Plan de Gestión del Proyecto de Software

GII + GADE

Pablo Castaño Urías

Moisés García Álvarez

Silvia Pascual Castillo

Paula Sestafe Zamorano

Universidad Rey Juan Carlos

Hoja de revisión

NOMBRE	FECHA	DESCRIPCIÓN
Paula Sestafe	3/04/2018	Realización del Diagrama de Gantt
Silvia Pascual	3/04/2018	Comienzo de la realización del Plan de Proyecto Finalizado apartado I (PGPS) Realización del Diagrama de Pert
Pablo Castaño	4/04/2018	Comienzo de la realización del Plan de Proyecto Finalizado apartado I (PGCS)
Silvia Pascual	4/04/2018	Finalizado apartado II (PGPS) Finalizado apartado III (PGPS)
Pablo Castaño	5/04/2018	Finalizado apartado II (PGCS) Finalizado apartado III (PGCS)
Silvia Pascual	5/04/2018	Finalizado apartado IV (PGPS)
Pablo Castaño	6/04/2018	Finalizado apartado IV (PGCS) Finalizado apartado V (PGCS) Finalizado apartado VI (PGCS)
Silvia Pascual	6/04/2018	Finalizado apartado V (PGPS)
Moisés García	10/04/2018	Verificación previa de la correcta ejecución del código
Moisés García	12/04/2018	Verificación previa del cumplimiento de los requisitos
Moisés García	16/04/2018	Añadir la opción de reiniciar el juego
Paula Sestafe	16/04/2018	Mostrar las minas que quedan por descubrir y el tiempo de la partida
Paula Sestafe	16/04/2018	Adaptar el código para nivel principiante
Paula Sestafe	16/04/2018	Incluir en el mensaje de

		ganar un campo para añadir el nombre del jugador
Paula Sestafe	17/04/2018	Adaptar el código para nivel intermedio
Moisés García	17/04/2018	Permitir guardar la partida actual y poder recuperarla
Paula Sestafe	18/04/2018	Adaptar el código para nivel experto
Paula Sestafe	19/04/2018	Adaptar el código para nivel personalizado
Paula Sestafe	19/04/2018	Crear menú que permita acceder a las diferentes modalidades del juego
Moisés García	20/04/2018	Guardar los 10 mejores tiempos en cada dificultad
Moisés García	21/04/2018	Documentación del código
Moisés García	21/04/2018	Verificación posterior de la correcta ejecución del código
Moisés García	23/04/2018	Verificación posterior del cumplimiento de los requisitos
Paula Sestafe	24/04/2018	Realización del PowerPoint

Prefacio

El objetivo de esta práctica es documentar, de manera adecuada y de acuerdo a lo estudiado en la asignatura de Ampliación de Ingeniería de Software y atendiendo a los estándares IEEE correspondientes en cada caso, el proceso de desarrollo de software de una aplicación del popular juego de Buscaminas.

Nuestro trabajo englobará toda la documentación previa a la codificación, además del desarrollo de la aplicación propiamente dicha siguiendo las pautas especificadas en el enunciado de la práctica y que se explicarán bastamente a lo largo del presente documento. También se incluirá, aunque no desarrollado como correspondería, las decisiones de mantenimiento que se tendrían que llevar a cabo para asegurar la persistencia en el tiempo del juego.

El PGPS está dirigido, primeramente, a todos los integrantes del proyecto para su consulta; pero también a los profesores de la asignatura para la evaluación de la práctica.

Tabla de contenidos

Lista de figuras

Lista de tablas

1. Introducción

1.1. Visión General del proyecto.

Con este proyecto queremos conseguir una versión optimizada del popular juego Buscaminas. A continuación se explicarán los datos más relevantes del proyecto de manera somera, siendo todos ellos desarrollados con mayor profundidad en futuros apartados de este documento.

Partiendo de un código base inicial proporcionado por los profesores, los pasos a seguir serán los siguientes:

- Verificación de su correcta ejecución.
- Verificación del cumplimiento de las siguientes especificaciones:
    - Crear un tablero de 30x30 casillas cuyo contenido es aleatorio y permanece oculto al usuario. Cada casilla debe contener, o bien una mina, o bien un número indicando la existencia de minas en casillas adyacentes a esta, o bien una casilla vacía.
    - Posibilitar el desarrollo de la partida utilizando el ratón del ordenador: el botón izquierdo servirá para descubrir las casillas en las que se haga clic, mientras que el botón derecho servirá para marcar esa casilla como posible ubicación de una mina.
    - Cuando el usuario haga click en una casilla, ocurrirá una de las siguientes cosas:
        - Si en la casilla había una mina, se acabará la partida.
        - Si en la casilla no había una mina, se desvelarán todas las casillas adyacentes vacías y con número.
    - La partida se considerará ganada cuando se hayan desvelado todas las casillas sin mina.
    - Al ganar o perder se mostrará un ventana con un mensaje que cerrará el juego al pulsar aceptar. En caso de haber ganado esa ventana mostrará el tiempo que se ha tardado.
- Modificación del código para incluir las siguientes mejoras:

- Añadir una opción en el menú que permita reiniciar el juego en cualquier momento, sin tener que cerrar y volver a ejecutar el programa de nuevo.
- Mostrar en la pantalla el número de minas que quedan por ser descubiertas, además del tiempo desde que se inició la partida.
- Permitir al usuario elegir diferentes niveles de dificultad:
  - Principiante (tablero 10x10 casillas y 10 minas).
  - Intermedio (tablero 16x16 casillas y 40 minas).
  - Experto (tablero 32x16 casillas y 99 minas).
  - Personalizado (permite definir el tamaño del tablero y la cantidad de minas).
- En el mensaje que se muestra por defecto al ganar la partida, incluir un campo para que el usuario introduzca su nombre o nick, y tenga la posibilidad de guardar o no su tiempo.
- Dar la opción al usuario de poder guardar el estado de la partida en cualquier momento, almacenando esta en un fichero que se podrá cargar cuando lo desee.
- Guardar los 10 mejores tiempos del nivel Principiante, Intermedio y Experto.
- Añadir un menú que incluya todos los requerimientos antes mencionados y por el que el usuario se pueda mover de forma fácil e intuitiva.

- Verificación de la correcta ejecución y cumplimiento de las especificaciones arriba mencionadas.
- Entrega al cliente, o en este caso, de nuevo, a los profesores.

Para ello contaremos con las siguientes herramientas y software: GitHub, NetBeans y el paquete de Microsoft Office. El trabajo se repartirá de manera equitativa entre los cuatro miembros del equipo para lograr alcanzar los objetivos en el tiempo previsto para ello.

Como se ha dicho anteriormente, el resto del documento pasa a detallar todo lo que pudiera ser pertinente para entender qué se quiere conseguir y cómo se va a realizar.

1.2. Productos finales.

El producto final que debe ser entregado constará del código fuente de la aplicación, además de un ejecutable que permita probar el juego, y la documentación necesaria para entender el código que se habrá generado con JavaDoc.

Además de la aplicación, será necesario entregar la siguiente documentación:

- Plan para la Gestión de Proyectos Software.
- Plan de Gestión de Configuración del Software.
- Presentación Power Point en formato PDF que recoja de forma resumida el trabajo realizado y que será utilizado como guión y material de apoyo para la presentación.

Todo ello será comprimido en una carpeta zip y entregado a través del Aula Virtual en el enlace proporcionado para tal efecto antes del 25 de abril a las 23:55 horas.

1.3. Evolución del Plan de Proyecto.

El proceso que seguirá el proyecto será el siguiente: tras una primera reunión para leer atentamente el enunciado de la práctica y establecer los objetivos a alcanzar, será necesario determinar el calendario al cual atenerse, el ritmo de trabajo y el reparto del mismo.

Uno de los primeros pasos debería ser la realización de los diagramas de Gantt y Pert, puesto que en ellos se especifica claramente las tareas a realizar, el tiempo asignado para ellas y las dependencias existentes entre unas y otras. Se puede encontrar más información sobre estos diagramas en los apartados 2.1., 5.1. y 5.2. del presente documento.

Una vez hecho esto, cada uno de los integrantes trabajaremos en el proyecto de forma paralela, ya sea juntos en salas habilitadas para ello como por separado en nuestras propias casas. Estas sesiones conjuntas serán especialmente útiles para resolver dudas y dificultades, comunicar los cambios realizados ya sea en la documentación como en el código, y acordar decisiones importantes que afecten a la totalidad de la práctica.

La comunicación, por tanto, será vital en el desarrollo de este proyecto, y en ningún caso se dejará a un miembro solo con una parte del mismo, sino que se tratará de hacer *feedback* para evitar posibles inconsistencias, errores subsanables o incongruencias.

Por último, destacar el uso de GitHub para la evolución de los cambios del desarrollo software. La explicación a esta herramienta se puede encontrar en posteriores apartados de este plan, y mucho más extensamente comentada, en el PGCS.

1.4. Documentos de referencia.

- IEEE 1058.1- 1987. Norma a partir de la cual hemos estructurado nuestro PGPS.
- Plan de Gestión de Configuración del Software, siguiendo la norma IEEE 828-2005.
- Contrato de mantenimiento.
- Plan de mantenimiento.
- Formulario de petición de mantenimiento.
- Informe de cambios del software.

1.5. Definiciones y acrónimos.

- **PGPS:** Plan para la Gestión de Proyectos Software.
- **PGCS:** Plan de Gestión de Configuración del Software.
- **JavaDoc:** Es una herramienta de Java que permite generar documentación básica para el programador a partir del código fuente.
- **IEEE (*Institute of Electrical and Electronics Engineers*):** Instituto de Ingeniería Eléctrica y Electrónica. Organización mundial líder en la creación de estándares.
- **IDE (*Integrated Development Environment*):** Entorno de desarrollo integrado. Aplicación informática que facilita al programador el desarrollo de software.
- **EDT (Estructura de descomposición del trabajo):** Del inglés WBS (*Work Breakdown Structure*), es una herramienta que consiste en la descomposición jerárquica del trabajo a ser realizado por el equipo de proyecto.
- **Github:** Plataforma que permite el alojamiento de proyectos en la nube y la colaboración entre distintos desarrolladores.
- **Proyecto:** Conjunto de actividades planificadas, ejecutadas y supervisadas que, con recursos finitos, tiene como objetivo crear un producto o servicio único.

2. Organización del Proyecto

2.1. Modelo de procesos.

EDT	Nombre de tarea	Duración	Comienzo	Fin
A	Inicio	0 días	vie 3/23/18	vie 3/23/18
B	**Proyecto**	**18 días**	**mar 4/3/18**	**mar 4/24/18**
B.1	Inicio	0 días	mar 4/3/18	mar 4/3/18
B.2	**Planificación y documentación**	**4 días**	**mar 4/3/18**	**vie 4/6/18**
B.2.1	Realización de la documentación	4 días	mar 4/3/18	vie 4/6/18
B.3	Fin	0 días	vie 4/6/18	vie 4/6/18
B.4	Inicio	0 días	mar 4/10/18	mar 4/10/18
B.5	**Verificar inicial del código**	**4 días**	**mar 4/10/18**	**vie 4/13/18**
B.5.1	Verificar que el código se ejecuta correctamente	2 días	mar 4/10/18	mié 4/11/18
B.5.2	Verificar que el código cumple los requisitos especificados	2 días	jue 4/12/18	vie 4/13/18
B.6	Fin	0 días	vie 4/13/18	vie 4/13/18
B.7	Inicio	0 días	lun 4/16/18	lun 4/16/18
B.8	**Mejora del código**	**5 días**	**lun 4/16/18**	**vie 4/20/18**
B.8.1	Añadir la opción de reiniciar el juego	1 día	lun 4/16/18	lun 4/16/18
B.8.2	Mostrar las minas que quedan por descubrir y el tiempo de la partida	1 día	lun 4/16/18	lun 4/16/18
B.8.3	**Crear niveles de dificultad**	**4 días**	**lun 4/16/18**	**jue 4/19/18**
B.8.3.1	Adaptar el código para nivel principiante	1 día	lun 4/16/18	lun 4/16/18
B.8.3.2	Adaptar el código para nivel intermedio	1 día	mar 4/17/18	mar 4/17/18
B.8.3.3	Adaptar el código para nivel experto	1 día	mié 4/18/18	mié 4/18/18
B.8.3.4	Adaptar el código para nivel personalizado	1 día	jue 4/19/18	jue 4/19/18
B.8.4	Guardar los 10 mejores tiempos en cada dificultad	1 día	vie 4/20/18	vie 4/20/18
B.8.5	En el mensaje de ganar incluir un campo para añadir el nombre del jugador	1 día	lun 4/16/18	lun 4/16/18
B.8.6	Permitir guardar la partida actual y poder recuperarla	2 días	mar 4/17/18	mié 4/18/18
B.8.7	Crear menú que permita acceder a las diferentes modalidades del juego	1 día	jue 4/19/18	jue 4/19/18
B.9	Fin	0 días	vie 4/20/18	vie 4/20/18
B.10	Inicio	0 días	sáb 4/21/18	sáb 4/21/18
B.11	**Verificación final del código**	**4 días**	**sáb 4/21/18**	**mar 4/24/18**
B.11.1	Verificar el buen funcionamiento del código	2 días	sáb 4/21/18	dom 4/22/18
B.11.2	Verificar que el código se ajusta a los requisitos	2 días	lun 4/23/18	mar 4/24/18
B.12	Fin	0 días	mar 4/24/18	mar 4/24/18
C	Fin	0 días	mar 4/24/18	mar 4/24/18

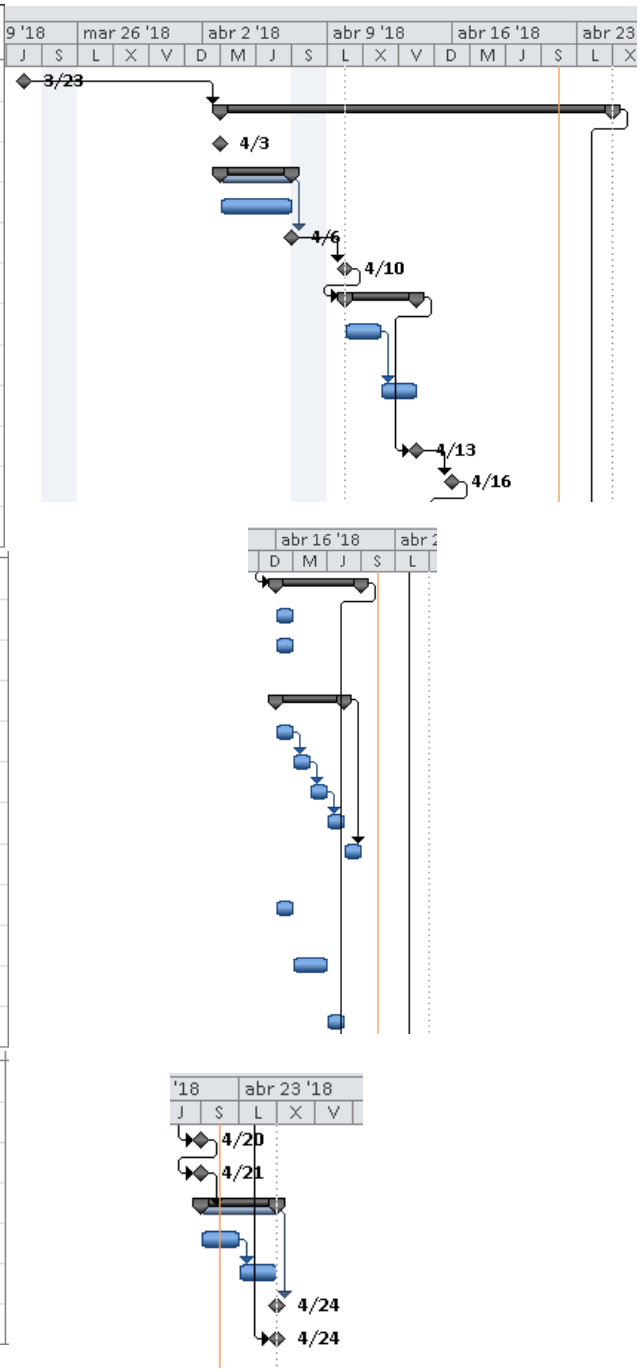

Figura 1. Diagrama de Gantt

A continuación presentamos el diagrama de Pert. En el apartado 5.2. volveremos sobre ello y desarrollaremos con más profundidad la división en paquetes de trabajo y las dependencias existentes entre ellos, y cómo a partir de ello hemos generado esta figura.

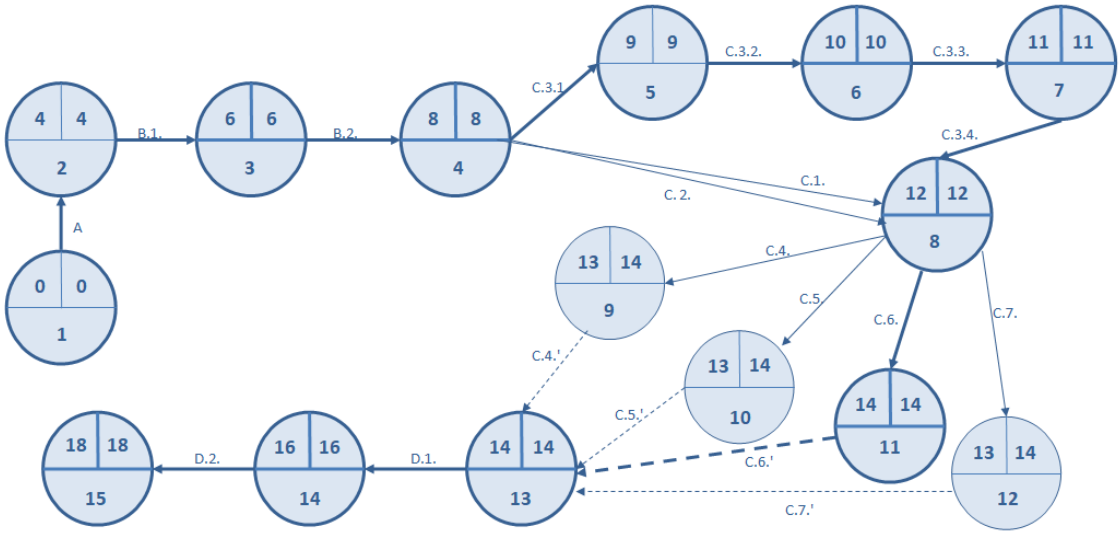

Figura 2. Diagrama de Pert

2.2. Estructura organizativa.

En este caso, hemos querido describir la estructura organizativa de manera visual a través de un EDT, o estructura de descomposición del trabajo.

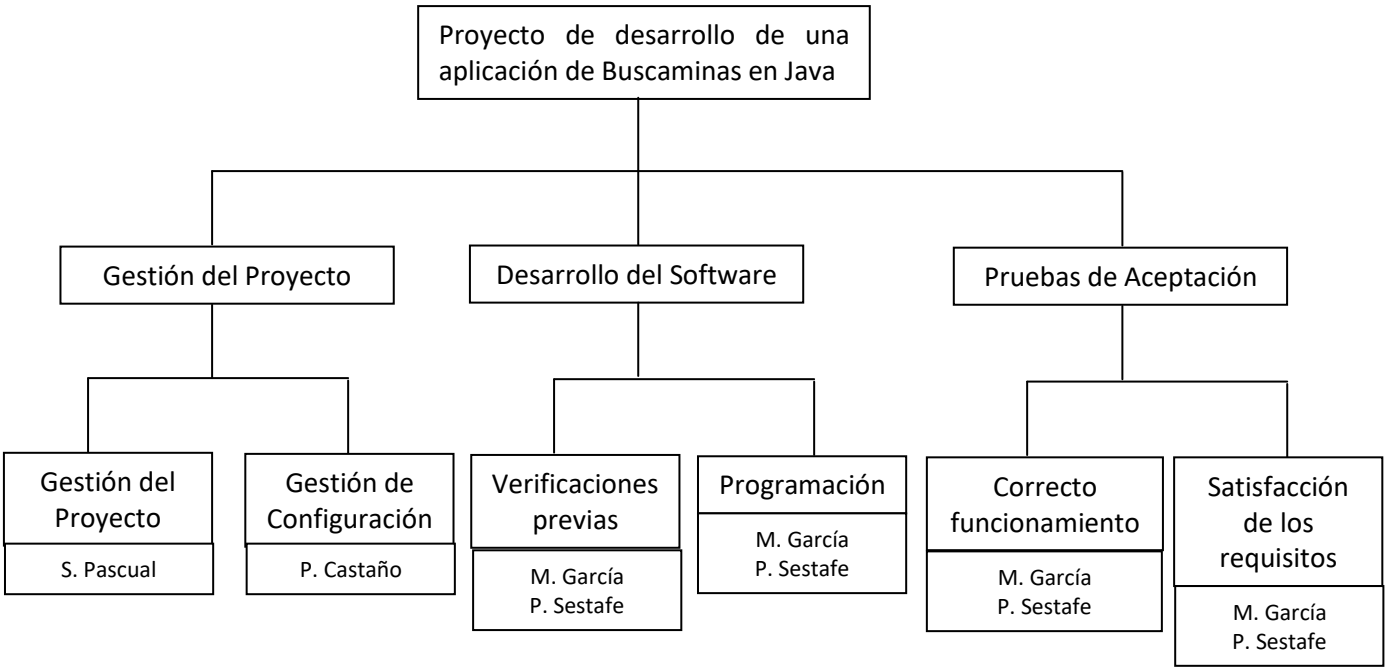

Figura 3. Estructura de descomposición del trabajo

La jerarquía de la organización se estructura de la siguiente manera:

- **Líder de Gestión de Proyecto.** Silvia Pascual.
- **Líder de Gestión de Configuración.** Pablo Castaño.
- **Líder de Desarrollo Software.** Paula Sestafe.
- **Líder de Pruebas.** Moisés García.
- **Líder en la Presentación del PowerPoint.** Paula Sestafe.

Las responsabilidades de cada uno han sido repartidas de manera informal en una primera reunión: la prioridad para el buen hacer de la práctica es que la parte del

desarrollo de software quedara a manos de aquellos que más manejo tenían con el lenguaje y los requisitos que se pedían.

Por supuesto, esto no significa que el primer reparto permanezca inalterable, pudiendo pasar un integrante que se ocupaba de documentación a pruebas de software, o un integrante que estaba codificando a ayudar con algún documento o preparar el PowerPoint.

Al respecto del PowerPoint, de cara a la presentación, consideramos que lo más sencillo para dividir en partes el total de la exposición es que cada uno se ocupe de contar aquello que es más cercano a la parte del trabajo que ha desarrollado en este mes. Por supuesto, esto no será objeto de que algún integrante sea totalmente ajeno a las partes que no le correspondían, simplemente, el nivel de seguridad y detalle que cada uno puede aportar explicando tareas de las que se ha ocupado es mayor, y por lo tanto, se puede asegurar una mayor calidad de la presentación.

En cualquier caso, se puede encontrar información más detallada sobre las responsabilidades de cada miembro en el apartado 2.4. del presente documento.

2.3. Fronteras e interfaces organizativas.

Este proyecto ha sido realizado en su totalidad por los cuatro integrantes que conforman el equipo de trabajo, cuyas responsabilidades se detallan en el apartado siguiente. Ninguna tarea ha sido subcontratada ni ha sido necesaria interacción alguna con cualquier otra entidad organizativa.

En cuanto al cliente, estando este representado por la figura del profesor de la asignatura de Ampliación de Ingeniería del Software, se ha encargado de proporcionarnos el código fuente a partir del cual se tendrán que hacer las modificaciones y mejoras pertinentes en cuanto a los requisitos que han sido pedidos. Al término de la práctica, la aplicación mejorada será entregada al mismo.

Figura 4. Diagrama de frontera

2.4. Responsabilidades.

Las responsabilidades de los distintos miembros del equipo trabajo se han dividido de forma acordada en una reunión, tratando de conseguir que estas fueran equilibradas en cuanto a dificultad, que todo el mundo tuviera trabajo en todo momento y que cada uno pudiera encargarse de aquello en lo que podía encontrarse con menos dificultades. Así:

- **Pablo Castaño:** Redacción del Plan de Gestión de Configuración.
- **Moisés García:** Codificación y pruebas, manejo con GitHub. Realización del JavaDoc.
- **Silvia Pascual:** Redacción del Plan de Proyecto.
- **Paula Sestafe:** Codificación y pruebas, manejo con GitHub. Realización del diagrama de Gantt. Realización del Power Point.

Cabe destacar que, a pesar de esta división de tareas, ideada para conseguir avanzar más rápido, el grupo estará en constante contacto para aclarar dudas y tomar decisiones que afecten a la totalidad del proyecto. Para ello se utilizarán herramientas de mensajería rápida tales como WhatsApp o Skype, y reuniones semanales en un espacio en común, siendo este, probablemente, salas de la Biblioteca del Campus de la Universidad.

3. Proceso de Gestión

3.1. Objetivos y prioridades de gestión.

En este sentido, es imprescindible destacar la importancia de la comunicación entre todos los integrantes del grupo de trabajo. No consideramos necesario generar informes formales de ningún tipo en el desarrollo del proyecto, dado que el grupo es reducido y podemos vernos con la suficiente frecuencia como para plantearse esta opción. Sin embargo, es vital para el buen hacer de la práctica y el cumplimiento de plazos y objetivos el transmitir cualquier avance que resulte relevante para conjunto del completo. Esto se aplica tanto a la parte de documentación como a la de codificación.

Centrándonos ahora en la parte del desarrollo software, para no lamentar pérdidas importantes o retrocesos en el avance, resulta conveniente recordar que no deberían subirse nuevas versiones del código al repositorio común si con las nuevas modificaciones se producen errores sintácticos o de ejecución de las partes del programa que funcionaban anteriormente. También se tratará en la medida de lo posible de no volver sobre parte del código que ya funcionaba, puesto que si hacemos algún cambio sobre él podemos provocar algún error y se perdería tiempo en subsanarlo.

También en este sentido, y dado que de la parte del desarrollo de software no se va a ocupar una persona en exclusiva, es importante avisar de que se va a avanzar con el código antes de hacerlo, y no únicamente cuando ya están hechos los cambios. De lo contrario, podría darse el caso de que dos personas trabajaran para conseguir un mismo requisito al mismo tiempo, malgastando recursos y tiempo con el que se podría haber avanzado en algo más.

3.2. Suposiciones, dependencias y restricciones.

Este proyecto se basa en el juego del Buscaminas clásico. Se parte de un código básico que se supone correcto. Aún así, uno de los primeros pasos antes de proceder a su modificación y ampliación es probar que verdaderamente, cumpla con los requisitos que se le piden.

Dado que la aplicación pedida debe realizarse a partir de un código ya proporcionado, todas las mejoras deben hacerse sobre este, procurando no alterar en la medida de lo posible sus métodos, salvo los que piden ser modificados explícitamente.

No se permitirá en ningún caso prescindir del código inicial para pasar a realizar el Buscaminas ampliado desde cero, o cambiar tantas cosas de este que ya en nada se parezca al original.

En cuanto a las restricciones a las que estamos sujetos, quizá la más importante y crucial para nosotros sea la del tiempo, que puede hacer que no se alcancen los objetivos pautados antes de la fecha límite de entrega.

3.3. Gestión de riesgos.

El uso de GitHub para este proyecto pretende facilitar la puesta en común de las últimas actualizaciones del código; sin embargo, su uso también conlleva unos riesgos que se detallan a continuación:

- Posibles errores a la hora de subir y compartir la última versión de la aplicación. A tal efecto, se intentará en la medida de lo posible que solo una persona esté modificando el código a la vez, ya que de lo contrario, podría darse el caso de

que dos personas subieran sus versiones al mismo tiempo, provocando pérdidas de código.

- Asimismo, será extremadamente importante hacer Commit y Push del código modificado (esto es, guardar y subir al repositorio común) si los cambios hechos en él incluyen alguna mejora o avance, antes de hacer Pull de la última versión subida a la nube. De lo contrario, este trabajo se podría perder y ser en balde.

Otros posibles riesgos que puedan surgir a lo largo del proyecto son:

- Gestión del escaso tiempo disponible para la realización de esta práctica, debido a la existencia de otras prácticas de igual o mayor dificultad en las que estamos inmersos todos los miembros del grupo, y que podrían afectar al retraso o no cumplimiento de los objetivos marcados.

- Dependencia hacia algunas herramientas. Para el desarrollo del proyecto resulta necesario utilizar software como GitHub Desktop o páginas como MyApps. El fallo o problema con alguna de ellas puede afectar negativamente a la realización de la práctica.

- Aunque en esta práctica se tratará de avanzar cada uno en solitario, son muy importantes las sesiones de puesta en común que se realizarán, presumiblemente, cada semana. Para ello se debe contar con un espacio en común, que trataremos que sea una sala de la Biblioteca del Campus. Sin embargo, dadas las fechas, no se puede asegurar la existencia de salas libres en un horario apropiado para nosotros, y debemos contemplar ese problema y buscar lugares alternativos.

- En los ordenadores con los que se realizará el proyecto coexisten dos Sistemas Operativos distintos (Windows y Mac). Este hecho provoca el riesgo de fallos

entre versiones y posibles dificultades para abrir archivos provenientes de un S.O. distinto.

3.4. Mecanismos de supervisión y control.

Aunque no esté propiamente especificado como tarea, tras completar cualquiera de los requisitos de la aplicación del Buscaminas ampliado, se realizarán las pruebas pertinentes para comprobar que funciona bien antes de seguir avanzando con el desarrollo. Estas tareas de control no se harán únicamente por la persona que se está encargando del código, sino de cualquier otro integrante del equipo, para favorecer la detección de errores, dado que es más fácil que una persona externa, que no se ha encargado de programar pero conoce los requisitos que deben cumplirse pueda comprobar si estos verdaderamente se alcanzan, frente al desarrollador, que tiene la visión más sesgada.

3.5. Plan de personal.

Este proyecto será llevado a cabo por cuatro personas, todos nosotros pertenecientes a la doble titulación de Ingeniería Informática + Administración y Dirección de Empresas. Los cuatro trabajaremos al mismo nivel de capacidad y compromiso para con la práctica.

La jerarquía que hemos elegido para el proyecto se encuentra explicada en el apartado 2.2. del presente documento.

4. Proceso Técnico

4.1. Metodología, técnicas y herramientas.

A continuación se detallan las metodologías de desarrollo, herramientas y lenguajes que se utilizarán a lo largo de la práctica:

- **Lenguajes de programación.**
  - La aplicación se realizará usando exclusivamente **Java**, un lenguaje de programación orientado a objetos.

- **Entorno de desarrollo.**
  - El IDE elegido para la codificación es **NetBeans**, un programa completamente gratuito e indicado especialmente para programar en Java.

- **Programas de apoyo al desarrollo de software.**
  - **GitHub.** Plataforma que permite el alojamiento de proyectos en la nube y la colaboración entre distintos desarrolladores. Al respecto, indicar que todos los miembros tendrán instalado **GitHub Desktop**, desde el cual podrán descargarse de manera muy sencilla la última versión del código.
  - **JavaDoc.** Esta herramienta permitirá generar de manera automática a partir de los comentarios incluidos en el código Java la documentación necesaria para comprender el código.

- **Programas de apoyo a la documentación.**
  - **Paquete de Microsoft Office.** En concreto, necesitaremos utilizar los programas Microsoft Word, Microsoft Power Point y Microsoft Project.
  - **MyApps.** Herramienta ofrecida por la Universidad Rey Juan Carlos para acceder a todo tipo de software libre o con licencia sin necesitar la

instalación de ningún programa. En este caso concreto, su uso será necesario para el uso de Microsoft Project.

- **Programas para facilitar la comunicación entre los integrantes del proyecto.**
    - Herramientas como **WhatsApp** o **Skype** serán muy útiles para discutir problemas o decisiones concretas cuando no sea posible hacerlo en persona.

Los diagramas que aquí aparecen han sido realizados utilizando, en cada caso:

- **Diagrama de Gantt.** Microsoft Project.
- **Diagrama de Pert.** Microsoft Word.
- **Calendario.** Adobe Photoshop.

4.2. Documentación software.

Junto al resto de la documentación, se adjunta el Plan de documentación del proyecto software, siguiendo el estándar IEEE 829-1983.

4.3. Funciones de apoyo al proyecto.

Para lograr que el conjunto del proyecto quede cohesionado y consistente, resulta fundamental mantener la comunicación entre los integrantes del grupo. Así, para comunicarnos informalmente utilizaremos un grupo de la herramienta de mensajería instantánea WhatsApp, además de ser esta una vía de mandarnos unos a otros borradores de los documentos. También Gmail servirá para este último propósito.

Si nos centramos en el desarrollo del código, para ello utilizaremos la herramienta Github, que nos permite tener archivos alojados en la nube, posibilitando

también el poder trabajar simultáneamente en un mismo código y sin necesidad de enviarlo por ningún sitio.

La manera en la que lograremos que la última versión del código esté disponible en los cuatro ordenadores será la siguiente:

1. Uno de los integrantes del equipo (en este caso, el encargado será Moisés García) hará *Fork* (es decir, realizará una copia) al repositorio del profesor desde su cuenta de Github. De esta manera, tendrá como suyos todos los archivos que en el repositorio se encontraran. Los demás miembros del grupo haremos Fork al repositorio de Moisés, que actuará como padre o máster.
2. Con el programa Github Desktop instalado en nuestros ordenadores, nos loguearemos y haremos un "*Clone Repository*" al repositorio del padre, es decir, de Moisés. De esta manera, nos habremos bajado el programa al ordenador, y ya estaremos en disposición de realizar cambios y compartirlos instantáneamente.
3. Dentro de Github Desktop existen las siguientes opciones:
    a. **Commit.** Esta será el primer paso que deberemos realizar cuando queramos compartir nuestros cambios. Commit guardará la nueva versión del proyecto con una fecha y un nombre.
    b. **Push.** Una vez hecho el Commit, podemos elegir esta opción, con la que conseguiremos que todos los miembros que han clonado el repositorio máster puedan acceder a esta nueva versión.
    c. **Pull.** Cada vez que algún miembro del equipo haya hecho un Push, Github Desktop avisará de que hay una nueva versión, y podemos actualizarla haciendo un Pull.
4. Además, NetBeans permite enlazar directamente un proyecto abierto en su interfaz con Github. Utilizando esta opción, cada vez que hagamos Pull, la nueva versión del código se habrá actualizado instantáneamente en nuestro entorno de desarrollo.

5. Plan de Desarrollo

5.1. Paquetes de trabajo.

Para obtener los paquetes de trabajo se ha estudiado el proyecto, identificando en él tareas en las que podría dividirse, y estas tareas a su vez se han repartido en subtareas más pequeñas y sencillas. Pasando a detallarlas:

- **A.** Planificación y realización de la documentación requerida.
- **B.** Verificaciones previas a modificaciones del código.
    - **B.1.** Verificar que el código se ejecuta correctamente.
    - **B.2.** Verificar que el código cumple los requisitos especificados.
- **C.** Mejora del código.
    - **C.1.** Añadir la opción de reiniciar el juego en cualquier momento.
    - **C.2.** Mostrar las minas que quedan por descubrir y el tiempo de la partida.
    - **C.3.** Crear niveles de dificultad.
        - **C.3.1.** Adaptar el código para nivel principiante.
        - **C.3.2.** Adaptar el código para nivel intermedio.
        - **C.3.3.** Adaptar el código para nivel experto.
        - **C.3.4.** Adaptar el código para nivel personalizado.
    - **C.4.** Guardar los 10 mejores tiempos en cada dificultad.
    - **C.5.** Incluir un campo para añadir el nombre del jugador al ganar una partida.
    - **C.6.** Permitir guardar la partida actual y poder recuperarla.
    - **C.7.** Crear menú que permita acceder a las diferentes modalidades del juego.
- **D.** Verificación posterior a modificaciones del código.
    - **D.1.** Verificar el buen funcionamiento del juego.
    - **D.2.** Verificar que el código se ajusta a los requisitos.

A modo de nota, decir que la notación utilizada para clasificar las tareas (A, B, C...) es similar a la que aparece en el diagrama de Gantt (apartado 2.1.), pero no es igual, dado que al generar el EDT de manera automática, pasaba a numerar también los hitos (de duración 0), de forma que hemos preferido reenumerar las tareas en las que realmente se destina tiempo y recursos para que su lectura y comprensión no resulten tan engorrosos.

Actividad	**Duración (expresada en días)**
A.	4
B. 1.	2
B. 2.	2
C.1.	1
C.2.	1
C.3.1.	1
C.3.2.	1
C.3.3.	1
C.3.4.	1
C.4.	1
C.5.	1
C.6.	2
C.7.	1
D.1.	2
D.2.	2

Tabla 1. Duración de las actividades

También adjuntamos el diagrama de paquetes de trabajo, tal y como aparece especificado en la documentación de la asignatura:

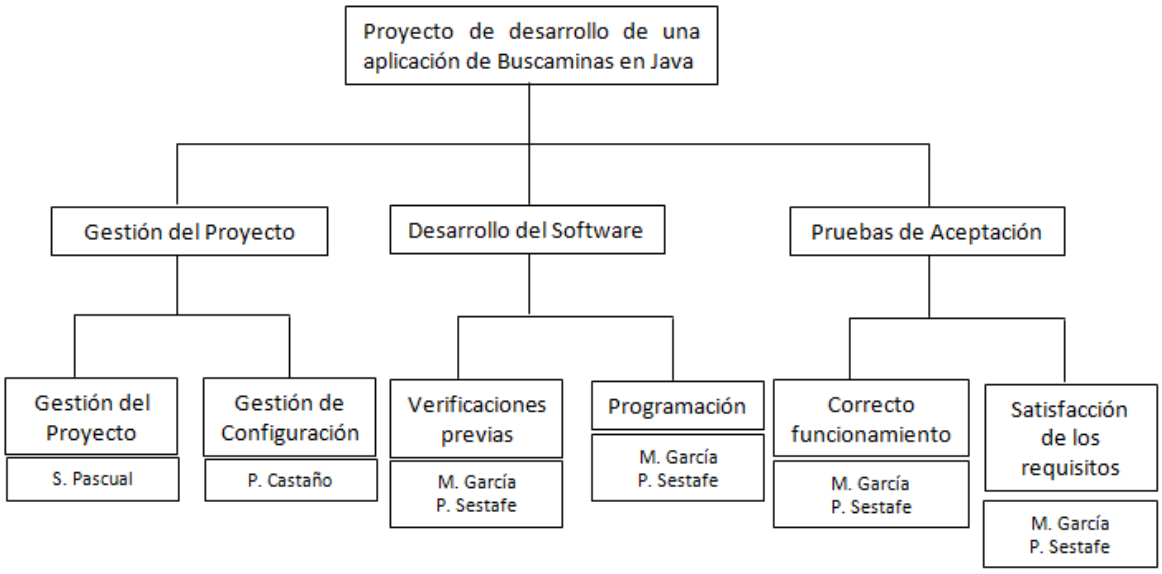

Figura 5. Diagrama de paquetes de trabajo

5.2. Dependencias.

Las dependencias existentes entre los paquetes de trabajo expuestos en el apartado anterior se reflejan en esta matriz de encadenamientos:

Actividades precedentes

Actividades siguientes	A.	B.1.	B.2.	C.1.	C.2.	C.3.1.	C.3.2.	C.3.3.	C.3.4.	C.4.	C.5.	C.6.	C.7.	D.1.	D.2.
A.															
B. 1.	X														
B. 2.		X													
C.1.			X												
C.2.			X												
C.3.1.			X												
C.3.2.						X									
C.3.3.							X								
C.3.4.								X							
C.4.				X	X				X						
C.5.				X	X				X						
C.6.										x					
C.7.				X	X				X						
D.1.											X	X	X		
D.2.														X	

Tabla 2. Matriz de encadenamientos

De manera más visual, se ha realizado el diagrama de Pert, donde además de ver las dependencias entre tareas, también nos será posible calcular el tiempo que destinaremos al total de la práctica:

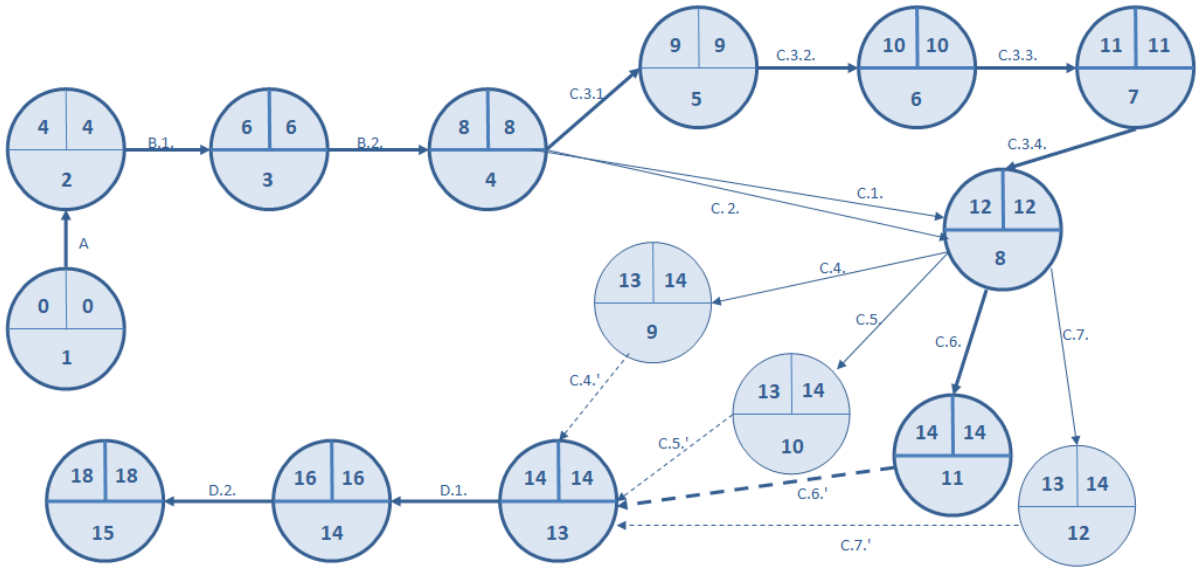

Figura 6. Diagrama de Pert.

Como se aprecia gracias a este diagrama, el tiempo total para el proyecto será de 18 días, y el camino crítico incluirá las siguientes tareas:

- **A.** Planificación y realización de la documentación requerida.
- **B.** Verificaciones previas a modificaciones del código.
    - **B.1.** Verificar que el código se ejecuta correctamente.
    - **B.2.** Verificar que el código cumple los requisitos especificados.
- **C.** Mejora del código.
    - **C.3.** Crear niveles de dificultad.
        - **C.3.1.** Adaptar el código para nivel principiante.
        - **C.3.2.** Adaptar el código para nivel intermedio.
        - **C.3.3.** Adaptar el código para nivel experto.
        - **C.3.4.** Adaptar el código para nivel personalizado.
    - **C.6.** Permitir guardar la partida actual y poder recuperarla.
- **D.** Verificación posterior a modificaciones del código.
    - **D.1.** Verificar el buen funcionamiento del juego.
    - **D.2.** Verificar que el código se ajusta a los requisitos.

5.3. Recursos.

Pasamos ahora a detallar los diferentes tipos de recursos necesarios para la realización de este proyecto, atendiendo a su naturaleza:

- **Recursos tangibles.**
  - **Activos físicos:** En este apartado destacamos la necesidad de cuatro ordenadores (uno por cada integrante) con buena conexión a Internet y, a ser posible, con una de las últimas versiones de su Sistema Operativo instalada.

    También será necesaria una sala de reuniones para favorecer el trabajo en equipo y la toma de decisiones.
  - **Activos financieros:** El proyecto supondrá coste 0.
- **Recursos intangibles.**
  - **Activos humanos:** Quizá el más importante de todos los recursos. El proyecto se realizará, como se ha venido comentando en apartados previos, por cuatro personas.
  - **Activos no humanos:**
    - **Tecnologías:** Como ya se han desarrollado ampliamente en el apartado 4.1., pasamos simplemente a enumerar que a este efecto, se utilizará NetBeans como entorno de desarrollo, Java como lenguaje de programación, y otros programas como el paquete de Office o GitHub, y remitimos a dicho apartado para más información.

5.4. Presupuesto y distribución de recursos.

El presupuesto designado al proyecto es de 0 euros, dado que no es necesario que medie ningún gasto en ningún momento de la práctica.

Los recursos de tipo humano se distribuirán según lo comentado en apartados anteriores, permaneciendo todos ellos de principio a fin del proyecto.

Sin embargo, si atendemos a las tecnologías, no todas ellas se utilizarán en todas las tareas. Concretando:

- **Proceso de documentación.** Paquete Microsoft Office.
- **Proceso de desarrollo y pruebas.** NetBeans, Java, GitHub, JavaDoc, etc.

5.5. Calendario.

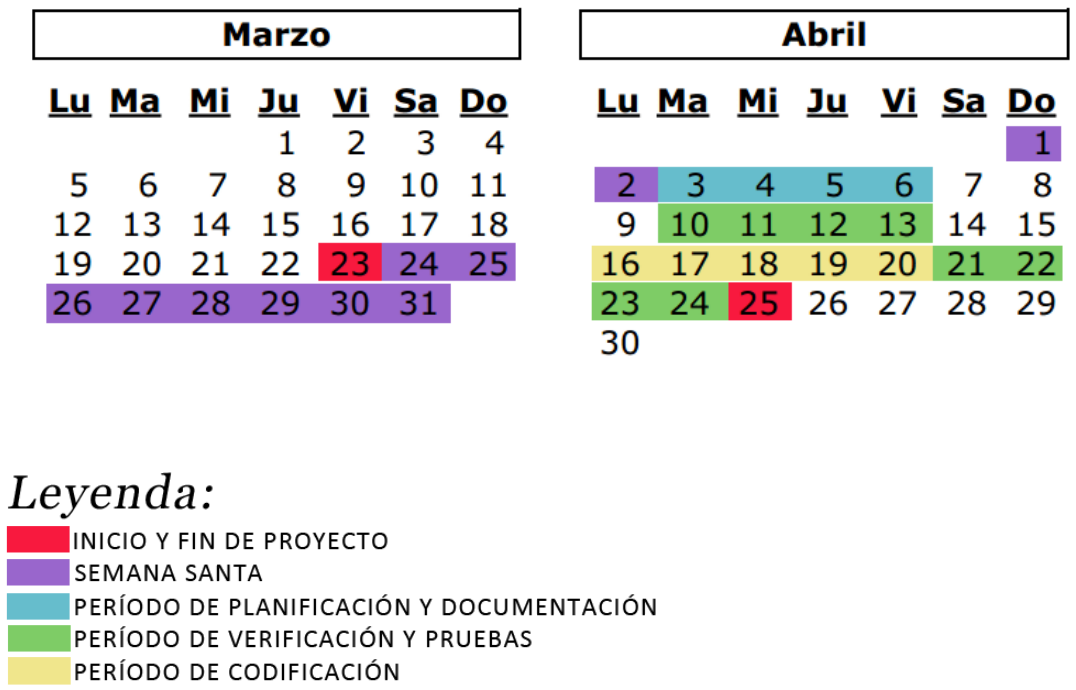

Figura 7. Calendario

Resulta necesario aclarar que el periodo de Semana Santa se consideran días en los que no se hará ningún tipo de avance por ninguno de los integrantes del grupo, siendo el día 3 de abril cuando comienza realmente el proyecto.

Como se aprecia, el proyecto debe empezar necesariamente con la planificación y consiguiente documentación que se pide (PGPS y PGCS). Una vez

pautados y explicitados los objetivos que hay que desarrollar en dichos documentos, es posible empezar a probar el código inicial, y hecho esto, empezar con el periodo de desarrollo propiamente dicho, que culminará con las últimas verificaciones para comprobar que todo funciona correctamente y se adecúa a los requisitos.

A modo de última nota, decir que se ha tratado de dejar libres los fines de semana; no obstante, la semana previa a la entrega no se puede asegurar que estos días no estén volcados a la práctica, aunque se tratará de que la carga de trabajo para esos días sea más reducida.