

Instituto Tecnológico de Costa Rica.

Escuela de computación.

II semestre 2023.

Programación Orientada a Objetos GR #1

Profesor: William Mata Rodríguez

Estudiante: José Pablo Vega Solano 2023367318, Kenneth Vargas Chacón -2023148829

Programa 2 - Ken Ken

Contenido de la documentación

[Enunciado del proyecto 3](#_Toc150005180)

[Temas investigados 13](#_Toc150005181)

[Software de manejo de versiones y trabajo colaborativo 13](#_Toc150005182)

[Aprendizajes: 13](#_Toc150005183)

[Aplicación: 13](#_Toc150005184)

[Bibliografía: 13](#_Toc150005185)

[Manejo de archivos en formato XML 14](#_Toc150005186)

[Aprendizajes: 14](#_Toc150005187)

[Aplicación: 14](#_Toc150005188)

[Bibliografía: 14](#_Toc150005189)

[Threads o hilos 15](#_Toc150005190)

[Aprendizajes: 15](#_Toc150005191)

[Aplicación: 15](#_Toc150005192)

[Bibliografía: 15](#_Toc150005193)

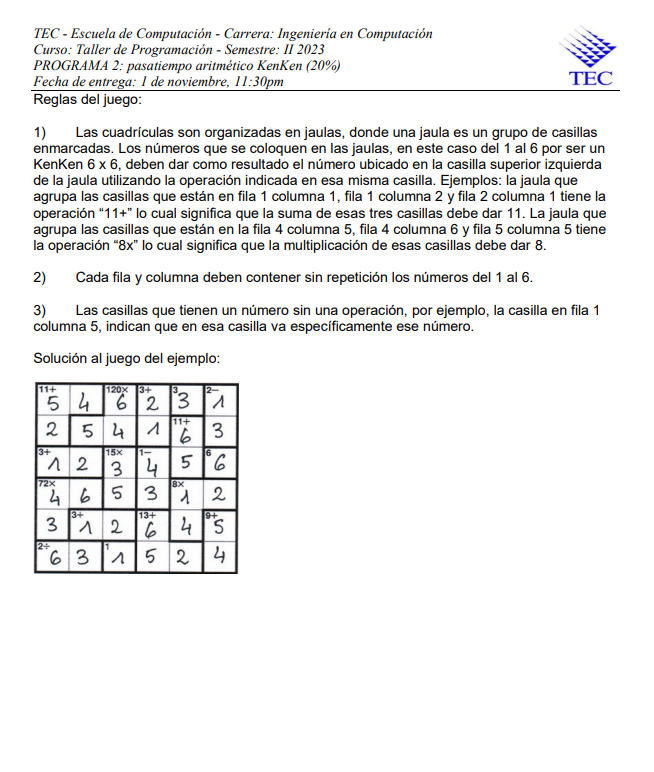
[Solución 16](#_Toc150005194)

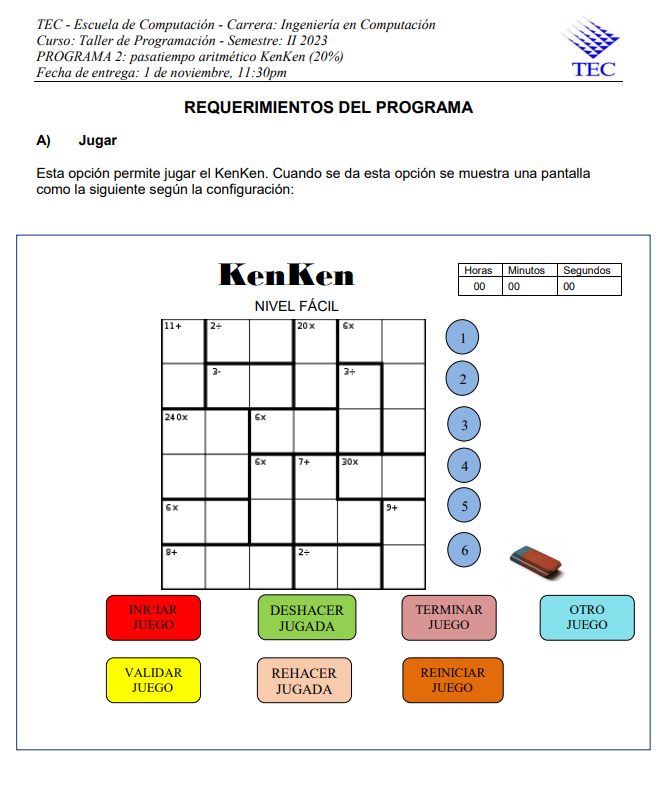
[Conclusiones del trabajo 17](#_Toc150005195)

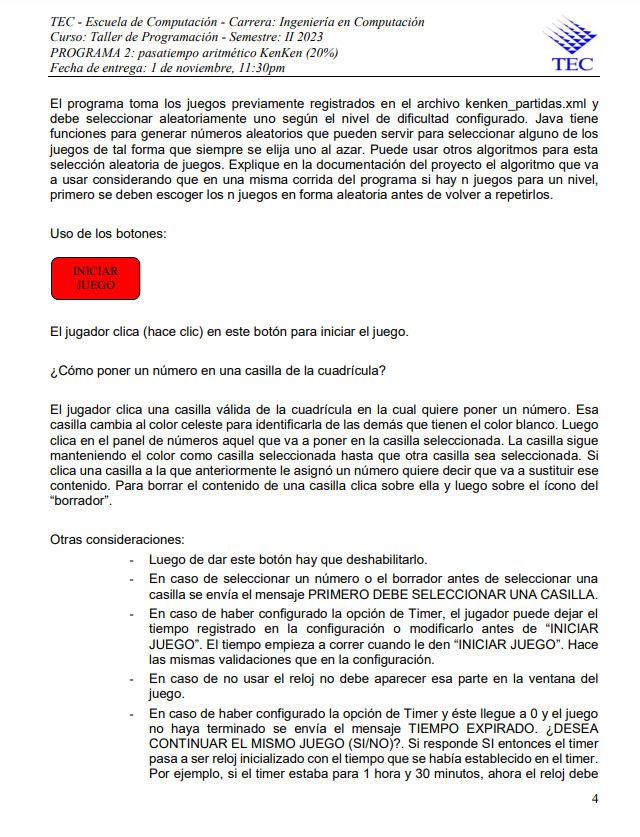
[Lista de revisión del proyecto 18](#_Toc150005196)

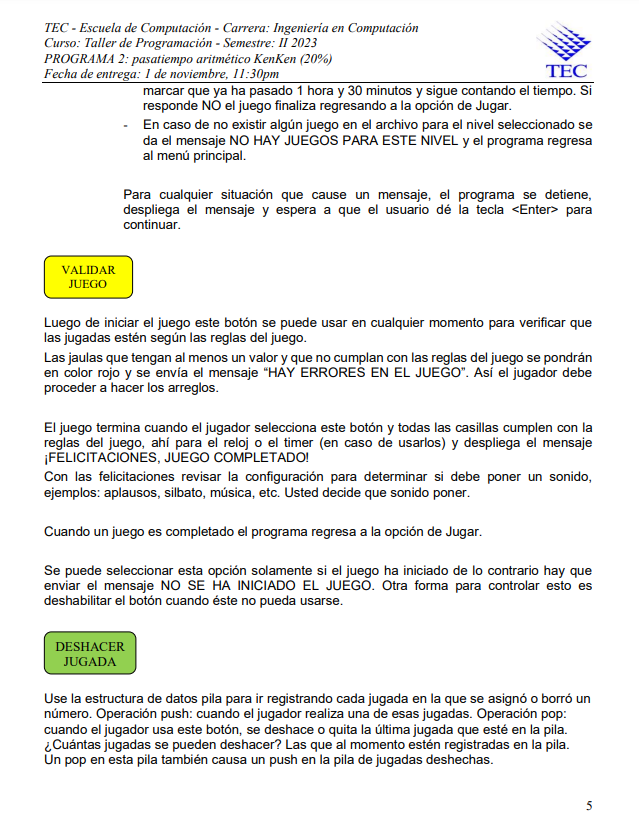
[Manual de usuario 19](#_Toc150005197)

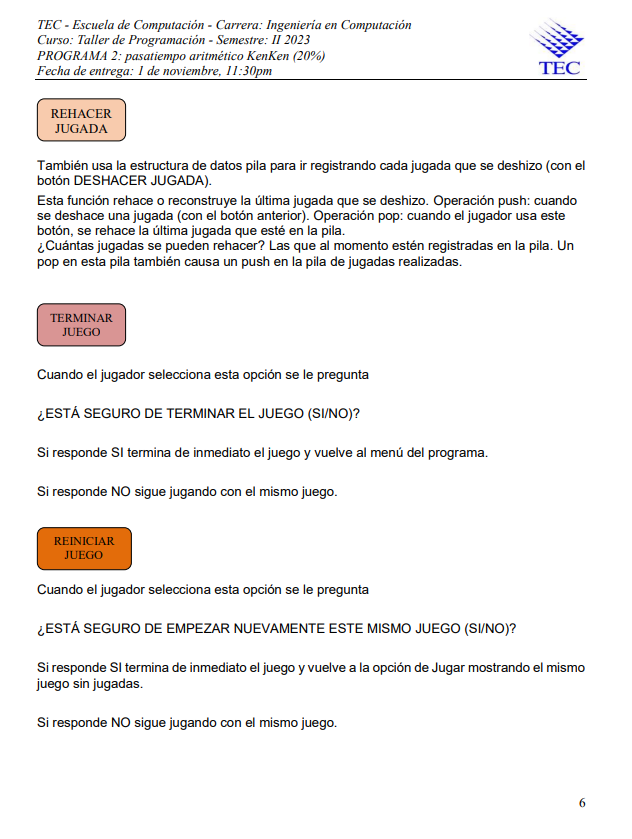
# Enunciado del proyecto

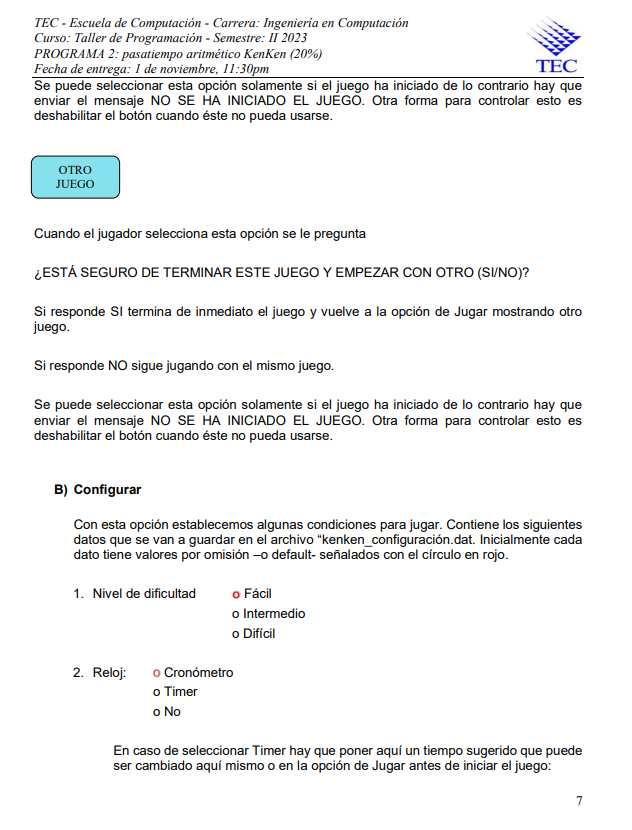


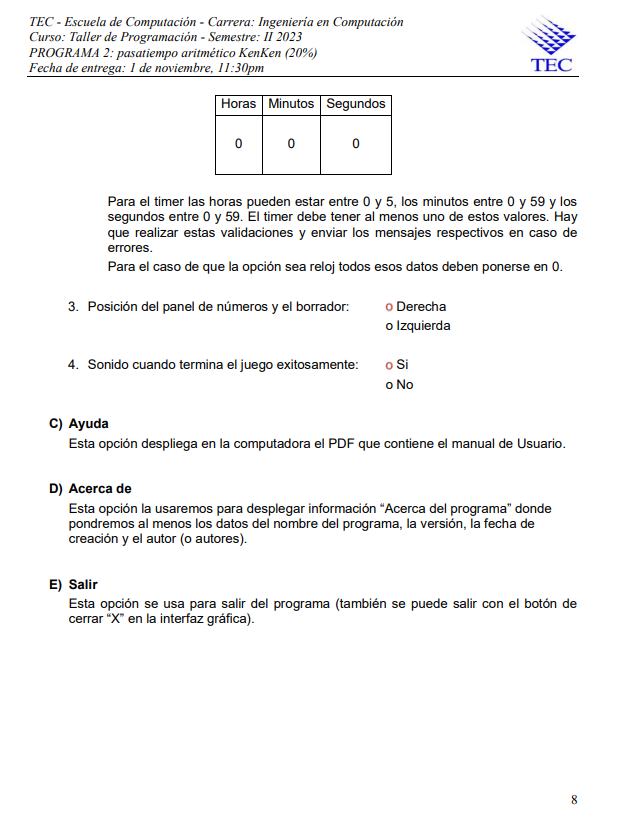


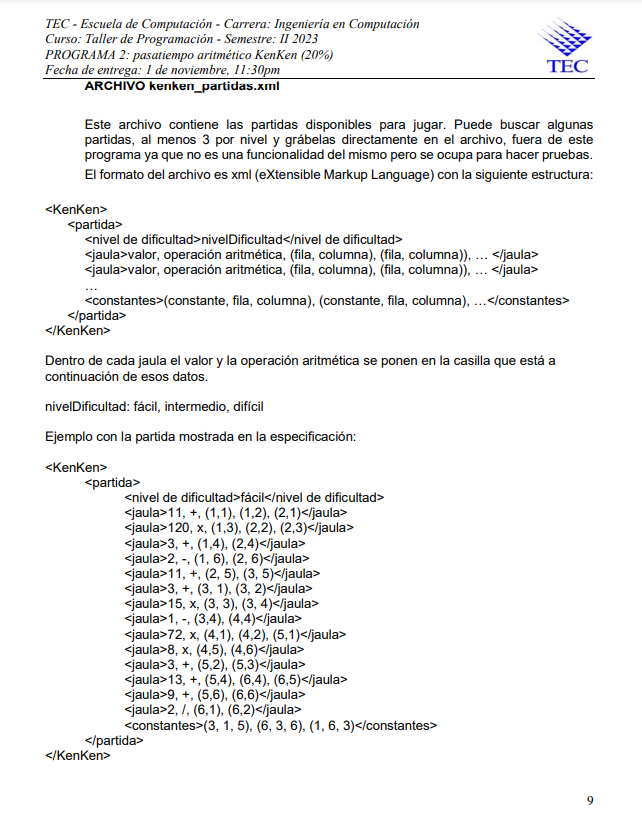


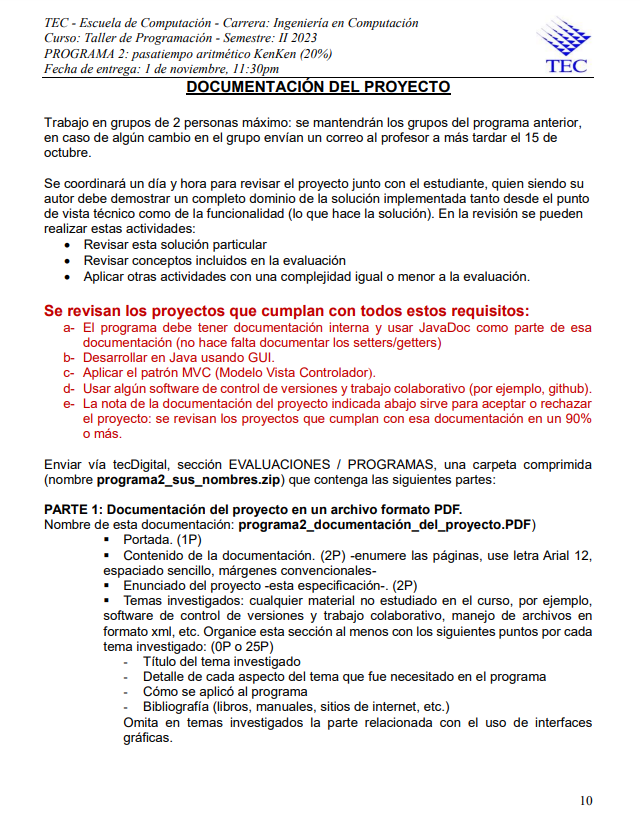












# Temas investigados

## Software de manejo de versiones y trabajo colaborativo

### Aprendizajes:

Para esto tuvimos que comprender cómo funciona bit con NetBeans, puesto a que ya conocíamos para qué funciona gis inicialmente.

Aprendimos cómo inicializar git, inicializar un repositorio, cómo subir datos, volver a ellos y cómo notificar mediante el menú commit las actualizaciones hechas en cada versión.

También comprendimos cómo visualizar los cambios entre una versión y otra y cómo navegar entre estas.

En lo que es el trabajo colaborativo, pensamos en utilizar IntelliJIDEA, pero no es tan cómodo como NetBeans a la hora de hacer una interfaz gráfica. Por esto nuestro trabajo colaborativo se basó un manejo de versiones en GitHub.

Github permite crear un repositorio en la nube, por lo que podemos manejar las versiones en ambos dispositivos.

### Aplicación:

Realmente solamente aplicamos lo que aprendimos para manejar eficientemente nuestro trabajo.

### Bibliografía:

### <https://hashblogeando.wordpress.com/2019/01/10/usando-git-en-netbeans/>

## Manejo de archivos en formato XML

### Aprendizajes:

XML es un lenguaje de marcado que se basa en etiquetas. Este lenguaje nos permite almacenar datos y dividirlos o clasificarlos mediante estas etiquetas. Mientras más grande sea un archivo XML, más información contiene. En este lenguaje se crean etiquetas con el nombre que se desee y se pueden anidar estas etiquetas, almacenando los datos de una etiqueta como un atributo de otra etiqueta.

También aprendimos a leer estos archivos y dependiendo de su etiqueta, asignar un valor dentro del programa.

### Aplicación:

Hicimos un archivo XML con el formato sugerido para encapsular cada partida, su dificultad y sus jaulas y celdas.

También hicimos una función para poder leer y reconocer estas etiquetas.

### Bibliografía:

<https://aws.amazon.com/es/what-is/xml/#:~:text=El%20lenguaje%20de%20marcado%20extensible,datos%20y%20aplicaciones%20de%20terceros>.

<https://youtu.be/AZihBEg8VBk?si=PgDRrs2Kxd1IxhD6>

<https://youtu.be/p59RiQoaRWg?si=0EGt9pav147AcwQC>

<https://www.delftstack.com/es/howto/java/java-read-xml/>

## Threads o hilos

### Aprendizajes:

Un thread es algo que permite la ejecución paralela, permitiendo que varias funciones se ejecuten simultáneamente sin que el programa colapse.

El método run permite que el thread se ejecute y funcione. Un thread debe tener un atributo booleano para controlar el bucle del run.

El método sleep permite liberar los recursos y esperar un tiempo específico entre una función y otra.

Un thread puede operar de forma independiente, pero es más eficiente cuando está asociado a otra clase o a una ventana.

Es necesario hacer un "extend" de thread al declarar la clase para que esta sea un thread.

### Aplicación:

La aplicación de threads en nuestro programa fue únicamente para controlar el cómo avanza el tiempo en el cronómetro y en el temporizador de la partida.

Implementamos un thread para un Label de cada tipo. Realmente ambos threads y Labels son muy similares, solamente que fue necesario hacerlos por separado porque tienen una funcionalidad distinta.

En los threads se implementaron relativamente los mismos métodos, pero en los labels se necesitó crear un constructor más específico para el Label del temporizador, pues es necesario inicializarlo con valores asignados en ejecución.

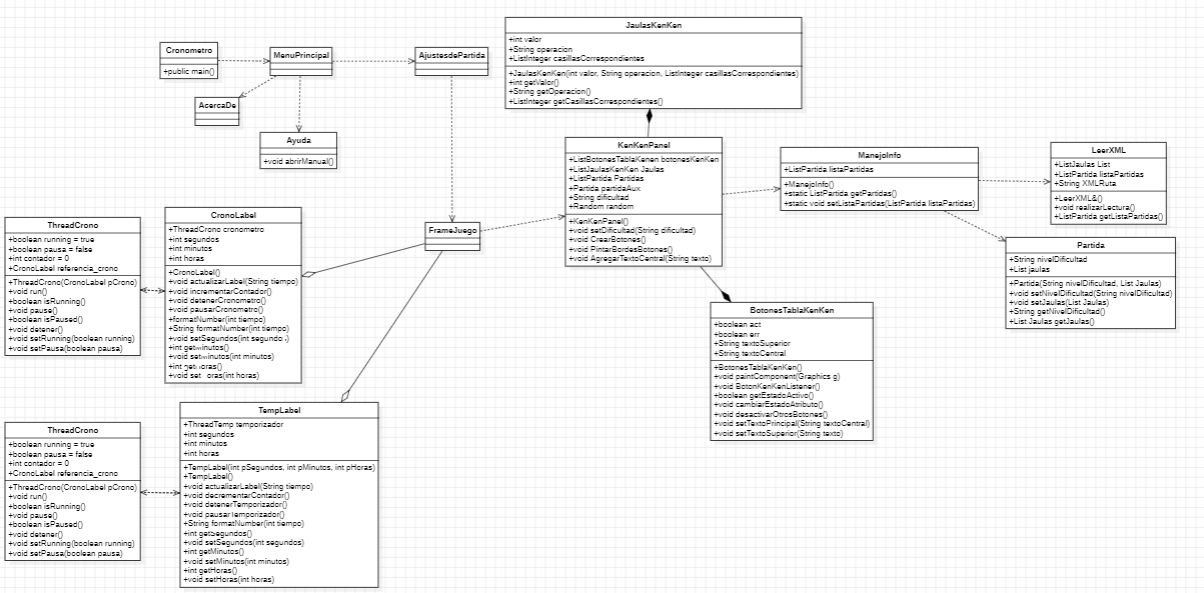
El resto del funcionamiento dentro de estas estructuras es bastante simple, pues solamente controla los atributos dentro de estas clases mediante setters y getters.

### Bibliografía:

<https://ed.team/blog/como-funcionan-los-hilos-en-programacion>

Además, recibimos ayuda de unos compañeros de carrera que están con otro profesor, el cuál les enseñó bastante de este tema. Realmente fue uno de ellos el que nos propuso hacer estos contadores con threads, para poder manejarlos más fácilmente.

# Solución



# Conclusiones del trabajo

En este trabajo aprendimos muchas cosas. La más importante de ellas fue el cómo manejas distintos objetos entre distintas clases, viendo necesarias los distintos tipos de relaciones.

Desde el inicio decidimos no trabajar con una matriz, pues preferimos mantener la facilidad a la hora de desplegar los botones antes que la facilidad a la hora de manejar los datos.

Nuestro programa tiene una estructura bastante interesante, pues aprendimos también de manejar los objetos entre clases de manera efectiva, aprendimos cómo manejar los datos desde un sistema, poder ver los datos desde otro y finalmente controlarlo todo desde otro, aplicando así el modelo MVC.

Lamentablemente dejamos gran parte del juego de manera incompleta por factores externos, pero logramos entender mucho de lo que necesitábamos para lograr el objetivo del proyecto. Es un proyecto muy interesante. Esperamos realizar el tercer proyecto basado en este de manera completa.

# Lista de revisión del proyecto

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Concepto | Puntos | % de avance  100/%/0 | Puntos obtenidos | Análisis de resultados |
| Opción Jugar:  Desplegar cuadriculado  Desplegar información de cada casilla según partida  Enmarcado de jaulas según partida  Desplegar el resto de la opción  Algoritmo de selección de partida | 5  5  8  5  1 | 5  5  8  5  1 | 24 | Completo |
| Iniciar Juego:  Actualizar casillas (colocar y  borrar números)  Controles del proceso | 5  5 | 5  5 | 10 | Completo |
| Reloj en tiempo real usado en el control del juego | 10 | 10 | 10 | Completo |
| Validar Juego | 10 | 5 | 5 | Se cuenta parcialmente con la función.  No quedó completa por lo no es utilizable en el juego |
| Sonido cuando el jugador gana | 5 | 2 | 2 | Tenemos la clase |
| Deshacer jugada | 5 | 0 | 0 | Nada |
| Rehacer jugada | 5 | 0 | 0 | Nada |
| Otro Juego | 3 | 3 | 3 | Completo |
| Reiniciar Juego | 2 | 2 | 2 | Completo |
| Terminar Juego | 1 | 1 | 1 | Completo |
| Opción configurar | 6 | 6 | 6 | Completo |
| Aplicación patrón MVC | 10 | 7 | 7 | Según lo que comprendemos, aplicamos parcialmente este modelo, separando las funcionalidades. |
| Ayuda (despliegue del PDF del manual de usuario) | 1 | 1 | 1 | Completo |
| Documentación del proyecto: Manual de usuario  Resto de la documentación | 4  4 | 4  4 | 8 | Completo |
| TOTAL | 100 |  | 79 |  |
| Funcionalidades desarrolladas adicionalmente |  |  |  |  |

# Manual de usuario

Archivo adjunto