe")));
            break;

        case E_RELATION_NOT_EXIST:
            mes.information(0,QString("Error"),QString::fromStdString(information));
            break;

        case E_CYCLE:
            mes.information(0,QString("Aviso"),QString("Se produce un ciclo, ya existe la relacion\n").append(QString::fromStdString(information)));
            break;

        case E_RELATION_ALREADY_EXIST:
            mes.information(0,QString("Aviso"),QString("Ya existe la relacion\n").append(QString::fromStdString(information)));
            break;

        case E_ACTIVITY_ALREADY_EXIST:
            mes.information(0,QString("Aviso"),QString("La actividad ").append(QString::fromStdString(information)).append(QString(" ya existe")));
            break;

        case E_VALUE_RESOURCE:
            mes.information(0,QString("Aviso"),QString::fromStdString(information));
            break;
        case E_RESOURCE_NOT_ALLOCATED:
            mes.information(0,QString("Aviso"),QString("El recurso ").append(QString::fromStdString(information)));
            break;

        case E_RESOURCE_NOT_EXIST:
            mes.information(0,QString("Aviso"),QString("El recurso ").append(QString::fromStdString(information)).append(QString(" no existe")));
            break;

        case E_RESOURCE_ALREADY_EXIST:
            mes.information(0,QString("Aviso"),QString("Ya existe el recurso: \n").append(QString::fromStdString(information)));
            break;
    }
}
```

```

        break;

    case E_OPEN_FILE:
        mes.information(0,QString("Aviso"),QString("No se pudo abrir el fichero \n").append(QString::fromStdString(information)));
        break;

    case E_RESOURCE_MAX_EXCEEDED:
        mes.information(0,QString("Aviso"),QString("Se excede el max de unidades del recurso:
\n").append(QString::fromStdString(information)));
        break;

    case E_ALGORITHM_NOT_EXIST_RESOURCES:
        mes.information(0,QString("Aviso"),QString("El algoritmo: ").append(QString::fromStdString(information)).append(QString("\nnecesita que
el proyecto tenga recursos")));
        break;

    case E_DEPENDENCES:
        mes.information(0,QString("Aviso"),QString("No se cumple la dependencia entre: \n").append(QString::fromStdString(information)));
        break;

    case E_FORMAT_FILE:
        mes.information(0,QString("Aviso"),QString("El fichero: \n").append(QString::fromStdString(information)).append(QString("\nNo termina
con .gpi")));
        break;
    case E_UPDATE_PATH:
        mes.information(0,QString("Aviso"),QString("El path del fichero
\n").append(QString(" no se puede modificar mientras esta abierto\n")));
        break;
    default:
        cout<<information<<endl;
        break;
    }
}
}

```

7.2.5 Path.h

```

#ifndef Path_H_
#define Path_H_
class Path{
    int duration;                                // duracion del camino
    std::vector<Activity*>* path;// actividades que forman parte del camino
public:
    Path(std::vector<Activity*> *path);           Path(Path *path);
    Path(Activity* act);                          Path();
    ~Path();
    void copy(Path* path);                      //copia la informacion del camino que se le pasa
    void clear(){ this->path->clear();}          // resesta el camino
    std::vector<Activity*> * getPath(){ return path;} // calcula la duracion del camino segun las duraciones de las actividades que forman parte
    void calculDuration();                         // calcula la duracion del camino segun las duraciones de las actividades que forman parte
    int getDuration(){ return duration;}           // calcula la duracion del camino segun las duraciones de las actividades que forman parte
    void setDuration(int duration){ this->duration=duration; } // establece la duracion del camino
    bool includesActivity(Activity * act);         // comprueba si la actividad forma parte del camino
    std::string toString();                        // devuelve en string el camino con los nombres de las actividades que forman
};

#endif;

```

7.2.6 Path.cc

```

#include "Activity.h"
#include "Path.h"
/*********************************************************************
/* Constructores
*/
Path::Path(std::vector<Activity*> *path_other){
    /*Crear el camino a partir de un conjunto de acitvidades*/
    path= new std::vector <Activity *>;
    for(int i=0; i<path_other->size();i++) { Activity * act=path_other->at(i);
                                                path->push_back(act); }
}
Path::Path(Path *path_other){
    /*Crear el camino a partir de otro dado*/
    path= new std::vector <Activity *>;
    for(int i=0; i<path_other->getPath()->size();i++){ Activity * act=path_other->getPath()->at(i);
                                                path->push_back(act); }
}
Path::Path(Activity* act){ path= new std::vector <Activity *>; path->push_back(act);}
Path::Path(){ path= new std::vector <Activity *>;}
/*Destruktor*/
Path::~Path(){
    delete(path);
}
/*********************************************************************
/* Metodos
*/
void Path:: copy(Path* path_other){
    /*copiar las acitvidades de otro camino*/
    for(int i=0; i<path_other->path->size(); i++)
        /*resetear el camino*/
        path_other->getPath()->clear();
}
void Path::calculDuration(){
}

```

```

/*Calcular la duracion del camino como suma de la
duracion de las actividades que lo componen*/
this->duration=0;
for(int i=0; i<this->path->size();i++)
    duration+=path->at(i)->getTNormal();
}

bool Path::includesActivity(Activity * act){
    /*Comprobar si esa actividad esta dentro del camino*/
    if(act==NULL)      return false;
    for(int i=0; i<this->path->size();i++)      if(path->at(i)->getName()==act->getName())
        return true;
    return false;
}

std::string Path::toString(){
    /*Obtener las actividades que forman el camino en formato string*/
    std::string path_text="";
    for(int i=0; i<this->path->size(); i++) {
        if(i!=0)    path_text.append("-");
        path_text.append(path->at(i)->getName());
    }
    return path_text;
}

```

7.2.7 Probability.h

```

#include <cmath>
#include <errno.h>

/* Coefficients in rational approximations.*/
static const double a[] =
{
    -3.969683028665376e+01, 2.209460984245205e+02, -2.759285104469687e+02,
    1.383577518672690e+02, -3.066479806614716e+01, 2.506628277459239e+00
};

static const double b[] =
{
    -5.447609879822406e+01, 1.61585368580409e+02, -1.556989798598866e+02, 6.680131188771972e+01, -1.328068155288572e+01
};

static const double c[] =
{
    -7.784894002430293e-03, -3.223964580411365e-01, -2.400758277161838e+00,
    -2.549732539343734e+00, 4.374664141464968e+00, 2.938163982698783e+00
};
static const double d[] =
{
    7.784695709041462e-03, 3.224671290700398e-01, 2.445134137142996e+00, 3.754408661907416e+00
};

#define LOW 0.02425
#define HIGH 0.97575
/*********************************************************************
/*Funciones para obtener probabilidades que siguen una distribucion normal */
********************************************************************/

double GetCumulativeDensityNormal(const double x)
{
    const double c0 = 0.2316419; const double c1 = 1.330274429; const double c2 = 1.821255978;
    const double c3 = 1.781477937; const double c4 = 0.356563782; const double c5 = 0.319381530; const double c6 = 0.398942280401;
    const double negative = (x < 0 ? 1.0 : 0.0); const double xPos = (x < 0.0 ? -x : x); const double k = 1.0 / (1.0 + (c0 * xPos));
    const double y1 = (((((c1*k+c2)*k)+c3)*k)+c4)*k; const double y2 = 1.0 - (c6*std::exp(-0.5*xPos*xPos)*y1);
    return ((1.0-negative)*y2) + (negative*(1.0-y2));
}

double GetCumulativeDensityNormal( const double x, const double mean,const double stddev){
    return GetCumulativeDensityNormal( (x - mean) / stddev);
}

/*Inversa de la distribucion normal*/
double ltqnorm(double p){
    double q, r; errno = 0;
    if (p < 0 || p > 1){ errno = EDOM; return 0.0; }
    else if (p == 0) { errno = ERANGE; return -HUGE_VAL /* minus "infinity" */; }
    else if (p == 1) { errno = ERANGE; return HUGE_VAL /* "infinity" */; }
    else if (p < LOW){
        /* Rational approximation for lower region */
        q = sqrt(-2*log(p)); return (((((c[0]*q+c[1])*q+c[2])*q+c[3])*q+c[4])*q+c[5]) / (((d[0]*q+d[1])*q+d[2])*q+d[3])*q+1);
    }
    else if (p > HIGH){
        /* Rational approximation for upper region */
        q = sqrt(-2*log(1-p)); return -(((c[0]*q+c[1])*q+c[2])*q+c[3])*q+c[4])*q+c[5]) / (((d[0]*q+d[1])*q+d[2])*q+d[3])*q+1);
    }
    else{
        /* Rational approximation for central region */
        q = p - 0.5; r = q*q;
        return (((((a[0]*r+a[1])*r+a[2])*r+a[3])*r+a[4])*r+a[5])*q / (((((b[0]*r+b[1])*r+b[2])*r+b[3])*r+b[4])*r+1);
    }
}

double GetCumulativeDensityNormalInv( const double x, const double mean,const double stddev){
    double normal=ltqnorm(x); double result=normal*stddev+mean; return result;
}

```

7.2.8 Project.h

```

#include "Activity.h"
#include "Path.h"
#include <deque>
#include <string>
#include <sstream>
#include <fstream>
using namespace std;
#ifndef PROJECT_H_
#define PROJECT_H_
#define INF 1000000000000
class Project {
    /*ACTIVIDADES*/
    vector<Activity*> activities; // actividades del proyecto
    /*CAMINOS CRITICOS*/
    vector<Path*> *critical_paths; // caminos criticos del proyecto
    /*RECURSOS*/
    vector<Resource *> * resources; // recursos del proyecto
    /*COSTES*/
    float overrun; // sobrecoste
    /*ACTIVIDADES AUXILIARES*/
    Activity * begin; // actividad auxiliar de inicio
    Activity * end; // actividad auxiliar de fin
    /*DEBUG*/
    string report; // nombre del fichero donde guardar la informacion de debug
    string sub_report; // nombre del subfichero segun el algoritmo que se debuguee
    ofstream output; // puntero fichero debug
    bool enabled; // modo debug habilitado o no

private:
    int getPositionActivity(string name); // devuelve la posicion de la actividad dentro del vector de actividades
    int getPositionResource(string name); // devuelve la posicion del recurso dentro del vector de recursos
    bool validateDependences(); // valida que los instantes de tiempo entre relaciones sean correcto, dtodas las sucesoras deben empezar mas tarde que las predecesoras
    bool validateLimitationMaxUnitsAllocated(); // valida que la asignacion de cada uno de los recursos, por dia, no supera el maximo de unidades
    cada dia
        void clear(); // borra los datos calculados del proyecto
    /*CAMINO CRITICO*/
    void extractCriticalPaths( Path* path, Activity * partida ); // extrae los caminos criticos que forman parte del proyecto
    /*NIVELACION RECURSOS*/
    void levelResources(vector<Resource*> *resources) // nivela los recursos de forma que para mejorar debe mejorar la suma de los CV de los recursos
    double calculCostResource(string name_resource); // calcula el coste de la asignacion de ese recurso a lo largo de los dias que dura el proyecto
    /*CALCULO MIN COSTE MIN DURACION*/
    void decrement_days(vector<Path*> *paths,int max_decr_dias,Path* act_to_modif); // actualiza la duracion de las actividades
    float get_min_cost(vector<Path*> paths_a_modif, int n_act, Path * activities_to_modif_pruebas, Path* activities_to_modif_sol); // selecciona la mejor combinacion de actividades que reducen los caminos criticos que no superen el sobrecoste permitido
    Path* select_activities_min_cost(vector<Path*> *paths, float& cost, int &max_decr_days, int cost_searched); // selecciona la mejor combinacion de actividades que reducen los caminos criticos
    Path* select_activities_min_cost(vector<Path*> *paths, float& cost, int &max_decr_days);
    void extractPaths( vector<Path*> *paths, Path * path, Activity* partida ); // obtiene todos los caminos que forman parte del proyecto
    /*LIMITACION RECURSOS*/
    float limitationResourcesSerie(int rule); // limitar recursos con el esquema serie y regla de prioridad rule
    float limitationResourcesParalel(int rule); // limitar recursos con el esquema paralelo y regla de prioridad rule
    float HBRPMultiPasada(); // limitar recursos con multipasada
    float limitationResourcesSerie( deque<Activity*> elegir, string step); // proceso de adelanto/traso (step) de las actividades despues de la limitacion de recursos con esquema serie
    float limitationResourcesParalel( deque<Activity*> elegir, string step); // proceso de adelanto/traso (step) de las actividades despues de la limitacion de recursos con esquema paralelo
    /*DEBUG*/
    void mode_debug(string text); // metodos para mostrar los pasos de los algoritmos si esta habilitado el modo debug
    void mode_debug(string before,vector<Activity*> *path, string after);
    void mode_debug(string before,deque<Activity*> path, string after);
    void mode_debug(string before,vector<disp_resource> disp, int t, string after);
    void mode_debug(string before,set<disp_instant, CompareDisponibilidades> *disponibility, string after);

public:
    Project(); ~Project(void);
    vector<Path*> * getCriticalPaths(){ return critical_paths;};
    int sizeCriticalPaths(){ return critical_paths->size();}; // devuelve los caminos criticos
    vector<Activity*> * getActivitys(){ return activities;};
    int sizeActivities(){ return activities.size();}; // numero de actividades
    vector<Resource *> * getResources(){ return resources;};
    int sizeResources(){ return resources->size();}; // devuelve los recursos
    Activity * getEnd(){ return end; }; // numero de recursos
    Activity * getBegin(){ return begin; }; // actividad de fin
    float getTotalCost(); // devuelve el coste total de proyecto, incluyendo el sobrecoste
    float getOverrun(){ return overrun; }; // devuelve el sobrecoste del proyecto
    /*ACTIVIDADES*/
    void addActivity(string name, int tn, int tp, float cnormal, float cuport); // añade una actividad incluyendo el tiempo tope y coste de oportunidad
    void addActivity(string name, int tn, float cnormal); // añade una actividad
    void modifyActivity(string name, int tn, int tp, float cnormal, float cuport); // modifica la actividad nom
    Activity * getActivity(string name); // devuelve la actividad name
    void deleteActivity(string name); // borra la actividad name
    void updateTMax(); // actualiza el tiempo maximo de las actividades segun el tiempo minimo que tengan
    /*RELACIONES*/
    void addRelation(string orig, string dest); // añade una relacion de orig a dest
    void deleteRelation(string orig, string dest); // borra la relacion de orig a dest
    /*RECURSOS*/
    Resource * getResource(string name); // devuelve el recurso name
    void addResource(string name, int units_max); // añade un recurso
    void modifyResource(string name, int units_max); // modifica el recurso
    void deleteResource(string name); // borra el recurso name
}

```

```

void allocateResourceActivity(string resource, string activity, int units);           // asigna una cantidad de recurso a una actividad
void deleteResourceActivity(string resource, string Activity);                      // quita la asignacion del recurso a la actividad
void incrementUnitsResourceActivity(string resource, string activity, int units); //incrementa el numero de unidades del recurso asignadas a la actividad
void decrementUnitsResourceActivity(string resource, string activity, int units); //decrementa el numero de unidades del recurso asignadas a la actividad

vector<double> getAllocationResourcePerDay(string name);                         // devuelve en el vector las unidades asignadas del recurso cada dia
/*ALGORITMO CAMINO CRITICO*/
void calculCriticalPath();                                                       // algoritmo de calculo de camino critico
/*ALGORITMO DE NIVELACION DE RECURSOS*/
bool levelAllResources();             // nivela todos los recursos permitiendo, de forma que para mejorar debe mejorar la suma de los CV de los recursos
bool levelResources(vector<string> names_rec);                                // nivela los recursos names_Rec independientemente unos de otros
/*ALGORITMO ACKOFF-SASIENI: MINIMO COSTE MINIMA DURACION*/
void calculPathMinTimeMinCost();                                                 // algoritmo minimo coste minima duracion
void calculPathMinTimeMinCost(int cost_searched); // algoritmo min coste min duracion sin que el sobrecoste diario sea mayor a cost_searched
/*ALGORITMO limitacion DE RECURSOS*/
bool limitarResources(int rule, int eschema);                                    // limitacion de recursos con una regla de prioridad y esquema
bool retrasoAdelantoActivities(int eschema_retraso, int eschema_adelanto); // proceso de adelanto/traso despues de la limitacion
/*ANALISIS*/
float getProbabDurationLessThan(int x, double var);                           // devuelve la probabilidad de que el proyecto finalize antes de x dias
float getProbabDurationBetween(int x,int y, double var);                      // probabilidad de que en fin de proyecto este entre x e y
int getDurationByProbability(double x, double var);                          // instante fin del proyecto con una probabilidad maxima x
string analizeDelayTMin(int days, string name_activity);                     // devuelve el informe de realizar el analisis de flexibilidad si se retrasa una actividad
string analizeAnticipateTMin(int days,string name_activity); // devuelve el informe de realizar el analisis de flexibilidad si se adelanta una actividad
/*DEBUG*/
void setReportDebug(string path);                                              // asigna un directorio donde guardar los resultados
bool startDebug(std::string sub_report_chosed);                             // empieza a depurar
void finishDebug();                                                          // termina de depurar
void enableDebug();                                                          // habilita el modo debug
void disableDebug();                                                         // deshabilita el modo debug
};

string num_to_str(float n); // funcion auxiliar para transformar floats en cadenas
#endif

```

7.2.9 Project.cc

```

#include <iostream>
#include <set>
#include <math.h>
#include "Project.h"
#include <algorithm>
#include "Rules.h"
#include <math.h>
#include <queue>
#include <deque>
#include "Probability.h"
#include "Resource.h"
#include "Activity.h"
#include "Path.h"
#include "Exceptions.h"

//funcion auxiliar para transformar los floats a string
string num_to_str(float n) { stringstream s; s << n; return s.str();}
/* **** CONSTRUCTOR ****/
Project::Project(void){
    this->critical_paths= new std::vector<Path*>; this->resources= new std::vector<Resource*>;
    this->begin=new Activity("Inicio",0,0,0); this->end=new Activity("Fin",0,0,0); //modo debug desabilitado
    this->enabled=false;
    this->report="report";
}
/* **** DESCONSTRUTOR ****/
Project::~Project(void)
{
    for(int i=0; i<critical_paths->size();i++)
        delete(critical_paths->at(i));
    for(int i=0; i<resources->size();i++)
        delete(resources->at(i));
    delete(this->resources);
    for(int i=0; i<activities.size();i++)
        delete(activities.at(i));
    delete(end);
    delete(begin);
}
/* **** METODOS ****/
void Project::clear(){
    /*resetear los calculos del proyecto*/
    for(int i=0; i<activities.size();i++) activities.at(i)->clear();
    this->critical_paths->clear(); this->end->clear();
}
/* **** ACTIVIDADES ****/
void Project::modifyActivity(std::string name,int t_normal, int t_tope, float cost_normal, float cost_unit_oportunity){
    /*modificar actividad*/
    Activity* act=getActivity(name);
    if(!act)
        showErrorInfo(E_ACTIVITY_NOT_EXIST, name);
    else
        act->modify(t_normal, t_tope, cost_normal, cost_unit_oportunity);
}

```

```

Activity * Project::getActivity(std::string name){
    Activity * act;
    /*Obtener actividad*/
    if( name==begin->getName() ) return begin;
    if( name==end->getName() ) return end;
    for(int i=0; i<this->activities.size();i++){
        act=(Activity *)this->activities.at(i);
        if(act->getName().compare( name)==0) return activities[i];
    }
    return NULL;
}
int Project::getPositionActivity(std::string name){
    Activity * act;
    /*Posicion del vector actividades donde se encuentra la actividad*/
    for(int i=0; i<this->activities.size();i++){
        act=(Activity *)this->activities.at(i);
        if(act->getName().compare( name)==0) return i;
    }
    return -1; // no existe
}
void Project::deleteActivity(std::string name){
    int pos;
    Activity * activ=getActivity( name );
    if(!activ) showErrInfo(E_ACTIVITY_NOT_EXIST, name);
    else{
        pos=this->getPositionActivity( name );
        /*Borrar relaciones de precedencia*/
        while(activ->getList_Activities_pred()->size()>0){
            this->deleteRelation(activ->getList_Activities_pred()->at(0)->activity->getName(), activ->getName());
        }
        /*Borrar relaciones de sucesion*/
        while(activ->getList_Activities_suc()->size()>0){
            this->deleteRelation(activ->getName(), activ->getList_Activities_suc()->at(0)->activity->getName());
        }
        /*Borrar actividad del proyecto*/
        this->activities.erase(this->activities.begin()+pos);
        delete(activ);
    }
}
void Project::addActivity(std::string name,int tn, int tp, float cnormal, float cuoprt){
    /*Crear y añadir actividad al proyecto*/
    Activity *act;
    if(getActivity(name)) showErrInfo(E_ACTIVITY_ALREADY_EXIST, name);
    else{
        act=new Activity(name,tn, tp, cnormal, cuoprt);
        /*No tiene ninguna relacion aun, asi que se relaciona con inicio y fin */
        this->addRelation(this->begin->getName(), name); this->addRelation(name, this->end->getName());
    }
}
void Project::addActivity(std::string name,int tn, float cnormal){
    Activity *act;
    /*Crear y añadir actividad al proyecto*/
    if(getActivity(name)) showErrInfo(E_ACTIVITY_ALREADY_EXIST, name);
    else{
        act=new Activity(name,tn, cnormal); activities.push_back(act);
        /*No tiene ninguna relacion aun, asi que se relaciona con inicio y fin */
        this->addRelation(this->begin->getName(), name); this->addRelation(name, this->end->getName());
    }
}
void Project::updateTMax(){
    /*limpiar valores anterior de t.max*/
    for(int i=0; i<activities.size();i++) activities.at(i)->setTMax(-1);
    end->setTMax(-1);
    /*Calcular tiempos maximos*/
    begin->setTMax(begin->calculMaxTime());
    /*Calcular Hogura Libre*/
    begin->calculHoguraFree(end->getTMax());
}
/********************************************/
/* RELACIONES */
void Project::addRelation(std::string name_orig, std::string name_dest){
    bool deleteIni=false; bool deleteFin=false;
    Activity* orig=getActivity( name_orig ); Activity* dest=getActivity( name_dest );
    /*Existe las actividades?*/
    if(!orig) showErrInfo(E_ACTIVITY_NOT_EXIST, name_orig);
    else if(!dest) showErrInfo(E_ACTIVITY_NOT_EXIST, name_dest );
    /*Se produce un ciclo?*/
    else if(orig->getPositionRelationPredecesor( name_dest )!= -1 || dest->getPositionRelationSucesor( name_orig )!= -1)
        showErrInfo(E_CYCLE, "desde "+dest->getName()+" a "+orig->getName());
    /*Ya existia esta relacion?*/
    else if(orig->getPositionRelationSucesor( name_dest )!= -1 || dest->getPositionRelationPredecesor( name_orig )!= -1)
        showErrInfo(E_RELATION_ALREADY_EXIST, "desde "+orig->getName()+" a "+dest->getName());
    /*Relacion sobre una misma actividad?*/
    else if( name_orig.compare( name_dest )==0 ) showErrInfo(E_CYCLE, "desde "+dest->getName()+" a "+orig->getName());
    else{
        /*Comprobar si antes de añadir la relacion era uno de las actividades de inicio, relacionandas con la actividad ficticia Inicio*/
        if(dest->predecesorsInitial()) deleteIni=true;
        /*Comprobar si antes de añadir la relacion era una de las actividades de fin, relacionandas con la actividad ficticia Fin*/
        if(orig->sucesorsEnd()) deleteFin=true;
        /*Añadir las relaciones*/
        dest->addPredecesor(orig); orig->addSucesor(dest);
        /*Si se relacionaban con Inicio y Fin, eliminar esa relacion*/
        if(deleteFin) this->deleteRelation( name_orig, end->getName() );
        if(deleteIni) this->deleteRelation( begin->getName(), name_dest );
    }
}

```

```

void Project::deleteRelation(std::string name_orig, std::string name_dest){
    int pos_orig;           int pos_dest;
    Activity* orig=getActivity( name_orig);           Activity* dest=getActivity( name_dest);
    if(!orig)                      showErrorInfo(E_ACTIVITY_NOT_EXIST, name_orig);
    else if(!dest)                  showErrorInfo(E_ACTIVITY_NOT_EXIST, name_dest);
    else{
        pos_orig=dest->getPositionRelationPredecesor( name_orig);           pos_dest=orig->getPositionRelationSucesor( name_dest);
        /*Existe relacion entre ambas actividades?*/
        if(pos_orig== -1)          showErrorInfo(E_RELATION_NOT_EXIST, dest->getName()+" no tiene como predecesor "+ name_orig);
        else if(pos_dest== -1)      showErrorInfo(E_RELATION_NOT_EXIST, orig->getName()+" no tiene como sucesor "+ name_dest);
        else{
            /*Eliminar relacion*/
            dest->getList_Activities_pred()->erase(dest->getList_Activities_pred()->begin() + pos_orig);
            orig->getList_Activities_suc()->erase(orig->getList_Activities_suc()->begin() + pos_dest);
            if(orig->getName().compare(begin->getName())!=0&&dest->getName().compare(end->getName())!=0{
                /*Comprobar si despues de borrar la relacion es una de las actividades de inicio*/
                if(dest->getList_Activities_pred()->size()==0)
                    this->addRelation(begin->getName(),dest->getName()); // añadir relacion con Inicio
                /*Comprobar si despues de borrar la relacion es una de las actividades de fin*/
                if(orig->getList_Activities_suc()->size()==0)
                    this->addRelation(orig->getName(),end->getName()); // añadir relacion con Fin
            }
        }
    }
}
bool Project::validateDependences(){
    bool result=true;
    /*Comprobar relaciones de precedencia entre actividades, solo en que una relacion no se cumpla, todo el proyecto no cumplira con esta validacion */
    for(int i=0; i<activities.size();i++) result*=activities.at(i)->validateDependences(); // AND logica
    return result;
}
float Project::getTotalCost(){
    float cost=0;
    /*Calcula el coste de todo el proyecto como suma del coste de la actividades mas el sobrecoste si existe*/
    for(int i=0; i<activities.size();i++) cost+=activities.at(i)->getCostNormal();
    cost+=overrun; return cost;
}

/*********************************************
/*          RECURSOS
/********************************************/
Resource * Project ::getResource(string name_rec){
    /*Obtener el recurso*/
    for(int i=0; i<this->resources->size();i++)
        if( name_rec==resources->at(i)->getName())
            return resources->at(i);
    return NULL; //no existe
}
bool Project::validateLimitationMaxUnitsAllocated(){
    int tiempo=end->getTMax(); int n_resources=0;
    Activity *act; Activity::resource * rec_asig;
    /*Comprueba que se cumple la limitacion para cada recurso*/
    for(int rec=0; rec<this->resources->size(); rec++) {
        /*Recorremos cada dia*/
        for(int dia=0; dia<tiempo;dia++){
            n_resources=0;
            /*Selecciona las actividades que se estan realizando ese dia*/
            for(int i=0; i<activities.size();i++){
                act=activities.at(i);
                if(act->getTMin()<=dia&&act->getTMin()+act->getTNormal()-1>=dia){
                    rec_asig=act->getResource( resources->at(rec)->getName());
                    if(rec_asig!=NULL) n_resources+=rec_asig->units_asig;
                }
            }
            /*Ese dia se supera el maximo de recursos permitidos?*/
            if(n_resources>resources->at(rec)->getUnitsMax()) return false;
        }
    }
    return true;
}
std::vector<double> Project :: getAllocationResourcePerDay( string name_rec){
    int tiempo=end->getTMax(); double n_resources=0;
    Activity *act; Activity::resource * rec_asig;
    Resource * resource=getResource( name_rec); std::vector<double> asig_resources;
    if(resource==NULL) showErrorInfo(E_RESOURCE_NOT_EXIST, name_rec);
    else{
        /*Recorremos cada dia*/
        for(int dia=0; dia<tiempo;dia++){
            n_resources=0;
            /*Selecciona las actividades que se estan realizando ese dia*/
            for(int i=0; i<activities.size();i++){
                act=activities.at(i);
                if(act->getTMin()<=dia&&act->getTMin()+act->getTNormal()-1>=dia){
                    rec_asig=act->getResource( name_rec);
                    if(rec_asig!=NULL) n_resources+=rec_asig->units_asig;
                }
            }
            /*Guardamos las unidades asignadas ese dia*/
            asig_resources.push_back(n_resources);
        }
    }
    return asig_resources;
}

```

```

int Project ::getPositionResource(string name){
    /*Posicion en el vector de recursos donde esta el recurso dado*/
    for(int i=0; i<this->resources->size();i++)           if( name==resources->at(i)->getName() )           return i;
    return -1; //no existe
}
void Project ::addResource(string name_resource, int units_max ){
    Resource * new_resource;
    /*Crea y añade un recurso al proyecto*/
    if(this->getResource(name_resource)!=NULL)  showErrorInfo(E_RESOURCE_ALREADY_EXIST, name_resource);
    else if(units_max<0)// no puede ser negativo
        showErrorInfo(E_VALUE_RESOURCE, "value incorrecto de units maximas de recurso "+ name_resource);
    else{ new_resource=new Resource( name_resource,units_max); resources->push_back(new_resource); }
}
void Project::deleteResource(string name_rec){
    int pos =getPositionResource( name_rec);     Resource * rec= getResource( name_rec);
    if(pos== -1 || rec==NULL)      showErrorInfo(E_RESOURCE_NOT_EXIST, name_rec);
    else{
        /*Borrar las asignaciones de ese recurso a las actividades*/
        for(int i=0; i<activities.size();i++)
            if(activities.at(i)->getResource(rec->getName())!=NULL) activities.at(i)->deleteResource(rec);
        /*Borrar recurso del proyecto*/
        resources->erase(resources->begin()+pos); delete(rec);
    }
}
void Project::allocateResourceActivity(string name_resource, string name_activity, int value){
    Resource * resource=getResource( name_resource);          Activity * Activity= getActivity( name_activity);
    /*La actividad y es recurso existen?*/
    if(resource==NULL)      showErrorInfo(E_RESOURCE_NOT_EXIST, name_resource);
    else if(Activity==NULL)  showErrorInfo(E_ACTIVITY_NOT_EXIST, name_activity);
    /*Valor valido?*/
    else if(value<0)  showErrorInfo(E_VALUE_RESOURCE, "Se esta intentando agregar un valor no valido al recurso "+ name_resource);
    else if(value>resource->getUnitsMax())
        showErrorInfo(E_VALUE_RESOURCE, "En la asignacion se esta excediendo el maximo del recurso "+ name_resource);
    else /*Asignar recurso*/
        Activity->allocateResource(resource,value);
}
void Project::deleteResourceActivity(string name_resource, string name_activity){
    Resource * resource=getResource( name_resource);          Activity * activity= getActivity( name_activity);
    /*Borrar asignacion del recurso a la actividad*/
    if(resource==NULL)      showErrorInfo(E_RESOURCE_NOT_EXIST, name_resource);
    else if(activity==NULL)  showErrorInfo(E_ACTIVITY_NOT_EXIST, name_activity);
    else activity->deleteResource(resource);
}
void Project:: modifyResource(string name_resource, int units_max){
    Resource * resource=getResource( name_resource);
    if(resource==NULL)      showErrorInfo(E_RESOURCE_NOT_EXIST, name_resource);
    else if(units_max<0)/*Valor valido?*/
        showErrorInfo(E_VALUE_RESOURCE, "valor incorrecto de unidades maximas de recurso "+ name_resource);
    else if(units_max<resource->getUnitsMax())
        /*si el numero de unidades maximas que se quiere asignar es menor al que tenia, solo se podra modificar si el recurso no esta asignado a ninguna actividad*/
        for(int i=0; i< activities.size();i++)
            if(activities.at(i)->getResource(name_resource)!=NULL)
                showErrorInfo(E_VALUE_RESOURCE, "No se puede reducir el numero de unidades si el recurso "+ name_resource+" esta asignado");
            break;
        }
    resource->setUnitsMax(units_max);
}
void Project::incrementUnitsResourceActivity(string name_resource, string name_activity, int units){
    Resource * resource=getResource( name_resource);          Activity * activity= getActivity( name_activity);
    /*Incrementa las unidades asignadas de un recurso a una actividad*/
    /*Existe el recurso y la actividad?*/
    if(resource==NULL) showErrorInfo(E_RESOURCE_NOT_EXIST, name_resource);
    else if(activity==NULL)  showErrorInfo(E_ACTIVITY_NOT_EXIST, name_activity);
    else if(units<0) /*Valor valido?*/
        showErrorInfo(E_VALUE_RESOURCE, "Se esta intentando asignar un valor no valido al recurso "+ name_resource);
    else activity->incrementResource(units, name_resource);
}
void Project::decrementUnitsResourceActivity(string name_resource, string name_activity, int units){
    Resource * resource=getResource( name_resource);          Activity * activity= getActivity( name_activity);
    /*Decrementa las unidades asignadas de un recurso a una actividad*/
    /*Existe el recurso y la actividad?*/
    if(resource==NULL)      showErrorInfo(E_RESOURCE_NOT_EXIST, name_resource);
    else if(activity==NULL)  showErrorInfo(E_ACTIVITY_NOT_EXIST, name_activity);
    else if(units<0) /*valor valido?*/
        showErrorInfo(E_VALUE_RESOURCE, "Se esta intentando disminuir un valor no valido al recurso "+ name_resource);
    else activity->decrementResource(units,resource->getName());
}

```

```

/*
***** ALGORITMO CAMINO CRITICO *****
*/
void Project::extractCriticalPaths(Path * path, Activity* partida ){
    Activity * act;
    /*Ultima actividad, se añade el camino critico calculado*/
    if(partida->sucesorsEnd()){
        Path * finished_path=new Path(path);           critical_paths->push_back(finished_path);
        path->clear();                            // lo reseteamos para poder usarlo
    }else {
        for(int i=0; i<partida->getList_Activities_suc()->size(); i++){
            act=partida->getList_Activities_suc()->at(i)->activity;
            /*La actividad es critica?*/
            if(act->getHTotal()==0){
                /*Guardamos el camino antes de añadirle la actividad*/
                Path * path2=new Path(path);           path->getPath()->push_back(act);
                /*Se añade y seguimos examinando su sucesora*/
                extractCriticalPaths(path, act);
                /*Cuando terminamos de examinar el anterior camino, habiendo añadido la sucesora,
                restauramos el camino que teniamos hasta entonces (sin la sucesora) y seguimos buscando entre las otras sucesoras*/
                path->copy(path2);                   delete(path2);
            }
        }
        path->clear();
    }
}
void Project::calculCriticalPath(){
    Activity *act;
    /*Limpiar*/
    clear();
    /*Calcular tiempos minimos*/
    end->setTMin(end->calculMinTime());
    /*Calcular tiempos maximos*/
    begin->setTMax(begin->calculMaxTime());
    /*depurar*/
    mode_debug("\n----- CAMINO CRITICO ----- \n");
    for(int i=0; i<this->activities.size();i++){
        act=activities.at(i);
        mode_debug("Actividad:" +act->getName() + " Tmin:" +num_to_str(act->getTMin())+ " Tmax:" +num_to_str(act->getTMax())+"\n");
    }
    /*Calcular Hogura Libre*/
    begin->calculHolguraFree(end->getTMax());
    /*Calcular caminos criticos*/
    Path * path= new Path();           extractCriticalPaths(path, this->begin);
}
/*
***** ANALISIS *****
*/
float Project::getProbabDurationLessThan(int x, double var){
    double media=end->getTMax();   double var_max=0;           double var_temp=0;
    /*Valido?*/
    if(x<0)          return 0;
    if(var<0)          return 0;
    /*Probabilidad de que el proyecto termine antes de x, segun la varianza*/
    return GetCumulativeDensityNormal( x,media,sqrt(var))*100;
}
float Project::getProbabDurationBetween(int x,int y, double var){
    double media=end->getTMax();
    /*Valido?*/
    if(x<0 || y>0)      return 0;
    if(var<0)          return 0;
    /*Probabilidad de que el proyecto termine entre x e y segun la varianza*/
    float norm_x=GetCumulativeDensityNormal( x,media,sqrt(var));           float norm_y=GetCumulativeDensityNormal( y,media,sqrt(var));
    return (norm_y-norm_x)*100;
}
int Project::getDurationByProbability(double x, double var){
    double media=end->getTMax();
    /*Valido?*/
    if(x<0 || x>1)      return 0;
    if(var<0)          return 0;
    /*Duracion del proyecto con una probabilidad x y una varianza dada*/
    return GetCumulativeDensityNormalInv( x, media,sqrt(var));
}
string Project::analyzeDelayTMin(int days, string name_activity){
    int hlr;           int n_resources=0;
    Activity * activity= getActivity( name_activity);  Activity * act;
    Resource * rec;   Activity::resource * rec_asig;
    string inform="";  bool exceded_limitation=false;
    /*ANALISIS FLEXIBILIDAD*/
    if(activity==NULL) {           showErrorInfo(E_ACTIVITY_NOT_EXIST, name_activity);  return "";    }
    hlr=activity->getHLR();
    /*Valido?*/
    if(days<0)          return "Valor de dias no valido\n";
    /*Adelanta la actividad mas de instante de inicio del proyecto?*/
    if(days>activity->getTMin())      return "No se puede adelantar la actividad mas alla del instante de inicio de proyecto\n";
    /*Factible?*/
    if(days>hlr)          return "Retraso no factible, aumentaria la duracion del proyecto\n";
    /*Como se ve afectada la limitacion?*/
    /*Para los nuevos instantes que se realizaria la actividad*/
    for(int d=activity->getTMin()+days; d<=activity->getTMin()+days+activity->getTNormal();d++){
        /*Comprobar todos los recursos*/
        for(int r=0; r<activity->getResources().size();r++){
            rec=activity->getResources().at(r)->resource_asig;           n_resources=0;
            /*Selecciona las actividades que se estan realizando ese dia, sin incluir la actividad a analizar*/
            for(int i=0; i<activities.size();i++){

```

```

act=activities.at(i);
if(act->getTMin()<=d&&act->getTNormal()-1>=d && act->getName()!=activity->getName()) {
    rec_asig=act->getResource( rec->getName());
    if(rec_asig!=NULL) n_resources+=rec_asig->units_asig;
}
}
/*El consumo de recursos mas el consumo que haria esa actividad si se realizase en ese instante, supera el limite de recursos?*/
if(n_resources+activity->getResource(rec->getName())->units_asig>rec->getUnitsMax()) {
    /*Guardamos todos los analisis para cada recurso*/
    if(!exceeded_limitation)
        inform.append("La actividad se podria retrasar sin repercutir en la duracion del proyecto, pero:\n");
        inform.append("Excederia el maximo de recurso "+ rec->getName()+" en "+num_to_str(n_resources+activity->getResource(rec->getName())->units_asig-rec->getUnitsMax())+" unidades en el instante "+ num_to_str(d)+"\n");
        exceeded_limitation=true;
}
}
if(!exceeded_limitation) return "Retraso factible, no se modificaria la duracion del proyecto y se seguiria cumpliendo la limitacion de recursos\n";
else return inform;
}

string Project::analyzeAnticipateTMin(int days, string name_activity){
int hla; int n_resources=0;
Activity * activity= getActivity( name_activity); Activity * act_pred, *act;
Resource * rec; Activity::resource * rec_asig;
string pred_problem="", string inform="", bool exceeded_limitation=false;
/*ANALISIS FLEXIBILIDAD*/
if(activity==NULL) showErrorInfo(E_ACTIVITY_NOT_EXIST, name_activity); return "";
hla=activity->getHLA();
/*Valido?*/
if(days<0) return "Valor de dias no valido\n";
/*No se puede adelantar mas alla de el inicio del proyecto*/
if(days>activity->getTMin()) return "No se puede adelantar la actividad mas alla del instante de inicio del proyecto\n";
/*Es factible?*/
if(days>hla){
    if(activity->predecessorsInitial()) return "Adelanto no factible, es una actividad de inicio\n";
    /*Que predecesoras condicionan su inicio temprano*/
    for(int i=0; i<activity->getList_Activities_pred()->size();i++){
        act_pred=activity->getList_Activities_pred()->at(i)->activity;
        if(act_pred->getTMin()-act_pred->getTNormal()->activity->getTMin()-days{
            if(pred_problem.size()>0) pred_problem.append("-");
            pred_problem.append(act_pred->getName());
        }
    }
    inform.append("Adelanto no factible, no se cumpliran las dependencias con las predecesoras ").append(pred_problem).append("\n");
    return inform;
}
/*Como se ve afectada la limitacion?*/
/*Para los nuevos instantes que se realizaria la actividad*/
for(int d=activity->getTMin()-days; d<=activity->getTMin()-days+activity->getTNormal();d++) {
    /*Comprobar todos los recursos*/
    for(int r=0; r<activity->getResources().size();r++){
        rec=activity->getResources().at(r)->resource_asig; n_resources=0;
        /*Selecciona las actividades que se estan realizando ese dia, sin incluir la actividad a analizar*/
        for(int i=0; i<activities.size();i++){
            act=activities.at(i);
            if(act->getTMin()<=d&&act->getTNormal()-1>=d && act->getName()!=activity->getName()) {
                rec_asig=act->getResource( rec->getName());
                if(rec_asig!=NULL) n_resources+=rec_asig->units_asig;
            }
        }
        /*El consumo de recursos mas el consumo que haria esa actividad si se realizase en ese instante, supera el limite de recursos?*/
        if(n_resources+activity->getResource(rec->getName())->units_asig>rec->getUnitsMax()) {
            /*Guardamos todos los analisis para cada recurso*/
            if(!exceeded_limitation)
                inform.append("La actividad se podria retrasar sin repercutir en la duracion del proyecto, pero:\n");
                inform.append("Excederia el maximo de recurso "+ rec->getName()+" en "+num_to_str(n_resources+activity->getResource(rec->getName())->units_asig-rec->getUnitsMax())+" unidades el instante "+ num_to_str(d)+"\n");
                exceeded_limitation=true;
        }
    }
}
if(!exceeded_limitation) return "Adelanto factible, se cumpliran las dependencias del proyecto y se seguiria cumpliendo la limitacion de recursos\n";
else return inform;
}

/*****************************************/
/* DEBUG */
/*****************************************/
bool Project::startDebug(std::string sub_report_chosed){
    /*Habilitado el modo?*/
    if(!enabled) return false;
    /*Abre el fichero donde guardar la informacion de debug*/
    sub_report=sub_report_chosed; std::string rep=report+"_"+sub_report+".debug"; output.open(rep.c_str());
    if(!output.is_open()) showErrorInfo(E_OPEN_FILE,report+"_"+sub_report+".debug"); return false;
    return true;
}

/*Metodos para imprimir la informacion de debug*/
void Project::mode_debug(std::string text){
    if(enabled) return;
    if(!output.is_open()) return;
    output<text;
}

void Project::mode_debug(string before,std::vector<Activity*> * paths, string after){
    if(enabled) return;
}

```

```

        if(!output.is_open())           return;
        output<<before;
        for(int j=0; j<paths->size();j++)    output<<paths->at(j)->getName()<<" ";
        output<<after;
    }
void Project::mode_debug(string before,std::deque<Activity*> paths, string after){
    if(!enabled)                      return;
    if(!output.is_open())              return;
    output<<before;
    for(int j=0; j<paths.size();j++)   output<<paths.at(j)->getName()<<" ";
    output<<after;
}
void Project::mode_debug(string before,std::vector<disp_resource> disp, int t, string after){
    if(!enabled)                      return;
    if(!output.is_open())              return;
    output<<before;      output<<"disp"<<t<<"";
    for(int j=0; j<disp.size();j++){   output<<" "<<disp.at(j).resource->getName()<<"="<<disp.at(j).units_disp;
    output<<"";          output<<after;
}
void Project::mode_debug(string before,std::set<disp_instant, CompareDisponibilities> *disponibility, string after){
    set<disp_instant, CompareDisponibilities>::iterator it=disponibility->begin();
    //output<<before;
    while(disponibility->end()!=it){   mode_debug(before,it->resources,it->instant,after);     it++;
    output<<after;
}
void Project::finishDebug(){
    /*Cerrar el fichero de debug*/
    if(!enabled)                      return;
    if(output.is_open())              output.close();
}
void Project::enableDebug(){
    /*Habilitar el modo*/
    this->enabled=true;
}
void Project::disableDebug(){
    /*Desabilitar el debug*/
    if(enabled && output.is_open())  output.close();
    enabled=false;
}
void Project::setReportDebug(string path){
    if(output.is_open())              showErrorInfo(E_UPDATE_PATH,"debug");
    else                            this->report=path.append("report");
}

```

7.2.10 NivelationResources.cc

```
#include "Project.h"
#include <algorithm>
#include <math.h>
#include "Exceptions.h"
#include "Rules.h"
/*********************************************************************
*-----Algoritmo de nivelacion de recursos-----
*****
double Project :: calculCostResource( string name_resource){
    int time=end->getTMax();    double charge=0;           double cost_day=0;
    Activity::resource * rec_asig;           double average=0;           double s=0;   double t_min;
    int units_max=this->getResource(name_resource)->getUnitsMax();
    /*CALCULAR LA MEDIA PARA ESE RECURSO*/
    /*Recorremos cada dia del proyecto*/
    for(int day=0; day<time;day++){
        charge=0;
        /* Selecciona las actividades que se estan realizando ese dia*/
        for(int i=0; i<activities.size();i++){
            t_min=activities.at(i)->getTMin();
            if(t_min<=day&&t_min+activities.at(i)->getTNormal()-1>=day){
                rec_asig=activities.at(i)->getResource(name_resource);
                if(rec_asig!=NULL){
                    average+=rec_asig->units_asig;           charge+=rec_asig->units_asig;
                }
            }
        }
        /* Prueba factibilidad, se cumplira la limitacion de recursos?*/
        if(charge>units_max) {
            mode_debug("Se supera la limitacion de recursos\n");
            return INF; // como no es factible, coste infinito
        }
    }
    /* Media */
    average=average/time;
    /*CALCULO COEFICIENTE VARIACION */ /*Recorremos cada dia del proyecto*/
    for(int day=0; day<time;day++){
        /* S */
        cost_day=0;
        /* Selecciona las actividades que se estan realizando ese dia*/
        for(int i=0; i<activities.size();i++){
            t_min=activities.at(i)->getTMin();
            if(t_min<=day&&t_min+activities.at(i)->getTNormal()-1>=day){
                rec_asig=activities.at(i)->getResource(name_resource);  if(rec_asig!=NULL) cost_day+=rec_asig->units_asig;
            }
        }
        s+=(cost_day-average)*(cost_day-average);
    }
    s=s/time;   s=sqrt(s);   return (s/average);
}
bool Project::levelResources(std::vector<Resource*> *resources_level){
    Activity*act; double charges_origin=0;   double charges_calculated=0;
    std::deque<Activity*> ordered_activities; int max_instant;      int t_best_begin;      int instant;
    if(resources_level->size()==0){ showErrorInfo(E_ALGORITHM_NOT_EXIST_RESOURCES, "nivelacion de recursos"); return false; }
    /*Ordenar las actividades en orden decreciente segun el instante de finalizacion mas temprano*/
    for(int i=0; i<activities.size();i++) ordered_activities.insert(ordered_activities, activities.at(i),MIN_EFT_FIFO);
    reverse(ordered_activities.begin(),ordered_activities.end());
    mode_debug("***** NIVELACION RECURSOS *****\n\n");
    mode_debug("Orden de las actividades a mover: ",ordered_activities, "\n");
    /*Carga disposicion original*/
    for(int i=0; i<resources_level->size();i++) charges_origin+=calculCostResource(resources_level->at(i)->getName());
    mode_debug("\nCoriginal = "+num_to_str(charges_origin)+"\n\n");
    /*Calcula el instante de inicio de las actividades con menor carga de recursos*/
    for(int i=0; i<ordered_activities.size();i++) {
        act=ordered_activities.at(i);          t_best_begin=act->getTMin();           //parte de intante de inicio mas temprano
        /*Modificar el dia de comienzo de las actividades no criticas, consumiendo la holgura libre*/
        instant=act->getTMin()+1;             max_instant=act->getTMin()+act->getHFree();
        mode_debug(act->getName()+" TMin Original="+num_to_str(instant)+" H.Libre="+num_to_str(act->getHFree())+":\n");
        /*Mientras haya holgura libre*/
        while(instant<max_instant) {
            /*Iniciaza*/
            act->setTMin(instant);           charges_calculated=0;
            /*Calculamos la carga para cada recurso con el CV*/
            for(int r=0; r<resources_level->size();r++) charges_calculated+=calculCostResource(resources_level->at(r)->getName());
            mode_debug("Movemos la actividad "+act->getName()+" al instante "+num_to_str(instant)+" con un
CV="+num_to_str(charges_calculated)+"\n");
            /*La suma total de las cargas de los recursos mejora respecto a la original?*/
            if(charges_calculated<charges_origin){
                charges_origin=charges_calculated;       t_best_begin=instant;
                mode_debug("La nueva disposicion mejora la anterior\n");
            }
            instant++;
        }
        /*Asignamos la mejora*/
        act->setTMin(t_best_begin);
        mode_debug(" "+act->getName()+" se realiza en "+num_to_str(t_best_begin)+"\n-----\n");
    }
    /*Actualizar t.max y holguras*/
    updateTMax();
    /*Validar si despues de aplicar el algoritmo, se siguen cumpliendo las dependencias y la limitacion de recursos*/
    if(!validateDependencies()) exit(-1);
    if(!validateLimitationMaxUnitsAllocated()) exit(-1);
    return true;
}
```

```

bool Project::levelAllResources(){
    /*Para nivelar todos los recursos le pasamos a la funcion de nivelacion todos los recursos del proyecto, juntito con el empeoramiento permitido*/
    return levelResources( resources);
}
bool Project::levelResources(std::vector<std::string> names_rec){
    std::vector<Resource *> *resources= new std::vector<Resource *>;      Resource * resource;
    /*Transformamos los nombres de los recursos en recursos y los guardamos*/
    for(int i=0; <names_rec.size();i++){
        resource->this->getResource(names_rec.at(i));
        if(resource==NULL){           showErrorInfo(E_RESOURCE_NOT_EXIST, names_rec.at(i));           return false; }
        Else             resources->push_back(resource);
    }
    /*Pasamos a la funcion los recursos que queremos nivelar*/
    return levelResources(resources);
    delete(resources);
}

```

7.2.11 Limitación de Recursos

```

#include "Project.h"
#include <algorithm>
#include <math.h>
#include "Exceptions.h"
#include "Rules.h"
/*********************************************************************
/*ALGORITMO LIMITACION RECURSOS
*****
disp_instant createDisponibility(int t, Activity * act, set<disp_instant, CompareDisponibilities>::iterator disponibility ){
    disp_resource resource_in_t;          disp_instant resources_in_t; Activity::resource * rec;
    /*Para todos los recursos, añadirle a las unidades del instante anterior, las unidades que devolveria esta actividad*/
    for(int i=0; <disponibility->resources.size();i++) {
        rec=>act->getResource(disponibility->resources.at(i).resource->getName());
        resource_in_t.resource=disponibility->resources.at(i).resource;
        if(rec!=NULL) // la actividad consumia este recurso
            resource_in_t.units_disp=rec->units_asig+ disponibility->resources.at(i).units_disp;
        else
            resource_in_t.units_disp=disponibility->resources.at(i).units_disp;
        resources_in_t.resources.push_back(resource_in_t);
    }
    resources_in_t.instant=t;   return resources_in_t;
}
int get_sequencing_instant(std::set<disp_instant, CompareDisponibilities> *disponibility, Activity * act){
    int t_earliest=0;          set<disp_instant, CompareDisponibilities>::iterator it=disponibility->begin();           Activity * act_pred;
    /*Encontrar el instant mas pronto que puede empezar*/
    if(!act->predecesorsInitial()) // si no es una actividad de comienzo
    {
        /*Calcular cual de sus predecesoras termina mas tarde, que seria el instantane mas pronto que podria empezar la actividad*/
        for(int pred=0;pred<act->getList_Activities_pred()->size();pred++){
            act_pred=>act->getList_Activities_pred()->at(pred)->activity;
            if(act_pred->getTSecuenciacion()>act_pred->getTNormal()>t_earliest)
                t_earliest=act_pred->getTSecuenciacion()+act_pred->getTNormal();
        }
    }
    /*Situarse en el instante que mas pronto que puede empezar donde haya disponibilidad de recursos*/
    while(disponibility->end()!=it&& it->instant<t_earliest) it++; // asi tinc doute de si shauria de fer it-- si no es igual al instant
    /*Comprobar si todo el tiempo que dura la actividad desde que se inicia hay suficientes recursos*/
    while(disponibility->end()!=it&&it->instant<t_earliest+act->getTNormal() ){
        /*En la lista no hay mas disponibilidades, por lo que no hay impedimento para que comienze en el t_earliest*/
        if(disponibility->end()==it) return t_earliest; // t mas pronto
        /*Si no hay suficiente recurso en ese instante, la actividad comenzara el siguiente instante,
        siempre que este instante lo permita, cosa que se comprueba durante la siguiente interaccion*/
        if(act->isEnoughResource(it)){           it++;           t_earliest=it->instant;
        }else
            it++;
    }
    return t_earliest;
}
void update_disponibility(std::set<disp_instant, CompareDisponibilities> *disponibility, Activity *act, int t){
    int i=0;          Activity::resource *rec;      set<disp_instant, CompareDisponibilities>::iterator it=disponibility->begin();
    /*Dentro de la lista de disponibilidades, situarse en el momento que empezaria a consumir recursos*/
    while(disponibility->end()!=it&& it->instant<t){
        /*Consumir los recursos de los diferentes instantes de disponibilidad, hasta que termine la actividad*/
        while(disponibility->end()!=it&& it->instant<+act->getTNormal()){
            /*Actualizar disponibilidades de todos los recursos*/
            for(int k=0; k<it->resources.size();k++){
                rec=>act->getResource(it->resources.at(k).resource->getName());
                if(rec!=NULL)           //la actividad no tiene asignado ese recurso
                    it->resources.at(k).units_disp-=rec->units_asig;
            }
            it++;
        }
        /*Si el tiempo que dura la actividad va mas alla de los instantes de disponibilidad guardados,
        añadir una nueva disponibilidad, que sera el resultado de devolver a la disponibilidad anterior los recursos consumidos*/
        if(it->instant!=t+act->getTNormal())      disponibility->insert(createDisponibility(t+act->getTNormal(), act, it));
    }
}

```

```

bool all_predecesors_examined(Activity* act, std::vector<Activity*> examined){
    int j;
    /*Si es una actividad de inicio no hay problema*/
    if(act->predecesorsInitial()) return true;
    /*Examinar todas las predecesoras y comprobar que estan vistas*/
    for(int i=0; i<act->getList_Activities_pred()->size();i++) {
        /*Esta ya examinada*/
        for(j=0; j<examined.size();j++){
            if(act->getList_Activities_pred()->at(i)->activity->getName()==examined.at(j)->getName())
                break;
        }
        /*Alguna no se ha examinado aun*/
        if(j==examined.size())
            return false;
    }
    return true;
}

vector<procesing_activity> insert_proces(vector<procesing_activity> procesing, procesing_activity act ){
    int i;
    /*Ordena las actividades procesandose crecientemente segun su instante de finalizacion de proceso */
    if(procesing.size()==0){    procesing.push_back(act);    return procesing;    }
    for(i=0; i<procesing.size();i++){
        if(act.instant <procesing.at(i).instant){
            procesing.insert(procesing.begin()+i,act); //inserta el proceso en la posicion i
            return procesing;
        }
    }
    procesing.push_back(act);    return procesing;
}

float Project::limitationResourcesSerie(int rule){
    int n_act_to_process=activities.size();
    std::set<disp_instant, CompareDisponibilidades> *disponibility=new set<disp_instant, CompareDisponibilidades>;
    std::deque<Activity*> selected;    std::vector<Activity*> examined;
    int t=0;    float t_max=0;    disp_resource resource_in_t;    disp_instant resources_in_t;
    Activity * act;
    if(this->resources->size()==0){
        showErrorInfo(E_ALGORITHM_NOT_EXIST_RESOURCES, " limitacion de recursos en serie");
        return INF;
    }
    /*INICIALIZACION*/
    mode_debug("\n*****SERIE*****\n");
    /*Disponibilidad inicial, t=0*/
    for(int i=0; i<this->resources->size();i++){
        resource_in_t.resource=resources->at(i);    resource_in_t.units_disp=resources->at(i)->getUnitsMax();
        resources_in_t.resources.push_back(resource_in_t);
    }
    resources_in_t.instant=0;    disponibility->insert(resources_in_t);
    /*Actividades elegibles son las actividades de inicio*/
    for(int i=0; i<begin->getList_Activities_suc()->size();i++)    selected=insert(selected, begin->getList_Activities_suc()->at(i)->activity, rule);
    mode_debug(" disponibility,\n");
    /*Se examinan todas las actividades*/
    while(n_act_to_process>0) {
        mode_debug("\n\nITERACION _____\n");
        /*Actividad a secuenciar*/
        act=selected.front();    selected.pop_front();
        /*calcular instante en que se secuenciar*/
        t= get_sequencing_instant(disponibility, act);
        /*Secuenciar*/
        act->setTSecuencion(t);    examined.push_back(act);
        mode_debug(" La actividad "+act->getName()+" se secuencia en "+num_to_str(t)+"\n");
        /*Guardamos cuel es el instante de fin del proyecto, que será cuando finalize la actividad que termina mas tarde*/
        if(t>act->getTNormal()>t_max) t_max=t+act->getTNormal();
        /*actualizar elegibles con las sucesoras de la actividad secuenciada, solo si todas las predecesoras de cada sucesora han sido vistas*/
        for(int i=0; i< act->getList_Activities_suc()->size();i++)
            if(all_predecesors_examined(act->getList_Activities_suc()->at(i)->activity, examined)&&!act->sucesorsEnd())
                selected=insert(selected,act->getList_Activities_suc()->at(i)->activity, rule);

        /*actualizar disponibilidad*/
        update_disponibility(disponibility, act, t);
        mode_debug(" ,disponibility,\n");
        n_act_to_process--;
    }
    delete(disponibility);
    return t_max;
}

***** Paralel*****
float Project:: limitationResourcesParallel(int rule){
{
    std::vector<procesing_activity> procesing;    disp_instant disponibility;    std::deque<Activity*> selected;
    std::vector<Activity*> examined;    int t=0;    int j=0;
    disp_resource resource_in_t;    Activity * act; Activity::resource * rec;
    procesing_activity procesing_act, act_proces_fin;
    if(this->resources->size()==0){
        showErrorInfo(E_ALGORITHM_NOT_EXIST_RESOURCES, " limitacion de recursos en paralelo");
        return INF;
    }
    /*Incializacion*/
    mode_debug("\n*****PARALELO*****\n");
    /*Disponibilidad inicial, t=0*/
    for(int i=0; i<this->resources->size();i++) {
        resource_in_t.resource=resources->at(i);    resource_in_t.units_disp=resources->at(i)->getUnitsMax();
        disponibility.resources.push_back(resource_in_t);
    }
    disponibility.instant=0;
    /*Actividades elegibles son las actividades de inicio*/
    for(int i=0; i< begin->getList_Activities_suc()->size();i++) selected=insert(selected, begin->getList_Activities_suc()->at(i)->activity, rule);

    /*Secuenciar actividades*/
}

```

```

while(procesing.size()!=0 || t==0){
    /*Hay alguna actividad procesandose*/
    if(procesing.size()!=0){
        act_proces_fin=procesing.front(); // obtiene la actividad que termina antes de procesarse
        t=act_proces_fin.instant;
    }
    mode_debug("\n\nITERACION __T="+num_to_str(t)+"\n");
    /*actualizar el procesamiento, todas borrar las actividades que terminan en t y añadirlas a vistas*/
    while(procesing.size()>0&&act_proces_fin.instant==t){
        mode_debug(" La actividad "+act_proces_fin.act->getName()+" termina de procesarse\n");
        /*Añadirlas a vistas*/
        examined.push_back(act_proces_fin.act); procesing.erase(procesing.begin());
        /*Restaurar la disponibilidad*/
        for(int k=0; k<disponibility.resources.size();k++){
            rec=act_proces_fin.act->getResource(disponibility.resources.at(k).resource->getName());
            if(rec!=NULL) //no tiene asignado ese recurso
                disponibility.resources.at(k).units_disp+=rec->units_asig;
        }
        /*actualizar elegibles con las sucesoras de la actividad secuenciada, solo si todas las predecesoras de cada sucesora han sido vistas*/
        for(int i=0; i<act_proces_fin.act->getList_Activities_suc()->size();i++)
            if(all_predecesors_examined(act_proces_fin.act->getList_Activities_suc()->at(i)->activity,
            examined)&&!act_proces_fin.act->sucesorsEnd())
                selected=insert(selected,act_proces_fin.act->getList_Activities_suc()->at(i)->activity, rule);

        /*Siguiente procesandose*/
        if(procesing.size()>0) act_proces_fin=procesing.front();
    }
    mode_debug(" ,disponibility.resources,t,"\""\n"); mode_debug(" elegir=","selected,""\n");
    /*Comprobar que actividades elegibles se pueden secuenciar*/
    j=0;
    while(!selected.empty()&&j<selected.size()){
        /*Hay suficientes recursos para secuenciar esa actividad?*/
        if(selected.at(j)->isEnoughResource(disponibility)){
            /*Pasa a procesarse*/
            act=selected.at(j); procesing_act.act=act;
            procesing_act.instant=t+act->getTNormal(); procesing=insert_proces(procesing,procesing_act);
            act->setTSecuencionacion(t); selected.erase(selected.begin()+j);
            mode_debug(" Se secuencia "+act->getName()+" en "+num_to_str(t)+" y termina en
"+num_to_str(procesing_act.instant)+"\n");

            /*Consumo los recursos disponibles*/
            for(int k=0; k<disponibility.resources.size();k++){
                rec=act->getResource(disponibility.resources.at(k).resource->getName());
                if(rec!=NULL) //no tiene asignacion de ese recurso
                    disponibility.resources.at(k).units_disp-=rec->units_asig;
            }
            j++;
        }
        mode_debug(" ,disponibility.resources, t,"\""\n");
    }
    return t;
}
float Project::HBRPMultiPasada(){
    float best=INF; int best_rule=MIN_LFT_FIFO; int best_eschema=_SERIE; float temporal;
    if(this->resources==NULL) {
        showErrorInfo(E_ALGORITHM_NOT_EXIST_RESOURCES, "HBRP multi pasada");
        return INF;
    }
    mode_debug("#####MULTI PASADA #####\n\n");
    /*Para cada regla*/
    for(int rule=MIN_LFT_FIFO; rule<=MAX_NSUC_FIFO;rule++){
        /*Resultado con el esquema paralelo*/
        temporal=this->limitationResourcesParallel(rule);
        if(best>temporal) //es best, update
        {
            best=temporal; best_rule=rule; best_eschema=_PARALEL;
        }
        /*Resultado con el esquema secuencial*/
        temporal=this->limitationResourcesSerie(rule);
        if(best>temporal)// es mejor, actualizar
        {
            best=temporal; best_rule=rule; best_eschema=_SERIE;
        }
    }
    mode_debug("Mejor esquema y mejor regla con coste "+num_to_str(best)+" es la siguiente\n");
    if(best==INF) return INF; //no se ha podido aplicar el algoritmo
    /*Una vez se sepa cual es el mejor resultado, aplicarlo*/
    if(best_eschema==_SERIE) return limitationResourcesSerie(best_rule);
    else return limitationResourcesParallel(best_rule);
}
bool Project::limitarResources(int rule, int eschema){
    float t;
    if(eschema==_SERIE) t=limitationResourcesSerie(rule);
    else if(eschema==_PARALEL) t=limitationResourcesParallel(rule);
    else t=HBRPMultiPasada();
    if(t==INF) return false; //no se ha podido aplicar el algoritmo
    for(int i=0; i<activities.size();i++) activities.at(i)->setTMin(activities.at(i)->getTSecuencionacion());
    end->setTMin(t);
    this->updateTMax();
    if(validateDependences()) exit(-1);
    if(validateLimitationMaxUnitsAllocated()) //cumple con la limitacion diaria de recursos
        exit(-1);
    return true;
}
/****************************************/
/* RETRASO / ADEANTO */

```

```

/***********************/
int get_sequencing_instant_suc(std::set<disp_instant, CompareDisponibilities> *disponibility, Activity * act)
{
    int t_earliest=0;           set<disp_instant, CompareDisponibilities>::iterator it=disponibility->begin();
    /*Encontrar el instante mas pronto que puede empezar, dependiendo de cuando terminen sus sucesoras*/
    if(lact->sucesorsEnd()){
        for(int suc=0;suc<act->getList_Activities_suc()->size();suc++)
            if(act->getList_Activities_suc()->at(suc)->activity->getTSecuenciacion()>act->getList_Activities_suc()->at(suc)->activity-
getTNormal(>t_earliest)
                t_earliest=act->getList_Activities_suc()->at(suc)->activity->getTSecuenciacion()>act->getList_Activities_suc()-
at(suc)->activity->getTNormal();
        }
        /*Situarse en ese instante*/
        while(disponibility->end()!=it&& it->instant<=t_earliest) it++;
        while(disponibility->end()!=it&&it->instant< t_earliest+act->getTNormal() ){
            if(disponibility->end()==it)          return t_earliest;
            if(act->isEnoughResource(it)){       it++;           t_earliest=it->instant;
            }else                           it++;
        }
        return t_earliest;
    }
    bool dependenciesFinished(string step, Activity * act, std::vector<Activity *> examined){
        /*Comprobar si ya estan examinadas las dependencias*/
        if(step.compare("retraso")!=0) {
            /*En retraso tenemos que ver el proyecto del reves por lo que intenresa comprobar las sucesoras*/
            for(int i=0; i< act->getList_Activities_suc()->size();i++)
                if(act->getList_Activities_suc()->at(i)->activity->isContained(examined)&&lact->sucesorsEnd()) return false;
        }else{ /*En adelanto atenderemos a lasp redcesoras*/
            for(int i=0; i< act->getList_Activities_pred()->size();i++)
                if(!act->getList_Activities_pred()->at(i)->activity->isContained(examined)&&lact->predecesorsInitial()) return false;
        }
        return true;
    }
float Project::limitationResourcesParallel( deque<Activity*> selected, string step){
    std::vector<procesing_activity> procesing;      disp_instant disponibility;  std::vector<Activity*> examined;
    int t=0, j=0;   float t_max=0;           disp_resource resource_in_t;      Activity * act;
    Activity::resource * rec;   procesing_activity procesing_act, act_proces_fin;
    if(this->resources->size()>0){
        showErrorInfo(E_ALGORITHM_NOT_EXIST_RESOURCES, " proceso de "+step+" de actividades en paralelo");
        return INF;
    }
    /*Inicializacion*/
    mode_debug("\n\n*****PARALELO*****\n\n");
    /*Disponibilidad inicial, t=0*/
    for(int i=0; i<this->resources->size();i++) {
        resource_in_t.resource=resources->at(i);           resource_in_t.units_disp=resources->at(i)->getUnitsMax();
        disponibility.resources.push_back(resource_in_t);
    }
    disponibility.instant=0;
    /*Secuenciar actividades*/
    while(procesing.size()>0||t==0){
        mode_debug("\n\nITERACION __T="+num_to_str(t)+"\n");
        /*Hay alguna actividad procesandose?*/
        if(procesing.size()>0){   act_proces_fin=procesing.front();      t=act_proces_fin.instant;   }
        /*actualizar el procesamiento, todas borrar las actividades que terminan en t y añadirlas a vistas*/
        while(procesing.size()>0&&act_proces_fin.instant==t){
            mode_debug("La actividad "+act_proces_fin.act->getName()+" termina de procesarse\n");
            /*Añadir las a vistas*/
            examined.push_back(act_proces_fin.act);           procesing.erase(procesing.begin());
            /*Restaurar la disponibilidad*/
            for(int k=0; k<disponibility.resources.size();k++){
                rec=act_proces_fin.act->getResource(disponibility.resources.at(k).resource->getName());
                if(rec!=NULL) // consume de ese recurso
                    disponibility.resources.at(k).units_disp+=rec->units_asig;
            }
            if(procesing.size()>0)      act_proces_fin=procesing.front();
        }
        mode_debug(" ,disponibility.resources,t,""\n");           mode_debug(" elegir{"+selected+",}\n");
        /*Comprobar que actividades elegibles se pueden secuenciar*/
        j=0;
        while(!selected.empty()&&j<selected.size()){
            /*Hay suficientes recursos para secuenciar esa actividad?*/
            if(selected.at(j)->isEnoughResource(disponibility)&&dependenciesFinished(step,selected.at(j),examined)){
                /*Pasa a procesarse*/
                act=selected.at(j);           procesing_act.act=act;      procesing_act.instant=t+act->getTNormal();
                procesing=insert_proces(procesing,procesing_act);      act->setTSecuenciacion(t);
                selected.erase(selected.begin()++);
                mode_debug(" Se secuencia "+act->getName()+" en "+num_to_str(t)+" y termina en
"+num_to_str(procesing_act.instant)+"\n");
                /*Consumir los recursos disponibles*/
                for(int k=0; k<disponibility.resources.size();k++){
                    rec=act->getResource(disponibility.resources.at(k).resource->getName());
                    if(rec!=NULL)// no esta asignado ese recurso
                        disponibility.resources.at(k).units_disp-=rec->units_asig;
                }
            }
            j++;
        }
        mode_debug(" ,disponibility.resources, t,""\n");
    }
    return t;
}

float Project::limitationResourcesSerie( deque<Activity*> selected, string step){
    int n_act_to_process=activities.size();   std::set<disp_instant, CompareDisponibilities> *disponibility=new set<disp_instant, CompareDisponibilities>;
}

```

```

std::vector<Activity*> examined;           int t;           int t_max=0;
disp_resource resource_in_t;                disp_instant resources_in_t; Activity * act;
if(this->resources->size()==0{
    showErrorInfo(E_ALGORITHM_NOT_EXIST_RESOURCES, " proceso de "+step+" de actividades en serie");
    return INF;
}
/*INICIALIZACION*/
mode_debug("\n*****SERIE*****\n\n");
/*Disponibilidad inicial, t=0*/
for(int i=0; i<this->resources->size();i++) {
    resource_in_t.resource=resources->at(i);      resource_in_t.units_disp=resources->at(i)->getUnitsMax();
    resources_in_t.resources.push_back(resource_in_t);
}
resources_in_t.instant=0; disponibility->insert(resources_in_t);
/*Se examinan todas las actividades*/
while(n_act_to_process>0) {
    mode_debug("\n\nITERACION _____\n");
    /*Actividad a secuenciar*/
    act=selected.front(); selected.pop_front();
    /*calcular instante en que se secuenciará*/
    if(step.compare("retraso")==0) t= get_sequencing_instant_suc(disponibility, act); // comprueba las sucesoras
    else t= get_sequencing_instant(disponibility, act); //comprueba las predecesoras
    /*Secuenciar*/
    act->setTSecuenciacion(t); examined.push_back(act);
    mode_debug("La actividad "+act->getName()+" se secuencia en "+num_to_str(t)+"\n");
    /*Guardamos cual es el instante de fin del proyecto, que será cuando finalice la actividad que termina más tarde*/
    if(t>act->getTNormal()) t_max=t+act->getTNormal();
    /*actualizar disponibilidad*/
    update_disponibility(disponibility, act, t); mode_debug(" ,disponibility,\n"); n_act_to_process--;
}
delete(disponibility);
return t_max;
}

bool Project::retrasoAdelantoActivities(int eschema_retraso, int eschema_adelanto){
    float max1, max2; deque<Activity*> activities_ordered;
    do{
        mode_debug("\n##### ADELANTO/ATRASO#####\n");
        /*Ordenar actividades en orden decreciente de finalización*/
        for(int i=0; i<this->activities.size();i++) activities_ordered.insert(activities_ordered.activities.at(i), MIN_EFT_FIFO);
        reverse(activities_ordered.begin(), activities_ordered.end()); //orden decreciente
        /*RETRASO: aplicar esquema con regla*/
        mode_debug("||||| RETRASO|||||\n");
        mode_debug("elegir=",activities_ordered,"");
        if(eschema_retraso==_SERIE) max1=limitationResourcesSerie(activities_ordered, "retraso");
        else max1=limitationResourcesParalel(activities_ordered, "retraso");
        if(max1==INF) return false; //no se ha podido aplicar el algoritmo
        /*Traducir tiempos de secuenciación*/
        for(int i=0; i<this->activities.size();i++) activities.at(i)->setTMin(max1-activities.at(i)->getTSecuenciacion()-activities.at(i)->getTNormal());
        end->setTMin(max1);
        /*Comprobar que el algoritmo lo resuelve bien*/
        if(validateDependences()) exit(-1);
        if(validateLimitationMaxUnitsAllocated()) exit(-1);
        /*Ordenar actividades en orden creciente de inicio*/
        activities_ordered.clear();
        for(int i=0; i<this->activities.size();i++) activities_ordered.insert(activities_ordered.activities.at(i), MIN_EST_FIFO);

        /*ADELANTO: aplicar esquema con regla*/
        mode_debug("||||| ADELANTO|||||\n");
        mode_debug("elegir=",activities_ordered,"");
        if(eschema_adelanto==_SERIE) max2=limitationResourcesSerie(activities_ordered, "adelanto");
        else max2=limitationResourcesParalel(activities_ordered, "adelanto");
        if(max2==INF) return false; //no se ha podido aplicar el algoritmo
        /*actualizar t.min*/
        for(int i=0; i<activities.size();i++) activities.at(i)->setTMin(activities.at(i)->getTSecuenciacion());
        end->setTMin(max2);
        /*Actualizar t.max y holguras*/
        updateTMax();
        /*Comprobar si el algoritmo lo resuelve bien*/
        if(validateDependences()) exit(-1);
        if(validateLimitationMaxUnitsAllocated()) exit(-1);
    }
    /*Estos pasos se aplicaran mientras no se converja a una solución*/
    while(max1-max2!=0);
    return true;
}

```

7.2.12 MinCostMinDuratio.cc

```
#include "Project.h"
#include <string>
#include <iostream>
#include <sstream>
/*********************************************************************
 *          MIN COST MIN DURACION
 *****/
void Project::extractPaths(vector<Path*> *paths, Path * path, Activity* begin) {
    Activity * act;
    /*No hay mas actividades, se añade el camino a la lista*/
    if(begin->sucesoresEnd()){
        Path * finished_path=new Path(path); //guarda el camino y lo añade
        paths->push_back(finished_path);      path->clear(); //limpia el camino para posteriores usos
    }else {
        /*Tiene sucesoras y las examinamos*/
        for(int i=0; i<begin->getList_Activities_suc()->size(); i++) {
            act=begin->getList_Activities_suc()->at(i)->activity; Path * path2=new Path(path); // guardamos el camino antes de modificarlo
            /*Se añade una de las sucesoras y seguimos examinando su sucesores*/
            path->getPath()->push_back(act); extractPaths(paths, path, act);
            /*Cuando terminamos de examinar el anterior camino, restauramos el camino que teniamos hasta entonces y seguimos buscando*/
            path->copy(path2); delete(path2);
        }
    }
    path->clear();
}
float calcul_overrun(std::vector<Path*> paths, Path * activities){
    int cost=0; int j;
    /*No hay actividades seleccionadas*/
    if(activities->getPath()->size()==0) return INF;
    /*Comprobar si todos los caminos almenos tienen una actividad de las elegidas para reducir su t.normal */
    for(int i=0; i<paths.size(); i++) {
        for(j=0; j<activities->getPath()->size(); j++) if(paths.at(i)->includesActivity(activities->getPath()->at(j))) break;
        /*Hay almenos un camino que no tiene ninguna actividad a reducir*/
        if(j==activities->getPath()->size()) return INF; // no es una solucion factible
    }
    /*calcula el coste*/
    for(int i=0; i<activities->getPath()->size(); i++) {
        for(int j=0; j<paths.size(); j++) {
            if(paths.at(j)->includesActivity(activities->getPath()->at(i))){
                // solo contamos la actividad una vez, da igual en que camino este
                cost+=activities->getPath()->at(i)->getOportunityCost();
                break;
            }
        }
    }
    return cost;
}
float Project::get_min_cost(std::vector<Path*> paths_to_modif, int n_act, Path * activities_to_modif, Path * solution){
    float con_act=INF; Activity * act; float t_calculated; Path * aux=new Path(); Path * solution_with_act=new Path();
    Path * solution_without_act=new Path(); float sin_act;
    /*Probar todas las posibles combinaciones de actividades a modificar y elegir la de minimo coste*/
    /*Todas la actividades se han tenido en cuenta*/
    if(n_act==activities.size()){
        /*Calcula el sobrecoste de esa elección de actividades*/
        t_calculated=calcul_overrun(paths_to_modif,activities_to_modif); // es factible?
        solution->copy(activities_to_modif); return t_calculated;
    }else /*BACKTRACKING: solución sin esta actividad, y con esta actividad*/
    {
        aux=new Path(activities_to_modif); // guardamos las actividades seleccionadas hasta el momento
        /*Solución sin haber cogido esta actividad*/
        sin_act=get_min_cost(paths_to_modif, n_act+1, activities_to_modif, solution_without_act);
        activities_to_modif=new Path(aux); // restaurar la lista a como estaba antes de calcular la solución sin la actividad
        act=activities.at(n_act);
        /*Se puede reducir los días de esta actividad?*/
        if(act->getDiffTNormalTope(>0) {
            /*Solución habiendo añadido la actividad a la solución*/
            activities_to_modif->getPath()->push_back(act);
            con_act=get_min_cost(paths_to_modif, n_act+1, activities_to_modif, solution_with_act);
        }
        activities_to_modif=new Path(aux); // restaurar la lista a como estaba antes de calcular la solución con la actividad
        /*Es mejor solución, sin la actividad o con la actividad?*/
        if(con_act<sin_act){ solution->copy(solution_with_act); return con_act;
        }else{ solution->copy(solution_without_act); return sin_act; }
    }
}
int max(std::vector<Path*> *paths) //calcula cual camino dura mas
{
    int max=-1;
    for(int i=0; i<paths->size(); i++) if(max<paths->at(i)->getDuration()) max=paths->at(i)->getDuration(); return max;
}
int calcul_next_instant(std::vector<Path*> *paths, Path * activities){
    int j; int next_instant=-1; std::vector<Path*> *paths_without_activities= new std::vector<Path*>;
    /*Guardar los caminos que no incluyen actividades que se reducirán*/
    for(int i=0; i<paths->size(); i++){
        for(j=0; j<activities->getPath()->size(); j++) if(paths->at(i)->includesActivity(activities->getPath()->at(j))) break;
        /*El camino no tiene ninguna de las actividades*/
        if(j==activities->getPath()->size()) paths_without_activities->push_back(paths->at(i));
    }
    /*De los caminos sin actividades a reducir, entre estos obtener el que dura mas tiempo */
    next_instant=max(paths_without_activities);
    delete(paths_without_activities);
    return next_instant;
}
```

```

int min_dif_between_tnormal_ttope(std::vector<Activity*> *activities){
    float min=INF;
    /*entre las actividades ,encontrar la minima diferencia entre el t. normal y el tope*/
    for(int i=0; i<activities->size();i++) { if(min>activities->at(i)->getDifTNormalTope()) min=activities->at(i)->getDifTNormalTope();}
    return min;
}
Path * Project::select_activities_min_cost(std::vector<Path*> *paths, float &cost, int& max_decr_days){
    int Time=max(paths); // cual es la duracion del camino que tarda mas
    int next_instant=0; Path* activities_to_modif_pruebas=new Path(); Path* activities_to_modif_solution=new Path();
    std::vector <Path*> paths_to_modif;
    mode_debug("Maxima duracion a reducir = "+num_to_str(Time)+"\n");
    /*Guardar todos los caminos que se quieren reducir porque son los que mas tardan, que seran los caminos criticos*/
    for(int i=0; i<paths->size();i++) { if(paths->at(i)->getDuration()==Time) paths_to_modif.push_back(paths->at(i)); }
    /*calcula el coste de modificar los caminos criticos*/
    cost=get_min_cost(paths_to_modif, 0, activities_to_modif_pruebas, activities_to_modif_solution);
    /*No es posible reducir todos los caminos criticos*/
    if(cost==INF) return NULL;
    /*Se elige, de los caminos que no contienen actividades a reducir, el de mayor duracion*/
    next_instant=calcul_next_instant(paths,activities_to_modif_solution);
    /*obtener el minimo numero de dias que se pueden decrementar las actividades que forman parte de los caminos a reducir*/
    max_decr_days=min_dif_between_tnormal_ttope(*paths,*activities_to_modif_solution->getPath());
    /*Si hay algun camino que no contiene actividades a reducir*/
    if(next_instant!=-1) {
        /*calcular el minimo entre: combertir el camino no critico, de maxima duracion , en critico o los dias maximo que se pueden reducir todas las actividades segun su t.normal-t.tope */
        if(max_decr_days>Time-next_instant) max_decr_days=Time-next_instant;
    }
    /*numero de dias que se reduce * coste*/
    cost*=max_decr_days;
    delete(activities_to_modif_pruebas);
    return activities_to_modif_solution;
}
void Project::decrement_days(std::vector<Path*> *paths,int max_decr_days,Path*act_to_modif){
    int max_decr_days_allowed=max_decr_days; Activity * act; Path* path;
    mode_debug("\nDecremento de dias de los caminos = "+num_to_str(max_decr_days_allowed)+"\n-----\n");
    /*Reducir el t.normal de las actividades a modificar, lo maximo que se pueda*/
    for(int i=0; i< act_to_modif->getPath()->size();i++){
        act=act_to_modif->getPath()->at(i); act->setDifTNormalTope(act->getDifTNormalTope()-max_decr_days_allowed);
        /*Reducir la duracion de los caminos que incluyen la actividad*/
        for(int j=0; j<paths->size();j++){
            path=paths->at(j);
            if(path->includesActivity(act)) path->setDuration(path->getDuration()-max_decr_days_allowed);
        }
    }
}
Path * Project::select_activities_min_cost(std::vector<Path*> *paths, float &cost, int& max_decr_days, int cost_searched){
    int Time=max(paths); // cual es la duracion del camino que tarda mas
    int next_instant=0; Path* activities_to_modif_pruebas=new Path(); Path* activities_to_modif_solution=new Path();
    std::vector <Path*> paths_to_modif;
    stringstream cost_searched_s; int days=1;
    mode_debug("Maxima duracion a reducir = "+num_to_str(Time)+"\n");
    /*Guardar todos los caminos que se quieren reducir porque son los que mas tardan, que seran los caminos criticos*/
    for(int i=0; i<paths->size();i++) { if(paths->at(i)->getDuration()==Time) paths_to_modif.push_back(paths->at(i)); }
    /*calcula el coste de modificar los caminos criticos*/
    cost=get_min_cost(paths_to_modif, 0, activities_to_modif_pruebas, activities_to_modif_solution);
    /*No es posible reducir todos los caminos criticos*/
    if(cost==INF) return NULL;
    /*Se elige, de los caminos que no contienen actividades a reducir, el de mayor duracion*/
    next_instant=calcul_next_instant(paths,activities_to_modif_solution);
    /*obtener el minimo numero de dias que se pueden decrementar las actividades que forman parte de los caminos a reducir*/
    max_decr_days=min_dif_between_tnormal_ttope(*paths,*activities_to_modif_solution->getPath());
    /*Si hay algun camino que no contiene actividades a reducir*/
    if(next_instant!=-1) {
        /*calcular el minimo entre: combertir el camino no critico, de maxima duracion , en critico o
         los dias maximo que se pueden reducir todas las actividades segun su t.normal-t.tope */
        if(max_decr_days>Time-next_instant) max_decr_days=Time-next_instant;
    }
    /*Calculo de coste*/
    if(cost>cost_searched) // superamos el coste por dia establecido, ya no podremos reducir mas la duracion del proyecto
    {
        cost=INF; return NULL;
    }
    delete(activities_to_modif_pruebas); return activities_to_modif_solution;
}
void Project::calculPathMinTimeMinCost(int cost_searched){
    std::vector<Path*> *paths= new vector<Path*>; Path * path=new Path(); float cost;
    int max_decr_days; Path *act_to_modif; Activity * act;
    /* INICIALIZACION */
    mode_debug("*****INICIALIZACION*****\n\nCaminos:\n");
    this->overrun=0; //sobrecoste
    /*obtiene en paths todos los posibles caminos del proyecto*/
    extractPaths(paths, path,this->begin);
    /*calcula la duracion en dia de cada uno de los caminos*/
    for(int i=0; i<paths->size();i++) {
        paths->at(i)->calculDuration();
        mode_debug(" "+num_to_str(i)+" : "+paths->at(i)->getPath()+" "+num_to_str(paths->at(i)->getDuration())+"\n");
    }
    /*calcula el numero de dias que se puede reducir cada actividad , como Tiempo normal - Tiempo tope*/
    for(int i=0; i<activities.size();i++) {
        act=activities.at(i); act->setDifTNormalTope();
        mode_debug("T.Normal-T.tope "+act->getName()+" = "+num_to_str(act->getDifTNormalTope())+"\n");
    }
    mode_debug("\n\n***** EJECUCION *****\n\n");
}

```

```

act_to_modif=select_activities_min_cost(paths,cost,max_decr_days, cost_searched);
while(cost!=INF) {
    mode_debug("Sobrecoste Minimo = "+num_to_str(cost)+"\n");
    this->overrun+=cost;
    decrement_days(paths,max_decr_days,act_to_modif);
    if(cost>cost_searched) break;
    act_to_modif=select_activities_min_cost(paths,cost,max_decr_days, cost_searched);
}
/*Actualiza los parametros del proyecto segun los resultados*/
for(int i=0; i<activities.size();i++) { act=activities.at(i); act->setTNormal(act->getTTope()+act->getDifTNormalTope());}
this->calculCriticalPath();
delete(paths); delete(path); delete(act_to_modif);
}

void Project::calculoPathMinTimeMinCost(){
std::vector<Path*> *paths= new vector<Path*>; Path * path=new Path(); float cost; int max_decr_days;
Path *act_to_modif; Activity * act;
mode_debug("*****INICIALIZACION*****\n\nCaminos:\n");
this->overrun=0; //sobrecoste
extractPaths(paths, path,this->begin);
for(int i=0; i<paths->size();i++){
    paths->at(i)->calculDuration();
    mode_debug(" "+num_to_str(i)+"\t",paths->at(i)->getPath()," "+num_to_str(paths->at(i)->getDuration())+"\n");
}
for(int i=0; i<activities.size();i++) {
    act=activities.at(i); act->setDifTNormalTope();
    mode_debug("\nT.Normal-T.tope "+act->getName()+" = "+num_to_str(act->getDifTNormalTope())+"\n");
}
mode_debug("\n***** EJECUCION *****\n\n");
act_to_modif=select_activities_min_cost(paths,cost,max_decr_days);
while(cost!=INF){
    mode_debug("Sobrecoste Minimo = "+num_to_str(cost)+"\n");
    /*Sobrecoste*/
    this->overrun+=cost;
    decrement_days(paths,max_decr_days,act_to_modif);
    act_to_modif=select_activities_min_cost(paths,cost,max_decr_days);
}
for(int i=0; i<activities.size();i++)
{
    act=activities.at(i);
    act->setTNormal(act->getTTope()+act->getDifTNormalTope());
}
this->calculCriticalPath();
delete(paths);
delete(path);
delete(act_to_modif);
}
}

```

7.2.13 Rules.h

```

#include<deque>
#include "Activity.h"

/*
          ESQUEMAS
 */
#define _PARALEL 3000
#define _SERIE 3001
#define _MULTIPASADA 3002

/*
      REGLAS PRIORIDAD
 */
#define MIN_LFT_FIFO 2000 //instante de finalizacion mas tardio
#define MIN_EFT_FIFO 2001 //instante de finalizacion mas temprano
#define MIN_LST_FIFO 2002 //instante de comienzo mas tardio
#define MIN_EST_FIFO 2003 //instante de comienzo mas temprano
#define MIN_HT_FIFO 2004 // holgura total
#define MIN_HL_FIFO 2005 // hlgura libre
#define MAX_DEM_FIFO 2006 // demanda de recursos
#define MIN_DUR_FIFO 2007 // duracion de la actividad
#define MAX_NSUC_FIFO 2008 // numero de sucesoras

#ifndef RULES_H_
#define RULES_H_
/////////////////////////////// REGLAS DE PRIORIDAD PRIORity ///////////////////
/////////////////////////////// ELEGIR REGLA SEGUN OPCION ///////////////////
bool choiceRule(Activity * act1,Activity * act2); // aplica la regla indicada por rule
deque<Activity*> binsearch(deque<Activity*> activities, Activity * act, int left, int right, int rule); //algoritmo de divide y venceras
deque<Activity*> insert(deque<Activity*> activities, Activity * act, int rule_prio); //insertar actividad de forma ordenada
#endif;

```

7.2.14 Rules.cc

```
#include "Rules.h"
#include "Activity.h"

// REGLAS DE PRIORIDAD
bool MinLFT_FIFO(Activity * act1, Activity * act2){
    /*Minimo instante de finalizacion tardio*/
    return(act1->getTMax()+act1->getTNormal()<act2->getTMax()+act2->getTNormal());
}
bool MinEFT_FIFO(Activity * act1, Activity * act2){
    /*Minimo instante de finalizacion mas temprano*/
    return(act1->getTMin()+act1->getTNormal()<act2->getTMin()+act2->getTNormal());
}
bool MinHT_FIFO(Activity * act1, Activity * act2){
    /*Minima holgura total*/
    return(act1->getHTotal()<act2->getHTotal());
}
bool MinHL_FIFO(Activity * act1, Activity * act2){
    /*Minima holgura libre*/
    return(act1->getHFree()<act2->getHFree());
}
bool MinEST_FIFO(Activity * act1, Activity * act2){
    /*Minimo instante de comienzo mas temprano*/
    return(act1->getTMin()<act2->getTMin());
}
bool MinLST_FIFO(Activity * act1, Activity * act2){
    /*Minimo instante de comienzo mas tardio*/
    return(act1->getTMax()<act2->getTMax());
}
bool MinDUR_FIFO(Activity * act1, Activity * act2){
    /*Minima duracion de la actividad*/
    return(act1->getTNormal()<act2->getTNormal());
}
bool MaxNSUC_FIFO(Activity * act1, Activity * act2){
    /*Maximo numero de sucesoras*/
    return(act1->getList_Activities_suc()->size()>act2->getList_Activities_suc()->size());
}
bool MaxDEM_FIFO(Activity * act1, Activity * act2){
    int prior1=0; int prior2=0;
    /*Maxima demanda de recursos*/
    for(int i=0; i<act1->getResources().size();i++) prior1+=act1->getResources().at(i)->units_asig;
    for(int i=0; i<act2->getResources().size();i++) prior2+=act2->getResources().at(i)->units_asig;
    prior1=act1->getTNormal();
    prior2=act2->getTNormal();
    return(prior1>prior2);
}

// ELEGIR REGLA SEGUN OPCION
bool choiceRule(Activity * act1, Activity * act2, int rule){
    switch(rule){
        case MIN_LFT_FIFO: return MinLFT_FIFO(act1,act2); break;
        case MIN_EFT_FIFO: return MinEFT_FIFO(act1,act2); break;
        case MIN_LST_FIFO: return MinLST_FIFO(act1,act2); break;
        case MIN_EST_FIFO: return MinEST_FIFO(act1,act2); break;
        case MIN_HT_FIFO: return MinHT_FIFO(act1,act2); break;
        case MIN_HL_FIFO: return MinHL_FIFO(act1,act2); break;
        case MAX_DEM_FIFO: return MaxDEM_FIFO(act1,act2); break;
        case MIN_DUR_FIFO: return MinDUR_FIFO(act1,act2); break;
        case MAX_NSUC_FIFO: return MaxNSUC_FIFO(act1,act2); break;
        default: return false; break;
    }
}

// INSERCIÓN ORDENADA
deque<Activity*> binsearch(deque<Activity*> activities, Activity * act, int left, int right, int rule) {
    /*Esta funcion hace un ordenamiento de actividades segun una regla de prioridad, siguiendo una estrategia de divide y vencerás, con coste temporal de O(log n)*/
    if (left > right) return activities;
    if(left==right) {
        /*Ordenar segun la regla de prioridad*/
        if(choiceRule(act,activities.at(left), rule)) activities.insert(activities.begin() + left, act);
        else activities.insert(activities.begin() + left + 1, act);
        return activities;
    }
    int middle = (left+right)/2;
    /*Como la lista esta ordenada, si la partimos en 2, en que parte estara la posicion donde insertarla*/
    if (choiceRule(act,activities.at(middle), rule)) return binsearch(activities,act,left,middle,rule);
    else return binsearch(activities,act,middle+1,right,rule);
}

deque<Activity*> insert(deque<Activity*> activities, Activity * act, int rule_prio){
    /*Inserta la actividad act de forma ordenada*/
    if(activities.size()==0){
        activities.push_back(act);
        return activities;
    }else return binsearch(activities, act, 0, activities.size()-1, rule_prio);
}
```

7.2.15 Resource.h

```
#include <string>
#include <vector>
using namespace std;

#ifndef Resource_H_
#define Resource_H_
class Resource {
    std::string name;
    float units_max; // unidades disponibles del recurso
public:
    Resource(string name);    Resource(string name, float units_max);
    ~Resource();
    void setUnitsMax(int units){ units_max=units; }    int getUnitsMax(){ return units_max; };
    string getName(){ return name; };
};

/*DISPONIBILIDAD DE RECURSO EN UN INSTANTE EN LIMITACION DE RECURSOS*/
typedef struct {    Resource * resource; // estructura que representa las unidades disponibles de un recurso
    int units_disp; // recurso
}disp_resource; // unidades disponibles para asignar del recurso
typedef struct {    int instant; // estructura que representa las unidades disponibles de los recursos en un instante
    vector<disp_resource> resources; // unidades de cada recurso disponible en ese instante
}disp_instant;
class CompareDisponibilidades // clase auxiliar para permitir ordenar las disponibilidades por el instante
{
public:
    bool operator()(disp_instant d1, disp_instant d2) {
        if (d1.instant < d2.instant) return true;
        return false;
    }
};
#endif
```

7.2.16 Resource.cc

```
#include "Resource.h"
#include "Exceptions.h"
#include <string.h>
/*************************************
/* Constructores */
/*************************************
Resource::Resource(std::string name){ this->name=name; this->units_max=1; }
Resource::Resource(std::string name, float units_max){ this->name=name; this->units_max=units_max; }

/*************************************
/* Destructor */
/*************************************
Resource::~Resource(){}
```

7.3 *Manual del usuario*

Este documento ha sido creado para facilitar al usuario el manejo de la aplicación creada en el proyecto de GPI. Las imágenes o indicaciones pueden variar debido a que algunas características pueden no estar disponibles en el software final.

Ante cualquier duda con respecto al desarrollo y el funcionamiento del Software es aconsejable leer el resto de documentación incluida en el proyecto.

7.3.1 Ventana Principal

Al ejecutar el programa nos encontraremos con la siguiente vista. Esta es la ventana principal desde la que podremos gestionar actividades, fechas festivas o laborables, recursos, guardar o cargar proyectos, generar informes y analizar los resultados del proyecto obtenidos.

	Actividad	Recursos	Duración	Predecesores	Tiempo Tote	Coste N.	Coste C.
1	A		3		3	0	0
2	B		1		1	0	0
3	C		5		5	0	0
4	D		6	B	6	0	0
5	E		5	A-C	5	0	0
6	F		4	A-B	4	0	0
7	G		7	A-B-C	7	0	0
8	H		2	D-E-F-G	2	0	0
9	I		2	E	2	0	0

7.3.1.1 Barra de Herramientas



Añadir. Este botón nos abrirá una ventana para añadir actividades.



Editar. Cuando seleccionemos una fila de la Ventana principal este botón se activará, permitiéndonos abrir una ventana para editar la duración, recursos, y predecesores de la actividad seleccionada.



Eliminar. Al igual que con el anterior botón, este solo se activará con una fila seleccionada.



Analizar. Si se ha generado un proyecto podremos analizar sus resultados haciendo click en este botón.



Generar Informe. Al pulsar este botón abriremos el menú generar informe.



Gestionar Fechas. Este botón abre una nueva ventana desde la que podremos añadir, editar o eliminar fechas festivas además de indicar que días serán considerados como laborables extra.



Gestionar Recursos. Este botón abre una nueva ventana desde la que podremos añadir, editar o eliminar recursos.

7.3.1.2 Barra de menú

7.3.1.2.1 Archivo



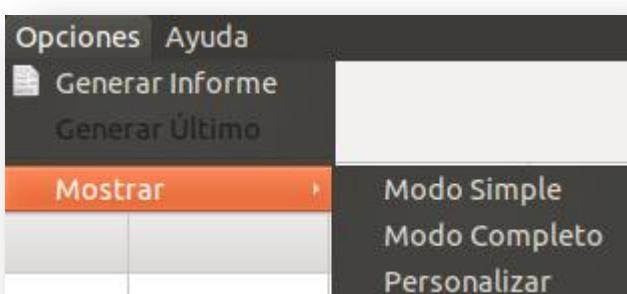
Guardar Proyecto: Guarda el proyecto abierto en la última ruta en la que se abrió o guardó el proyecto. Si todavía no existe la ruta se abrirá el menú de “Guardar Cómo”.

Guardar Cómo: Abre un menú para seleccionar dónde y cómo guardar el proyecto. Una vez se realiza este proceso se habilita “Guardar”.

Cargar: Abre un menú que nos permite seleccionar que proyecto deseamos cargar en la ventana principal.

Salir: Cierra la aplicación.

7.3.1.2.2 Opciones

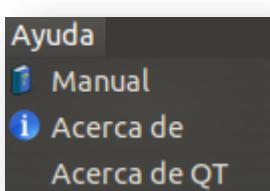


Generar Informe: Abre una ventana en la que podemos seleccionar que tipo de informe queremos generar.

Generar Último: Genera un informe del mismo tipo que el último generado, si no se ha generado todavía ninguno informe abrirá la ventana “Generar Informe”.

Mostrar: Permite cambiar las columnas que se mostrarán en la Ventana Principal.

7.3.1.2.3 Ayuda



Manual: Abre este documento.

Acerca de QT: Abre una ventana acerca del Software de Desarrollo QT.

Acerca de: Abre una ventana acerca de la aplicación.

7.3.2 Añadir Actividad

Nombre: Da nombre a la actividad

Duración: Añade la duración a la Actividad

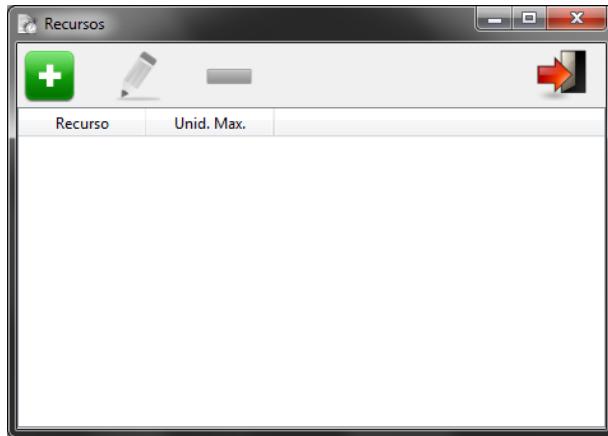


Predecesores - Asignar: Desde la ventana de la izquierda podremos marcar las actividades que preceden a la actividad que estamos creando.



Recursos – Asignar: Desde la siguiente ventana podremos asignar Recursos a la actividad.

7.3.3 Gestionar Recursos

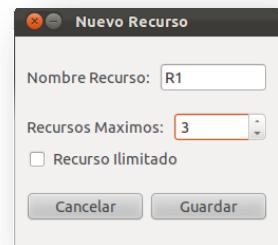


Desde ella podemos, añadir, eliminar o actualizar recursos. La tabla es editable (salvo el nombre).

Añadir Recurso: Al pulsar en este botón podremos añadir un nuevo recurso mediante la ventana de la derecha.

Editar Recurso: Permite editar el recurso seleccionado al pulsar este botón.

Eliminar Recurso: Al pulsar este botón elimina el recurso seleccionado.



7.3.4 Gestionar Fechas



Consta de dos partes divididas en tres pestañas:

7.3.4.1 Calendario:

Muestra un calendario real con las fechas festivas y laborables que hemos introducido.

7.3.4.2 Festivos y Laborables EXTRA:

En ambos se muestra una ventana idéntica y de mismo funcionamiento. En festivos se mostrarán las fechas festivas y en Laborables Extra las fechas que se quieran considerar como laborables sean o no festivo.

Añadir Fecha: Permite añadir una fecha mediante un cuadro de diálogo.

Actualizar Fechas: Actualiza el calendario para mostrar los nuevos festivos y laborables extra.

Eliminar Fecha: Elimina la fecha seleccionada.

Calendario			
		Festivos	Laborables Extra
	Fecha	Motivo	
1	01/01	Año Nuevo	
2	06/01	Día de Reyes	
3	19/03	San José	
4	01/05	Fiesta del T...	
5	15/08	La Asunción	
6	12/10	Fiesta Naci...	

7.3.5 Generar Informe

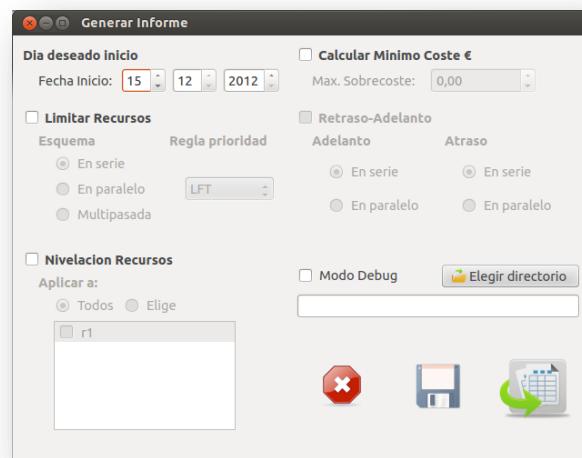
Selecciona los distintos tipos de algoritmos que deseas aplicar y sus configuraciones.

Modo Debug: crea un fichero por cada algoritmo en el directorio que tu eligas.

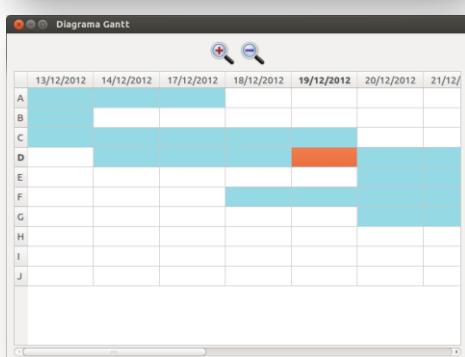
 Guarda las opciones para que podamos generar el informe más tarde.

 Genera un informe con las opciones seleccionadas y guarda estas opciones para generar otra vez un informe con las mismas opciones cuando queramos.

 Cancela las opciones seleccionadas y cierra la Ventana.



7.3.6 Informe

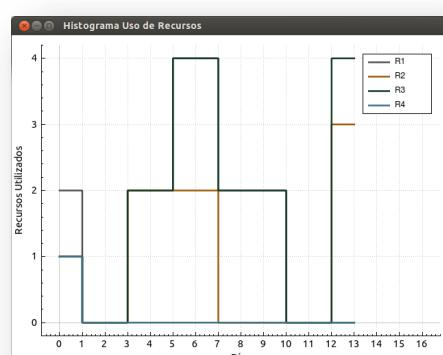


7.3.6.1 Informe Actividades:

En una tabla mostrará las actividades con sus inicios en formato de fechas además de otros datos como holgura libre, total y sucesores. También hay un breve informe sobre los posibles caminos críticos y una fecha de inicio y de fin de todo el proyecto.

7.3.6.2 Calendario Proyecto:

En el calendario se muestran resultados aquellos días en los que el proyecto se llevará a cabo.



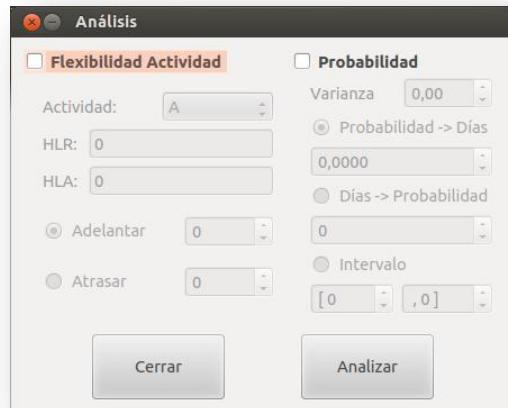
7.3.7 Análisis

7.3.7.1 Flexibilidad Actividad

Seleccionando la parte izquierda y una actividad obtendremos su análisis mediante un cuadro de diálogo al hacer click en Analizar.

7.3.7.2 Probabilidad Proyecto

Podemos realizar cálculos de probabilidad sobre el proyecto marcando su probabilidad e indicando que tipos de análisis de probabilidad deseamos realizar.



7.3.8 Atajos de teclado

ACCIÓN

- Añadir Nueva Actividad
- Editar Actividad Seleccionada
- Eliminar Actividad Seleccionada
- Abrir Calendario
- Abrir Recursos
- Generar Análisis
- Abrir Opciones Generar Reporte
- Generar Reporte con las últimas opciones guardadas
- Guardar Proyecto
- Guardar Proyecto Cómo
- Cargar Proyecto

Comando

- "Ctrl+N"
- "Ctrl+E"
- "Ctrl+X"
- "Ctrl+C"
- "Ctrl+R"
- "Ctrl+A"
- "Ctrl+O"
- "Ctrl+G"
- "Ctrl+S"
- "Ctrl+Shift+S"
- "Ctrl+L"