Universidad Politécnica
de Madrid

Escuela Técnica Superior de
Ingenieros Informáticos

Grado en Ingeniería Informática

Trabajo Fin de Grado

Generación Automática de Diagramas de Gantt

Autor: Carlos Mateos Martín

Tutora: Clara Benac Earle

Madrid, enero 2021

Este Trabajo Fin de Grado se ha depositado en la ETSI Informáticos de la Universidad Politécnica de Madrid para su defensa.

Trabajo Fin de Grado
Grado en Ingeniería Informática
Título: Generación Automática de Diagramas de Gantt
Enero 2021

Autor: Carlos Mateos Martín

Tutora:
Clara Benac Earle
Departamento de Lenguajes y Sistemas Informáticos e Ingeniería de Software
ETSI Informáticos
Universidad Politécnica de Madrid

Agradecimientos

A mi tutora Clara Benac,

quien me ha guiado y apoyado de forma paciente durante la realización de este trabajo prestándome su tiempo y esfuerzo.

A mi familia,

quienes han sido mi apoyo y ejemplo a seguir todos estos años. En especial a mis hermanos y a mis padres, por ser siempre el espejo en el que mirarme y una fuente continua de motivación.

Resumen

Los diagramas de Gantt son una representación bidimensional en la que en un eje se representa el tiempo y en el otro una lista de tareas. Mediante estos diagramas se puede ver las tareas que requieren más tiempo y las relaciones de dependencia que hay entre ellas. El presente trabajo está centrado en el desarrollo, test y documentación de una herramienta software, denominada AutoGantt, que facilita y automatiza la creación de este tipo de diagramas.

Existen multitud de herramientas que generan diagramas de Gantt, pero o bien son de pago o su sintaxis es compleja y, en muchas ocasiones, supone más trabajo estudiarla y entenderla que los beneficios que reporta su utilización. Es por este motivo que se ha buscado una interfaz entre el usuario y la herramienta simple, cuyo uso esté extendido y les resulte familiar: las hojas de cálculo. AutoGantt aúna las ventajas que ofrecen las hojas de cálculo, en cuanto a facilidad para la introducción de datos manualmente, y efectividad para la transmisión de la información que consiguen herramientas especializadas en confeccionar diagramas de Gantt. Con esta herramienta la creación de este tipo de diagramas se convierte en una tarea fácil de realizar y menos propensa a errores, ya que se oculta al usuario la compleja sintaxis de una potente herramienta de gestión de proyectos.

Abstract

Gantt charts are bidimensional charts which represent time in one of the axes and a list of tasks on the other one. These charts help people visualize the most time-consuming tasks and the relation among some of them. This work is focused on the process of design, develop, test and documentation of a software tool, named AutoGantt, that eases and automates the creation of this type of charts.

There are plenty of tools that make this kind of representations, but either they are under license or their syntax is complex, which makes them less efficient for occasional users. That is why we aimed for a method that abstracts for the user the syntax of a powerful project management tool. The solution we have found is spreadsheets, a widely known kind of software more likely to be familiar to users. AutoGantt joins the option to manually introduce data of spreadsheets and the visual power of representations made by specialized software making the task of creating Gantt charts easier and less error prone.

Tabla de contenidos

Tabla de ilustraciones

1 Introducción

AutoGantt, es una herramienta que une la simplicidad para introducir información de las hojas de cálculo y la efectividad para la transmisión visual de dicha información que ofrecen herramientas de planificación de tareas.

Este trabajo viene a cubrir una necesidad detectada por el grupo de investigación Babel de la ETSIINF. Los investigadores del grupo Babel, como tantos grupos de investigación, mandan propuestas para solicitar la financiación de proyectos de investigación a diferentes organismos (ayuntamientos, ministerios, la UE, etc). En dichas propuestas es obligatorio incluir un plan de trabajo detallado y, de manera casi obligatoria, un diagrama de Gantt. Existen multitud de herramientas que generan diagramas de Gantt, cada una con un formato de entrada diferente. El objetivo de este trabajo es facilitar la generación de diagramas de Gantt para usuarios que no están familiarizados con dichas herramientas de manera que puedan generar diagramas de Gantt de forma automática sin tener que aprender a usar las herramientas que los generan.

El código de AutoGantt y la documentación generada durante la realización de éste se encuentran disponibles en un repositorio público en GitHub accesible a través del siguiente enlace:

https://github.com/cmateos96/AutoGantt

La memoria está dividida en 5 capítulos estructurados de la siguiente manera.

En el **Capítulo 1** se desarrolla la introducción, se describe la introducción, la contribución del autor y se realiza una breve descripción de qué es un diagrama de Gantt, la definición de los objetivos que se persiguen con este proyecto y la planificación que se ha seguido para conseguir dichos objetivos.

En el **Capítulo 2** se hace una descripción del proceso de estudio de las diferentes alternativas de software de gestión de proyectos, así como las razones de elección de la herramienta TaskJuggler. También se exponen las diferentes alternativas de gestión de hojas de cálculo que se han tenido en cuenta. Para terminar este capítulo, se describen las tecnologías utilizadas durante el desarrollo de AutoGantt.

En el **Capítulo 3** se describe el proceso de diseño, la arquitectura software diseñada, la herramienta TaskJuggler y las limitaciones que ésta impone.

En el **Capítulo 4** se describe el proceso de desarrollo, el proceso de testeo, la estructura del árbol de directorios del proyecto y el formato de la hoja de cálculo que se ha creado para la consecución del objetivo. Para finalizar se especifican las instrucciones de uso y dónde encontrar la documentación del código desarrollado.

En el **Capítulo 5** se describen las conclusiones y se describen líneas futuras.

Por último, se pueden encontrar las fuentes bibliográficas utilizadas.

1.1 Contribución

Mi contribución en el presente trabajo es el estudio de las herramientas existentes en el mercado, el análisis de requisitos de la herramienta a desarrollar y el proceso de elección de los programas en los que me apoyaría, *TaskJuggler* y *Google Sheets*. El grueso de mi aportación es la realización del diseño, programación, documentación y testeo de una herramienta, AutoGantt, que convierte un archivo obtenido a partir de una hoja de cálculo en un archivo que un programa de gestión de proyectos es capaz de interpretar y convertir en un diagrama de Gantt.

1.2 Objetivos

En Internet se pueden encontrar multitud de herramientas de hojas de cálculo como por ejemplo Microsoft Excel o Google Sheets que nos permiten introducir datos fácilmente y visualizarlos gráficamente. También muchos programas que nos aportan representaciones gráficas más visuales que las prestadas por las propias hojas de cálculo. Por estos motivos se decidió combinar lo mejor de ambas herramientas, la simplicidad de las hojas de cálculo y la efectividad para comunicar información de las herramientas de planificación especializadas.

El objetivo del trabajo es desarrollar una herramienta que facilite la creación de diagramas de Gantt a los usuarios. Este objetivo ha sido dividido en los siguientes subobjetivos:

- **Objetivo 1**: elegir un formato para introducir datos que resulte familiar a los usuarios.
- **Objetivo 2**: elegir de entre las herramientas que permiten la creación de diagramas la más conveniente.
- **Objetivo 3**: diseñar y desarrollar una herramienta que, a partir del archivo de hoja de cálculo y utilizando el programa de gestión de proyectos elegido, realice de forma automática un diagrama de Gantt.
- **Objetivo 4**: verificar la calidad del código y el correcto funcionamiento de la herramienta desarrollada.
- **Objetivo 5**: documentar la herramienta desarrollada y el proceso que ha llevado el proyecto.

1.3 Descripción de tareas

En primer lugar, se expone una tabla que muestra la relación entre los objetivos que se especificaban en la sección anterior y las tareas, que se describen tras la misma.

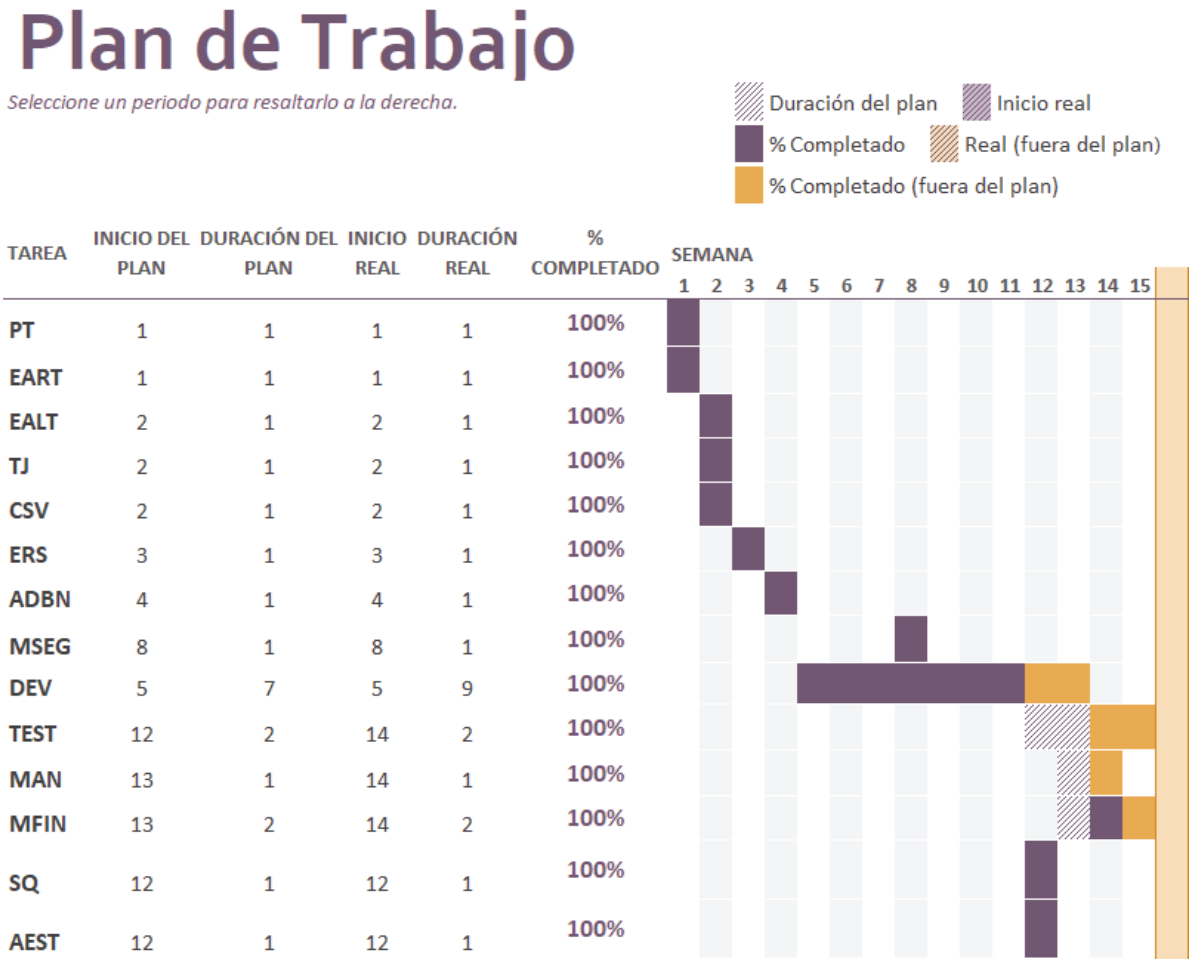
Objetivo	Descripción	Tareas
Objetivo 1	Elección formato introducción de datos	Estado del arte Estudio de las alternativas
Objetivo 2	Elección herramienta creación de diagramas	Estado del arte Estudio de las alternativas
Objetivo 3	Diseño y desarrollo de la herramienta	Especificación requisitos software Familiarización con TaskJuggler Análisis de requisitos y diseño de bajo nivel Desarrollo
Objetivo 4	Calidad del código y correcto funcionamiento	Implementación y ejecución de pruebas Despliegue de SonarQube Análisis estáticos del código y corrección de errores.
Objetivo 5	Documentación de herramienta y proyecto	Plan de trabajo Memoria de seguimiento Manual Memoria final

Tabla 1 Relación entre objetivos y tareas

La consecución de los objetivos descritos en la **Sección 1.2** se consiguió siguiendo un plan de trabajo formado por las tareas a continuación descritas. Junto a cada una de las tareas se indican, entre paréntesis, las siglas que las representan en el diagrama del plan de trabajo, **Sección 1.4.** Tras la descripción de cada una, se indica el número de horas de trabajo que ha llevado realizarla.

- Plan de trabajo (PT): se trata de un documento formal en el que se detallan tanto los objetivos como las tareas a realizar y el plazo planeado para realizarlas. Se acompaña un diagrama de Gantt para la visualización de este plan a lo largo del tiempo que dura el proyecto. (5h)
- Estado del arte (EART): qué es un diagrama de Gantt, qué representa, variaciones de esta representación a lo largo del tiempo. (6h)
- Estudio y elección de las herramientas de gestión de proyectos y hojas de cálculo: alternativas gratuitas, tanto de escritorio como de acceso mediante una aplicación web (EALT). (10h)
- Familiarización con la herramienta TaskJuggler (TJ): como realiza los diagramas, que datos y en que formato son necesarios como mínimo para el correcto funcionamiento de la herramienta. Aprender como representa TaskJuggler las tareas y los *milestones*. Estudiar las limitaciones que presentar la herramienta. (15h)
- Estudio del formato utilizado por *Google Sheets* y LibreOffice Calc y su capacidad de exportación con diferentes formatos. Estudio y elección del formato *comma separated value* CSV. (CSV) (5h)
- Especificación de requisitos software. (ERS) (5h)
- Análisis de requisitos y diseño de bajo nivel de la herramienta. (ADBN) (10h)
- Memoria de seguimiento. (MSEG) (10h)
- Implementación y documentación de la herramienta. (DEV) (140h)
- Implementación y ejecución de pruebas. (TEST) (25h)
- Escribir el manual de usuario de la herramienta a desarrollar. (MAN) (5h)
- Escribir memoria del proyecto, preparar y ensayar presentación. (MFIN) (45h)
- Despliegue de un servidor local SonarQube. (SQ) (2h)
- Análisis estáticos y corrección de errores detectados en el código. (AEST) (15h)

1.4 Plan de trabajo



TAREA	INICIO DEL PLAN	DURACIÓN DEL PLAN	INICIO REAL	DURACIÓN REAL	% COMPLETADO
PT	1	1	1	1	100%
EART	1	1	1	1	100%
EALT	2	1	2	1	100%
TJ	2	1	2	1	100%
CSV	2	1	2	1	100%
ERS	3	1	3	1	100%
ADBN	4	1	4	1	100%
MSEG	8	1	8	1	100%
DEV	5	7	5	9	100%
TEST	12	2	14	2	100%
MAN	13	1	14	1	100%
MFIN	13	2	14	2	100%
SQ	12	1	12	1	100%
AEST	12	1	12	1	100%

Ilustración 1 Plan de trabajo

1.4.1 Deviaciones respecto al plan de trabajo

Como se puede observar se han producido modificaciones respecto al plan. En primer lugar, a partir del comienzo del desarrollo empezaron a acumularse retrasos debidos principalmente al desconocimiento parcial del autor del lenguaje Python. Como consecuencia de la prolongación de esta tarea, se retrasaron el resto, produciéndose una acumulación de trabajo en las últimas dos semanas del plan. Cabe destacar que en el plan de trabajo original no se contempló el despliegue de SonarQube y realización de análisis estático del código, lo que acarreó alrededor de 17 horas más de trabajo.

2 Estado del arte

El objetivo de este trabajo es facilitar al usuario la creación de diagramas de Gantt. Existen multitud de herramientas para generar este tipo de diagramas, es por ello por lo que se ha realizado un estudio de las alternativas existentes en el mercado y se ha seleccionado la que mejor se acomoda a las necesidades. Por otra parte, se ha considerado que las hojas de cálculo son una forma habitual y familiar para los usuarios como forma de introducir datos. Por este motivo también se ha realizado un estudio y elección de los software de hojas de cálculo que mejor se acomodaban al proyecto y su filosofia.

2.1 Diagrama de Gantt: qué es, historia y variaciones.

Hoy en día, gracias en gran medida a la informatización, se ha conseguido eficiencia y eficacia a la hora de desarrollar tareas, tanto en el ambiente laboral como en la vida personal. La planificación es un elemento clave para llevar a cabo estas tareas. Hay multitud de métodos y de herramientas para la planificación de proyectos y uno de los más utilizados es el diagrama de Gantt.

En su origen, el diagrama de Gantt era una forma de representar gráficamente, mediante barras, las diferentes tareas que componen un proyecto y el tiempo estimado para su finalización. Este tipo de diagrama no indica las relaciones entre tareas ni las interdependencias, por lo que para aplicarlo en proyectos complejos se usan otras técnicas como la técnica de revisión y evaluación de programas o el método de la ruta crítica. Esta revisión de la representación original, que indica dependencias y relaciones entre tareas, es la que se conoce hoy en día como diagrama de Gantt. En adelante se denominará diagrama de Gantt a esta revisión del diagrama original.

El diagrama de Gantt es un diagrama bidimensional en el que se representa el tiempo en el tiempo en el eje horizontal y en el vertical el nombre de la tarea. Las dependencias se describen mediante una flecha desde el final de la tarea que genera la dependencia hasta el comienzo de la tarea dependiente.

En la ilustración que se encuentra a continuación se puede observar un diagrama de Gantt de un proyecto que comienza el día 7 de enero de 2021 y termina el 26 de febrero de ese mismo año. Este proyecto está compuesto por tres tareas principales, llamadas Tarea1, 2 y 3 y un milestone llamado Evento. Dos de ellas, las tareas 1 y 3 están compuestas por varias subtareas, lo que se representa mediante una barra negra. Se puede observar que Tarea 2 tiene una dependencia de Tarea 1.2 y Tarea 3 tiene una doble dependencia, por un lado, depende de que finalice Tarea 1.3 y por otro depende del *milestone,* que sucede el día 25 de enero.

ID	Name	Start	End
1	**Tarea 1**	**Thu 2021-01-07**	**Sat 2021-02-06**
1.1	Tarea 1.1	Thu 2021-01-07	Fri 2021-01-22
1.2	Tarea 1.2	Thu 2021-01-07	Wed 2021-01-27
1.3	Tarea 1.3	Thu 2021-01-07	Sat 2021-02-06
2	Evento	Mon 2021-01-25	Mon 2021-01-25
3	Tarea 2	Wed 2021-01-27	Fri 2021-02-26
4	**Tarea 3**	**Sat 2021-02-06**	**Fri 2021-02-26**
4.1	Tarea 3.1	Sat 2021-02-06	Fri 2021-02-26
4.2	Tarea 3.2	Sat 2021-02-06	Fri 2021-02-26

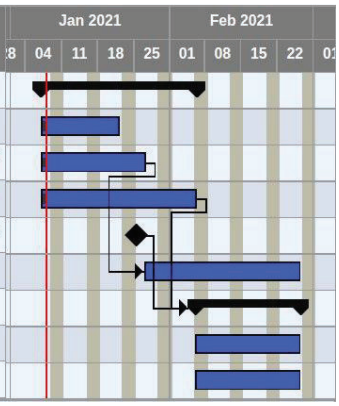


Ilustración 2 Estructura Diagrama Gantt

2.2 Estudio de alternativas

Como se menciona en la **Sección 2.1**, los diagramas de Gantt se han convertido en una herramienta muy extendida en la gestión de recursos y proyectos. Por este motivo muchas compañías, así como desarrolladores independientes han realizado herramientas software para la realización de esta clase de representaciones. Es por este motivo que no se va a desarrollar el software de creación de diagramas, en su lugar se va a utilizar una de las múltiples herramientas existentes.

Grandes compañías comercializan potentes herramientas de gestión de proyectos como Microsoft Project e infinidad de plantillas para Microsoft Excel. El principal inconveniente de estas alternativas, y gran parte de la motivación del proyecto, es que se trata de software propietario que requiere de licencia para su utilización. Sin embargo, también se han desarrollado gran cantidad de esta clase de herramientas gratuitas. De entre las últimas cabe destacar las siguientes cinco.

Sinnaps es una aplicación web con sede situada en Zaragoza. Se trata de un software de gestión de proyectos basado en la realización de diagramas de Gantt. Ofrece multitud de plantillas para distintos tipos de proyectos, lo que facilita su utilización. Como muchas otras herramientas de este tipo tiene soporte para gestión de recursos además de tareas. Cuenta con aplicación para sistema operativo Android. Las limitaciones que ofrece para el proyecto actual es que es una aplicación de uso gratuito por tiempo limitado y que requiere registro por parte de cada usuario que la utilice, además no cuenta con una API con la que poder usarla de forma programática.

OpenProj es otra importante alternativa de código abierto ampliamente utilizada a nivel mundial. Se trata de un software de gestión de proyectos multiplataforma. Su principal problema es que no se encuentra en desarrollo actualmente y no se actualiza ni presta soporte desde 2008.

Planner es otra gran aplicación para gestión de proyectos. Se trata de una aplicación desarrollada para el escritorio GNOME que nos permite la administración de tareas y recursos. Durante el estudio previo al desarrollo se perfiló como una de las herramientas más prometedoras, pero la incapacidad para conseguir ejecutarla y la falta de cualquier tipo foro o soporte hicieron que fuera descartada.

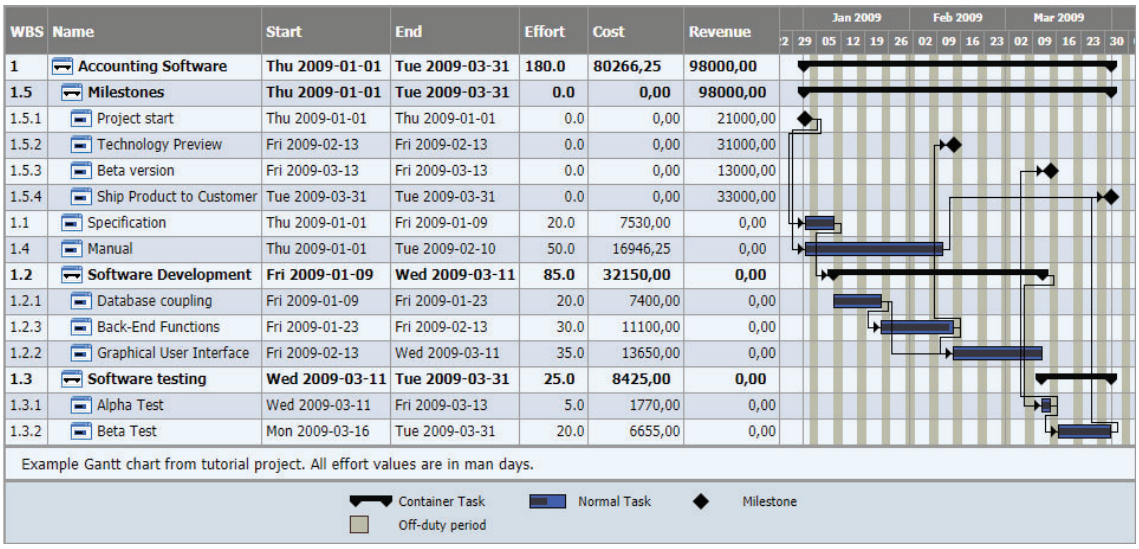
Cabe mencionar **NavalPlan**, una aplicación desarrollada como forma de impulso de la industria naval gallega. Por desgracia no se encuentra ya disponible.

TaskJuggler es una herramienta de gestión de proyectos llevada a cabo por Chris Schlaeger con la contribución de una comunidad que la mantiene en estado activo de desarrollo. El programa está publicado bajo la licencia GNU GPL2. Se hace una descripción detallada de este software en la **Sección 3.2**

2.2.1 Herramienta de gestión elegida y razones de elección

De todos los programar analizados en la **Sección 2.2**, el que mejor se adapta a los requisitos de este proyecto es el último, TaskJuggler. Se trata de un programa de código abierto que actualmente cuenta con dos versiones, una en desarrollo activo. La que está siendo desarrollada y cuenta con soporte es denominada TaskJuggler3. Se escogió porque permite el acceso programático, lo que hace que, una vez tratado el archivo exportados desde la herramienta de hojas de cálculo, sea fácil llamarlo para generar de forma automática el diagrama. La otra alternativa era TaskJuggler2 pero, además de que no cuenta con soporte y su desarrollo no se encuentra activo, el acceso se realiza mediante UI, por lo que no era interesante para este desarrollo.

Otra de las razones de elección fue que TaskJuggler genera los reportes en formato web. Los archivos generados son HTML, CSS y un archivo JavaScript lo que hace que prácticamente cualquier computador pueda abrirlo en un buscador de internet.



WBS	Name	Start	End	Effort	Cost	Revenue
1	**Accounting Software**	**Thu 2009-01-01**	**Tue 2009-03-31**	**180.0**	**80266,25**	**98000,00**
1.5	**Milestones**	**Thu 2009-01-01**	**Tue 2009-03-31**	**0.0**	**0,00**	**98000,00**
1.5.1	Project start	Thu 2009-01-01	Thu 2009-01-01	0.0	0,00	21000,00
1.5.2	Technology Preview	Fri 2009-02-13	Fri 2009-02-13	0.0	0,00	31000,00
1.5.3	Beta version	Fri 2009-03-13	Fri 2009-03-13	0.0	0,00	13000,00
1.5.4	Ship Product to Customer	Tue 2009-03-31	Tue 2009-03-31	0.0	0,00	33000,00
1.1	Specification	Thu 2009-01-01	Fri 2009-01-09	20.0	7530,00	0,00
1.4	Manual	Thu 2009-01-01	Tue 2009-02-10	50.0	16946,25	0,00
1.2	**Software Development**	**Fri 2009-01-09**	**Wed 2009-03-11**	**85.0**	**32150,00**	**0,00**
1.2.1	Database coupling	Fri 2009-01-09	Fri 2009-01-23	20.0	7400,00	0,00
1.2.3	Back-End Functions	Fri 2009-01-23	Fri 2009-02-13	30.0	11100,00	0,00
1.2.2	Graphical User Interface	Fri 2009-02-13	Wed 2009-03-11	35.0	13650,00	0,00
1.3	**Software testing**	**Wed 2009-03-11**	**Tue 2009-03-31**	**25.0**	**8425,00**	**0,00**
1.3.1	Alpha Test	Wed 2009-03-11	Fri 2009-03-13	5.0	1770,00	0,00
1.3.2	Beta Test	Mon 2009-03-16	Tue 2009-03-31	20.0	6655,00	0,00

Ilustración 3. Diagrama de Gantt creado mediante TaskJuggler

2.2.2 Herramienta de hoja de cálculo elegida

Dado que la finalidad del presente Trabajo de Fin de Grado es facilitar la creación de diagramas de Gantt a los usuarios, se ha buscado un método para introducir los datos que les sea fácil y a la vez familiar. Se ha llegado a la conclusión de que utilizar una herramienta de hojas de cálculo es una excelente alternativa para el conseguir el objetivo.

De entre las alternativas estudiadas ha habido dos que se ajustan al proyecto y a su filosofía: LibreOffice Calc y Google Sheets.

La primera es una herramienta de hojas de cálculo *open-source* multiplataforma

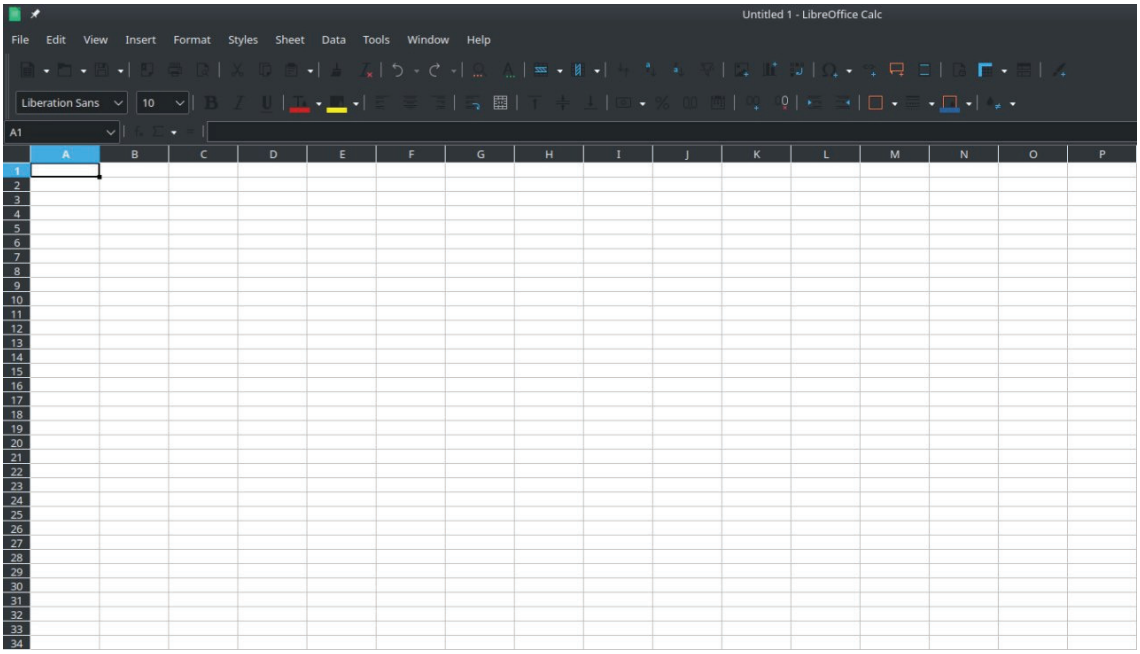


Ilustración 4 LibreOffice Calc

La segunda es una herramienta online de uso gratuito hasta el momento de publicación del presente trabajo. Se recomienda especialmente esta herramienta en ambientes colaborativos, ya que permite la edición de la hoja de cálculo por varias personas simultáneamente. Esto representa una ventaja frente a alternativas como LibreOffice Calc.

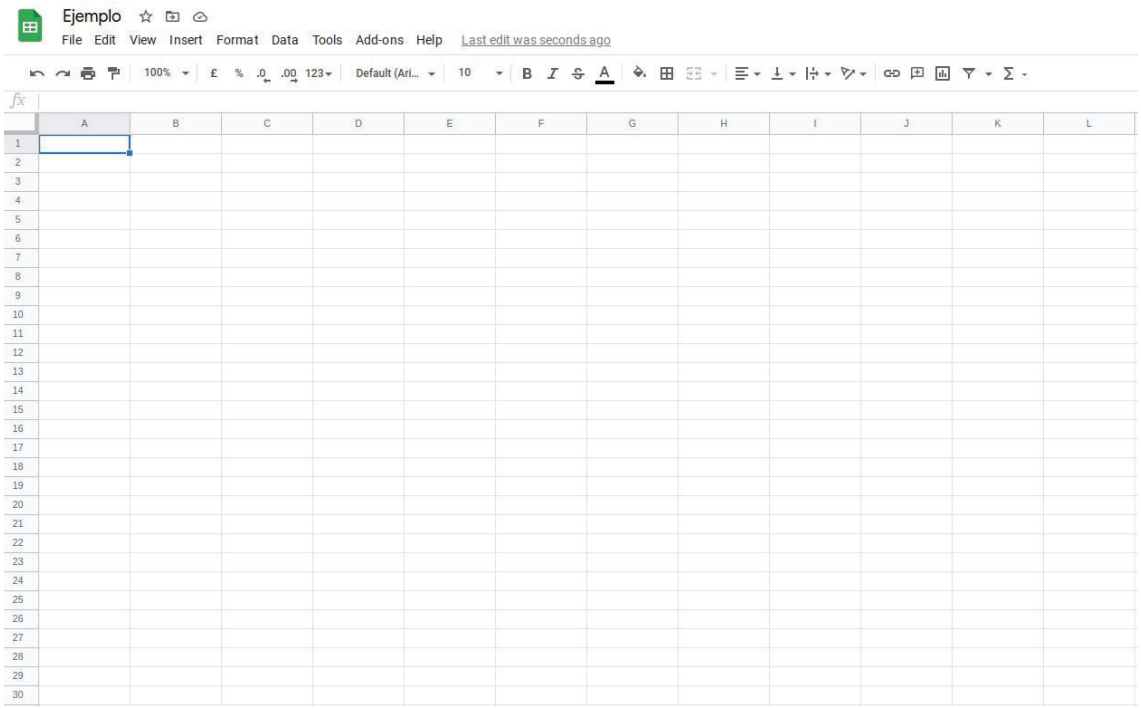


Ilustración 5 Google Sheets

2.3 Tecnologías utilizadas

En esta sección se describen las tecnologías empleadas para el desarrollo: el lenguaje de programación, el sistema de control de versiones, el *framework* de testeo y una herramienta de análisis estático del código llamada *SonarQube*.

2.3.1 Lenguaje Python 3.9

Python [1] es un lenguaje de programación interpretado, de alto nivel, dinámico y multiplataforma que busca la legibilidad del código. Es un lenguaje de programación multiparadigma administrado por la *Python Software Fundation*.

De acuerdo con el estudio realizado anualmente por TIOBE [2], Python se ha convertido en el segundo lenguaje de programación más usado a nivel global. Esto lo convierte en un lenguaje que probablemente sea conocido por otros desarrolladores, lo que permite que el código pueda ser examinado y extendido.

2.3.2 Sistema de control de versiones GIT

Los sistemas de control de versiones son software que ayudan a administrar las distintas versiones de productos desarrollados. En el sector del desarrollo de software se utilizan para mantener organizadas las distintas versiones que tiene uno o varios ficheros a lo largo de un desarrollo, de forma que sea fácil recuperarlos, modificarlos, eliminarlos o restaurarlos a conveniencia. Además, ayudan a coordinar equipos de desarrollo, permitiendo la comunicación referente al código entre los distintos desarrolladores.

En la actualidad los tres sistemas de control de versiones más utilizados son *Git, Subversion y Mercurial*. Y, de entre los tres, se ha elegido el primero porque el autor de este trabajo tenía experiencia previa.

"*Git es un software de control de versiones diseñado por Linus Torvalds, pensando en la eficiencia y la confiabilidad del mantenimiento de versiones de aplicaciones cuando éstas tienen un gran número de archivos de código fuente. Su propósito es llevar registro de los cambios en archivos de computadora y coordinar el trabajo que varias personas realizan sobre archivos compartidos*" [3]

Sobre este sistema de control de versiones se han desarrollado varios sistemas de hosting. De entre todas las opciones que ofrecen este servició se eligió GitHub como repositorio remoto debido a que ofrece licencia gratuita para estudiantes.

2.3.3 Analizador estático de código SonarQube

SonarQube es un analizador estático de código ofrecido como *open source*. Se trata de un programa que realiza de forma paralela varios estudios sobre el código bajo análisis en busca de bugs, de fallos de seguridad, fallos al seguir las convenciones del lenguaje o posibles mejoras que se pueden hacer sobre el código. Ayuda a mantener un código limpio y entendible por otros desarrolladores y ha sido integrado dentro del flujo de trabajo de desarrollo como herramienta de control continuo de la calidad. SonarQube muestra de forma visual los resultados de los análisis hechos sobre el código en desarrollo.

2.3.4 Framework de testeo Pytest

Se trata de uno de los framework más usados para testear programas desarrollados en Python. Algunas de las ventajas que ofrece es que puede ejecutar los test válidos para la librería estándar de testeo en Python, permite la ejecución de subconjuntos de test, tiene una sintaxis simple y su proceso de instalación es muy fácil. Además de todas estas ventajas es open source, lo que era requisito indispensable para el proyecto.

3 Diseño de AutoGantt

En este capítulo se describe el proceso de diseño de la herramienta AutoGantt: su diseño de alto nivel y su arquitectura software. Se analiza el software de gestión de proyectos subyacente, TaskJuggler, y las limitaciones impuestas por este.

3.1 Diseño de alto nivel

La herramienta está integrada por tres componentes:

Un **programa de hojas de cálculo** con capacidad para exportar como csv. Se ha elegido dicho formato por la facilidad que ofrece para trabajarlo, al tratarse de archivos planos de texto. En la **Sección 2.2.2** se indican que tanto LibreOffice Calc como Google Sheets han sido utilizadas indistintamente. Esto se debe a que el formato que tienen los archivos exportados como csv por ambas herramientas es muy similar. También se ha probado a utilizar Microsoft Excel, pero no se recomienda su uso, debido a que su formato por defecto es diferente al de las otras dos alternativas probadas, utiliza ";" en lugar de "," como separadores y no reconoce caracteres no ASCII, por lo que AutoGantt genera errores si se utiliza. Se puede configurar para forzar la codificación como UTF-8 y la utilización de "," como separador, pero queda fuera del alcance del proyecto y no se ha probado el correcto funcionamiento.

Una **herramienta de gestión de proyectos** que realice el diagrama de Gantt de forma automática, en este caso TaskJuggler.

Un programa que transforme la información obtenida del csv en el formato requerido por el software de gestión de proyectos. A este programa se le ha denominado **AutoGantt**.

Estos componentes se relacionan como se muestra en la ilustración que se muestra a continuación. Como se puede ver el formato que exporta el software de hojas de cálculo debe de ser csv y el que devuelve AutoGantt es un archivo con extensión .tjp, que es la extensión de los archivos de definición de proyecto de la herramienta TaskJuggler.

Ilustración 6 Diagrama de Componentes

Durante la ejecución del programa se pueden producir errores. En caso de que éstos se produzcan, se emitirá por la terminal un mensaje de error. Este mensaje puede estar producido por AutoGantt o por TaskJuggler.

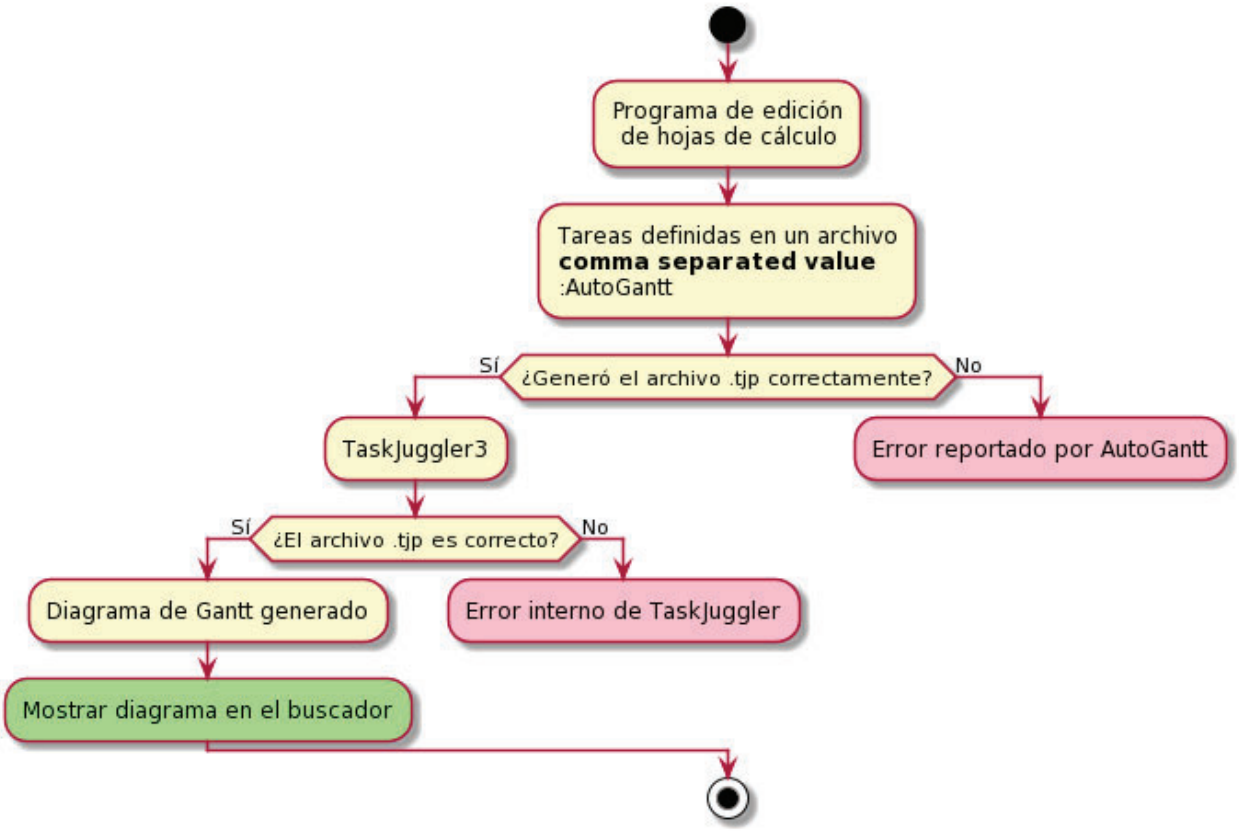


Ilustración 7 Diagrama de actividades de la herramienta

3.2 Descripción de TaskJuggler y restricciones impuestas

Dado que, como se ha dicho en la **Sección 2.1.1**, el programa que se encarga de la realización del diagrama es TaskJuggler, se ha realizado un estudio de las limitaciones impuestas por este software.

En primer lugar, ya que TaskJuggler está programado en Ruby, será necesario tener instalada al menos la versión 1.9.2 de este lenguaje en el ordenador donde se vaya a ejecutar el programa. También es necesario instalar RubyGems y la gema de TaskJuggler. Todo ello viene documentado en la sección de introducción de la página de documentación de TaskJuggler. [4]

En segundo lugar, el formato de los datos de salida de AutoGantt debe de ser interpretable por TaskJuggler. Para conseguirlo se ha realizado un estudio de la herramienta y el formato utilizado, así como de las restricciones impuestas por la misma. TaskJuggler utiliza archivos de descripción de proyectos para generar reportes, entre los que se encuentran los diagramas de Gantt. Estos archivos de descripción están divididos en lo que el creador llama "propiedades". Estas definen proyectos, tareas, recursos, etc. A su vez cada propiedad está definida por una serie de campos como son el nombre, el inicio o las dependencias que, en su conjunto, describen una de estas propiedades. Para la realización de este proyecto se han estudiado las propiedades milestone, tarea [5], reportes [6] [7]y proyecto [8] y se han analizado en busca de restricciones. La única restricción impuesta por TaskJuggler es que el campo ID de las tareas y del proyecto tienen que empezar por letra o "_" y continuar por cualquier secuencia de estos caracteres y números

Por último, con ayuda de los ejemplos [9], se estudió el formato e identación los archivos de definición para poder aplicarlo al archivo producido por AutoGantt.

3.3 Proceso de diseño y arquitectura software

Una vez realizado el estudio de las restricciones impuestas por TaskJuggler y teniendo claros los componentes del programa a desarrollar se ha realizado el proceso de diseño.

En primer lugar, se ha decidido hacer una herramienta invocada por consola de comandos y se ha hecho un diseño a bajo nivel de la herramienta. El diseño consiste en un script que invoque a AutoGantt y coordine la salida de éste con la entrada de TaskJuggler, cuya salida será el diagrama de Gantt deseado. Por este motivo, el diseño de alto nivel presentado en la **Sección 3.1** se ha transformado en una arquitectura software formada por 4 componentes:

1. **Google Sheets o LibreOffice Calc**: herramientas que permiten la introducción de datos manualmente en una hoja de cálculo y posibilitan la exportación de dicha información como csv.

2. **AutoGantt.sh**: se trata de un pequeño script escrito en bash, cuya funcionalidad es abstraer las rutas de los archivos, consiguiendo así que el módulo principal pueda ser llamado desde cualquier punto del árbol de directorios. Invoca a AutoGantt.py y, con el archivo resultante, invoca

a TaskJuggler. Además, se encarga de crear una nueva carpeta, cuyo nombre es “Gantt Results” para almacenar los archivos útiles para el usuario generados durante la ejecución del programa. Por último, tratará de abrir el HTML resultante con el buscador por defecto del sistema.

3. **AutoGantt.py**: es el módulo principal de la herramienta desarrollada. Se encarga de transformar la información recibida como csv para que pueda ser interpretada por el programa TaskJuggler. Este módulo orquestra el funcionamiento del resto de los módulos desarrollados, se describirá el funcionamiento de estos y su arquitectura interna en el **Capítulo 4**. También se encarga de comprobar el formato de determinados campos para cumplir con las restricciones descritas en la **Sección 3.2.**

4. **TaskJuggler3**: es una potente herramienta de gestión de proyectos que recibe el archivos de definición de proyectos con extensión “.tjp” y realiza reportes, entre los cuales se encuentra el diagrama de Gantt.

3.4 Descripción hoja de cálculo y del comma separated value (CSV)

Una vez estudiado el TaskJuggler, y en paralelo a al desarrollo, se ha diseñado una hoja de cálculo con las columnas necesarias para la definición de tareas y milestones y un campo para especificar la duración estimada del proyecto. Cada tarea viene definida por las columnas ID, NOMBRE, SUBTAREAS, INICIO, DURACIÓN, DEPENDENCIAS. A su vez, cada milestone está definido por: ID, NOMBRE, INICIO y DEPENDENCIAS.

Se adjunta la plantilla, además de en la carpeta Documents, en los enlaces públicos que se pueden encontrar a continuación.

https://upm365-my.sharepoint.com/:x:/g/personal/carlos_mateos_martin_alumnos_upm_es/EWIPQAqPEGdGupZFwSPd8DcBcgcKK9t7JTJy3UtgLD3nLg?e=0HyaLE

https://drive.google.com/file/d/1mR1sY0pMNLMPfncUgMsGACJXfsk0Vav0/view?usp=sharing

4 Desarrollo y testeo de AutoGantt

En este capítulo se describen los módulos desarrollados y su interacción, el proceso de testeo, la estructura de directorios y la documentación de AutoGantt.

4.1 Desarrollo módulos AutoGantt

En cuanto al desarrollo de AutoGantt.py y el resto de los módulos, hay que destacar que se ha dividido el código en módulos plenamente independientes, pudiendo ser reutilizados para otros proyectos y haciendo fácil la expansión de la funcionalidad del programa.

Hay que distinguir entre dos tipos de módulos: los que definen objetos y/o funciones de la herramienta, y los de pruebas, archivos cuyo nombre empieza por "test_" y de los cuales no se va a hacer un análisis en la memoria, se deja el código y la documentación en el repositorio de GitHub que se indica en el **Capítulo 1**.

La interacción entre los distintos módulos comienza con la ejecución del módulo principal, AutoGantt.py, que obtiene la ruta del archivo de donde se encuentran definidas las tareas y el nombre del proyecto.

A continuación, invoca al módulo Parse.py, el cual devuelve la duración del proyecto y una lista con todas las tareas, encapsuladas en objetos Tarea.

Acto seguido, invoca al módulo GestionTareas.py para obtener de un árbol de tareas del proyecto para. Por último, procesa la información contenida en dicho árbol para generar el archivo de definición para TaskJuggler.

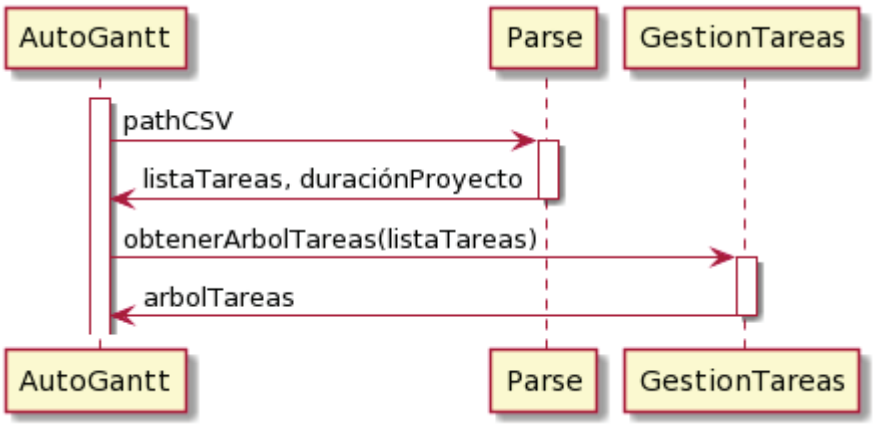


Ilustración 8 Diagrama de secuencia de AutoGantt

En el anterior diagrama se mostraban los componentes que realizan la mayor parte de las funciones del software, pero no solo se han desarrollado esos módulos. Tarea, Proyecto y Árbol son módulos que nos ayudan a modelizar entidades y encapsular sus datos. El módulo Parse genera objetos Tarea para poder pasárselos a el módulo AutoGantt. El módulo GestionTareas realiza un árbol a partir de la lista de Tareas que le pasa AutoGantt. Por último, el propio módulo principal hace una instancia de objeto Proyecto, que encapsula, entre otros, los datos de las tareas y su estructura jerárquica en forma de árbol. Es así como el programa es capaz de representar todas las tareas descritas en el comma separated value en un formato que TaskJuggler es capaz de interpretar.

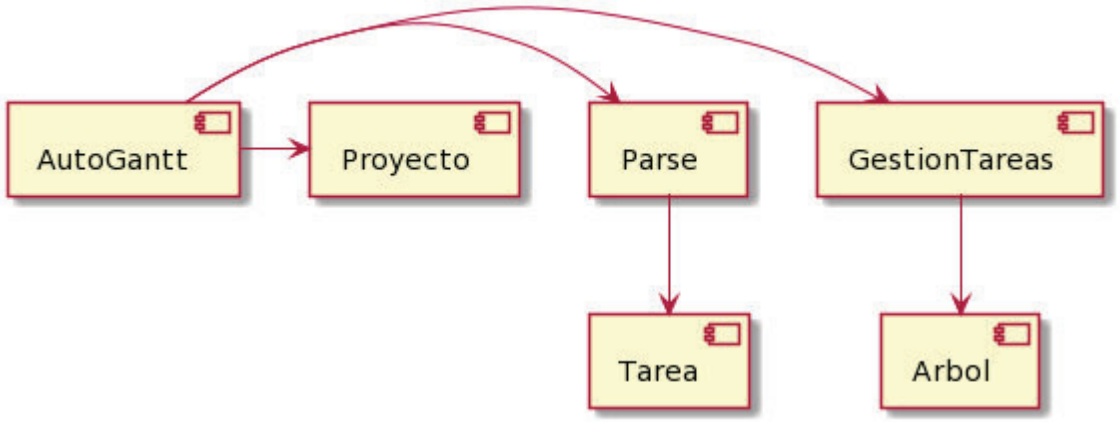


Ilustración 9 Diagrama de Componentes de AutoGantt

4.2 Test

Durante el desarrollo de AutoGantt se han realizado test de varias maneras.

En primer lugar, ha realizado testeo exploratorio consistente en la ejecución de pruebas de funcionamiento del código. Dichas pruebas consistían en la ejecución de llamadas a las distintas funciones del programar comprobando visualmente los resultados. Para este fin, se han desarrollado funciones que permiten la visualización de estructuras de datos como los árboles o de los tipos de los campos de los objetos. Estos test se escribían, ejecutaban, comprobaban y borraban.

En segundo lugar, se diseñó y codificó un conjunto de test unitarios para cada uno de los módulos Python desarrollados. Este conjunto de test fue pensado para realizar al menos una ejecución de cada una de las líneas de código del proyecto. El código de dichas pruebas se puede encontrar en el repositorio, dentro de la carpeta AutoGantt/Code/.

Por último, se realizaron una serie de pruebas de integración. Se ejecutó la herramienta con listas de tareas y milestones hechas siguiendo las instrucciones suministradas en la **Sección 4.4.1**. Gracias a la ejecución de dichas pruebas se pudieron corregir errores menores y se pudo comprobar que las instrucciones comunicaban correctamente los pasos a seguir.

4.3 Descripción de la estructura de directorios del proyecto

En cuanto a la estructura de directorios del proyecto hay cuatro carpetas cuyo contenido está bien diferenciado:

1. Code: contiene todo el código del proyecto.
2. Program_Files: contendrá el archivo de definición de tareas (.csv) y el archivo de definición de proyecto para TaskJuggler (.tjp).
3. Documents: contiene la plantilla a rellenar y una subcarpeta, llamada Modules_documentation, con la documentación de cada uno de los módulos.
4. Examples: almacena algunos archivos que se han dejado a modo de guía para futuros usuarios.

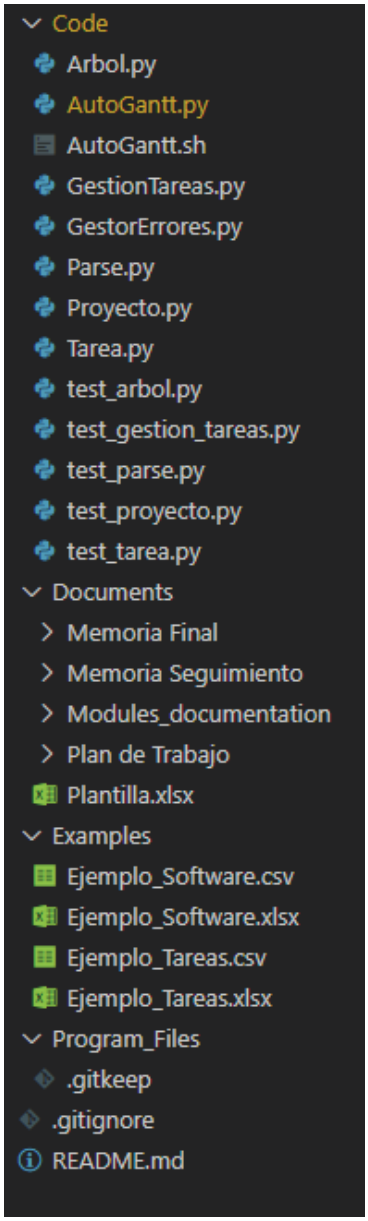


Ilustración 10 Estructura de directorios del proyecto

4.4 Documentación

La documentación relativa al código desarrollado se encuentra en formato HTML en la carpeta AutoGantt/Documents/ Modules_documentation. A continuación, se describen las instrucciones de uso de la herramienta.

4.4.1 Instrucciones de uso

Para el correcto funcionamiento de la herramienta se deben seguir los siguientes pasos:

1. Añadir las tareas en la hoja de cálculo AutoGantt/Program_Files/Plantilla.xlsx de acuerdo con las siguientes reglas:

1.1 No modificar el orden ni el nombre de las columnas (celdas en negrita)

1.2 Para cada tarea del proyecto rellenar una fila de la siguiente forma:

1.2.1 Rellenar el campo ID con una cadena de caracteres que empiece por letra. Será necesario para rellenar los campos subtareas y dependencias. Se generará uno automáticamente si no lo rellena

1.2.2 Rellenar la celda nombre (obligatorio)

1.2.3 Rellenar la celda subtareas con un id** o la concatenación de varios id separados por & (ej. id1&id2&id3)

1.2.4 Rellenar la celda inicio si tiene una fecha de inicio fija. La fecha debe tener formato AAAA-MM-DD

1.2.5 Rellenar la celda duración con alguno de los siguientes formatos (no especificar para las tareas con subtareas):

- #d: Número de días que dura la tarea (ej. "10d" para una tarea que dure 10 días)

- #w: Número de semanas que dura la tarea (ej. "10w" para una tarea que dure 70 días)

- #m: Número de días que dura la tarea (ej. "2m" para una tarea que dure 60 días)

- #y: Número de días que dura la tarea (ej. "1y" para una tarea que dure 365 días)

- #: Número de días que dura la tarea (ej. "10" para una tarea que dure 10 días)

1.2.6 Rellenar la celda dependencias con un id** o la concatenación de varios id separados por & (ej. id1&id2&id3)

**Son válidos tanto los id de tareas como los de milestones.

2. Añadir los milestones en el mismo archivo donde se definen las tareas de acuerdo con las siguientes reglas:

2.1 No modificar el orden ni el nombre de las columnas (celdas en negrita)

2.2 Para cada tarea del proyecto rellenar una fila de la siguiente forma:

2.2.1 Rellenar la celda ID con una cadena que empiece por letra (obligatorio).

2.2.2 Rellenar la celda nombre (obligatorio)

2.2.3 Rellenar la celda uno de los campos restantes (Inicio y Dependencias) siguiendo las normas 1.2.4 y 1.2.6

3. Rellenar la celda a la derecha de duración (valor por defecto 365) con la duración estimada para el proyecto siguiendo alguno de los formatos descritos en el apartado 1.2.5

4. Exportar como comma-separated value (.csv) en la carpeta AutoGantt/Program_Files.

5. Renombrar el archivo .csv al del título del proyecto. (Se utilizará a la hora de generar reportes)

6. Ejecutar el archivo AutoGantt/Code/TFG.sh (cambiar los permisos si fuera necesario). Los archivos resultantes de la ejecución quedarán en el mismo directorio desde el que se ejecutó el script en una carpeta llamada Gantt_Results.

5 Resultados y conclusiones

El presente Trabajo de Fin de Grado venía a cubrir la necesidad de generar diagramas de Gantt de forma automática, facilitando la labor a todos los usuarios que necesitan crear esta clase de diagramas. También buscaba abstraer la complejidad del formato de algunas herramientas de creación de diagramas utilizando una interfaz más agradable para los usuarios. Todo esto se buscaba hacerlo utilizando únicamente programas de código abierto.

Como resultado del trabajo desarrollado y habiendo cumplido los objetivos marcados en la **Sección 1.2**, se ha obtenido una herramienta, AutoGantt, que satisface la necesidad que nos llevó a plantearlo. Todo el proyecto permanecerá en el repositorio indicado al comienzo del presente trabajo y podrá ser utilizado y modificado por cualquier miembro de la comunidad universitaria de la Universidad Politécnica de Madrid.

Como conclusión y desde un punto de vista personal, me ha resultado una de las asignaturas más interesantes y provechosas del grado. Durante la realización del presente TFG he trabajado y aprendido mucho sobre Python, lenguaje de programación crucial en estos tiempos. Pero no solo eso, también he aprendido muchas herramientas software, como SonarQube, lo que también me ha obligado a aprender sobre administración de sistemas. Además, he aprendido mucho sobre el proceso de desarrollo software, desde la definición de requisitos hasta el testeo pasando por el desarrollo o el diseño de software. Es la primera vez que he realizado en solitario todas estas tareas, y es por esto por lo que considero que ha sido una experiencia muy enriquecedora.

5.1 Líneas futuras

Dado el carácter limitado de la herramienta desarrollada, es fácilmente ampliable en líneas de trabajo futuras, tales como:

La inclusión de una base de datos como fuente de la información en lugar de un programa de hojas de cálculo. Esto haría mucho más fácil, para usuarios avanzados, el poder tener la información en una computadora remota y el software trabajando en local con dicha información. Además, permitiría la generación automática de un diagrama de Gantt que tuviera la información actualizada. Esto último podría conseguirse asociando AutoGantt a un disparador de la base de datos, de tal forma que cuando cambiase cualquier dato se ejecutara la herramienta con la información actualizada.

Por otra parte, se podría plantear el despliegue como aplicación web ya que los reportes generados son archivos HTML, lo que hace que sea fácil mostrarlos al usuario de forma más limpia, ya que no implica abrir el buscador expresamente para mostrar los resultados.

Por último, dado que el software subyacente es una herramienta de gestión de proyectos, otra posible línea de trabajo futuro sería expandir las funcionalidades de AutoGantt más allá de la generación de diagramas de Gantt, incluyendo datos financieros asociados al proyecto, asociando recursos y personal a las tareas y llevando una contabilidad de las horas trabajadas por cada empleado en el proyecto.

6 Bibliografía

[1] «Python,» Wikipedia, 2020. [En línea]. Available: https://es.wikipedia.org/wiki/Python. [Último acceso: 12 11 2020].

[2] «TIOBE Index for November 2020,» TIOBE, 11 2020. [En línea]. Available: https://www.tiobe.com/tiobe-index/. [Último acceso: 12 11 2020].

[3] «Git,» Wikipedia, 2020. [En línea]. Available: https://es.wikipedia.org/wiki/Git. [Último acceso: 12 11 2020].

[4] TaskJuggler, «TaskJuggler.org,» [En línea]. Available: https://taskjuggler.org/tj3/manual/index.html. [Último acceso: Octubre 2020].

[5] TaskJuggler, «TaskJuggler.org,» [En línea]. Available: https://taskjuggler.org/tj3/manual/task.html. [Último acceso: Octubre 2020].

[6] TaskJuggler, «TaskJuggler.org,» [En línea]. Available: https://taskjuggler.org/tj3/manual/textreport.html. [Último acceso: Octubre 2020].

[7] TaskJuggler, «TaskJuggler.org,» [En línea]. Available: https://taskjuggler.org/tj3/manual/taskreport.html. [Último acceso: Octubre 2020].

[8] TaskJuggler, «TaskJuggler.org,» [En línea]. Available: https://taskjuggler.org/tj3/manual/project.html. [Último acceso: Octubre 2020].

[9] TaskJuggler, «TaskJuggler.org,» [En línea]. Available: https://taskjuggler.org/examples.html. [Último acceso: Octubre 2020].

[10] «Gantt Chart,» Wikipedia, [En línea]. Available: https://en.wikipedia.org/wiki/Gantt_chart. [Último acceso: 12 11 2020].

Este documento esta firmado por

	**Firmante**	CN=tfgm.fi.upm.es, OU=CCFI, O=Facultad de Informatica - UPM, C=ES
	Fecha/Hora	Mon Jan 25 13:10:26 CET 2021
	Emisor del Certificado	EMAILADDRESS=camanager@fi.upm.es, CN=CA Facultad de Informatica, O=Facultad de Informatica - UPM, C=ES
	Numero de Serie	630
	Metodo	urn:adobe.com:Adobe.PPKLite:adbe.pkcs7.sha1 (Adobe Signature)