Universidad Internacional de La Rioja

Escuela Superior de Ingeniería y Tecnología

Grado en Ingeniería Informática

Desarrollo de aplicación para distribuir costaleros automáticamente usando algoritmos de optimización matemática.

Trabajo fin de estudio presentado por:	Pedro José Luque Ordóñez
Director/a:	Pablo Aragón Moreno
Fecha:	08/03/2025
Repositorio del código fuente:	https://github.com/pedrojluke/tfg-unir

A mis padres,

A Clara, Rodrigo y Beatriz.

"Caminante, no hay camino,

Se hace camino al andar.

Al andar se hace camino,

Y al volver la vista atrás

Se ve la senda que nunca

Se ha de volver a pisar."

(Antonio Machado, Proverbios y Cantares)

Resumen

El objetivo de este Trabajo Fin de Grado es dar solución eficiente al problema de la asignación óptima de costaleros en los pasos de Semana Santa, mediante optimización matemática. Este proceso ha sido desarrollado tradicionalmente por el equipo de capataces, en base a la experiencia previa, pero sin ningún tipo de criterio matemático definido, haciendo de esta tarea una empresa ineficiente, ardua y sobre todo, costosa en tiempo.

Para ello se proyecta, mediante programación lineal y optimización matemática, minimizar la diferencia de altura entre los costaleros y sus trabajaderas, persiguiendo que todos estén lo más cerca posible de la trabajadera, minimizando el uso de suplementos para compensar dicha falta de altura.

Se han examinado diferentes algoritmos cómo los voraces, back tracking con y sin poda, programación dinámica y el algoritmo de Kuhn – Munkres, eligiendo este último cómo la mejor solución para este problema.

El resultado de este trabajo es un hito en la organización de cuadrillas de costaleros, ya que se reduce el tiempo necesario para la distribución de estos, repercutiendo en una mejor Inter movilidad dentro del paso. Además, facilita la gestión de posibles imprevistos, ya que ante un posible imprevisto, el tiempo de reacción es limitado, y tener una herramienta que automáticamente te dé la mejor solución ayuda.

La solución tecnológica propuesta es una aplicación multiplataforma en React-Native, conectado con una solución en la nube para el manejo de los datos cómo es Firebase, permitiendo al capataz administrar ensayos, pasos o costaleros de forma eficiente.

Palabras clave: Costalero, Semana Santa, Capataz, programación lineal, Optimización, Algoritmia.

Abstract

The purpose of this project is to use mathematical optimisation to allocate Easter processions float-bearers efficiently.

Traditionally, this process has been carried out manually by the overseer and his team based on their experience and without a mathematical approach. As a result, this task is often inefficient, arduous, and most importantly, time-consuming.

To address this issue, a linear programming and mathematical optimization model-based has been developed to reduce the height difference between float-bearers and their corresponding support beams. The main goal is to ensure that all float-bearers are as close as possible to each support beam, reducing the need for height compensation with additional supports.

Several algorithmic approaches were researched, including greedy algorithms, backtracking with and without pruning, and dynamic programming. Ultimately, the Kuhn – Munkres algorithm, which was selected, due to is the most suitable solution for this problem, given its efficiency in finding an optimal allocation.

The implementation of this system represents a milestone in the management of float-bearer teams. It significantly reduces the time required for their allocation and improves internal mobility within the Passion floats. Moreover, it facilitates decision – making in unforeseen situations, where a fast reassignment of bearers is required.

The proposed technological solution consists of a cross-platform application developed in React-Native, integrated with a cloud-based backend using Firebase. This system enables the overseer to efficiently manage bearers, Passion floats and procession trials efficiently.

Keywords: Float-Bearers, Holy Week, Overseer, Linear Programming, Optimization, Algorithmics.

Índice de contenidos

1. Introducción 1

1.1. Motivación 2

1.2. Planteamiento del trabajo 3

1.3. Estructura del trabajo 5

2. Contexto y Estado del Arte 7

2.1. Análisis del contexto 7

2.1.1. Procesión de Semana Santa 9

2.1.2. Cuadrilla de Costaleros 11

2.1.3. Métodos de distribución de costaleros 15

2.1.4. El rompecabezas del capataz 16

2.2. Estado del arte 20

2.2.1. CostalGo 21

2.2.2. GesHdad 22

2.2.3. GHerCof 23

2.2.4. HermandApp 24

2.2.5. Comparativa entre las diferentes aplicaciones 26

2.3. Conclusión 27

3. Análisis del problema y justificación del algoritmo elegido 28

3.1. Resolución manual del problema 28

3.2. Modelado matemático 32

3.3. Programación lineal 34

3.4. Algoritmos que dan solución al problema 36

3.4.1. Back tracking o vuelta atrás 36

3.4.2. Algoritmos voraces (Greedy) 37
3.4.3. Programación dinámica 37
3.4.4. Algoritmo Kuhn - Munkres 38
3.5. Decisión final 39
4. Objetivos y metodología de trabajo 41
4.1. Objetivo general 41
4.2. Objetivos específicos 42
4.3. Metodología de trabajo 43
4.3.1. Organización del proyecto 44
4.3.2. Fase de Discovery 48
4.3.3. Fase de especificación de requisitos 48
4.3.4. Fase de desarrollo 49
4.4. Herramientas usadas 49
4.4.1. Miro 49
4.4.2. Visual Studio Code 51
4.4.3. Zotero 51
4.4.4. MeuScrum 52
4.4.5. GitHub 52
5. Diseño y especificación de requisitos 53
5.1. Resultado de la fase de Discovery 53
5.2. Definición de actores 54
5.3. Creación de historias de usuario 55
5.3.1. Gestión de pasos 55
5.3.2. Gestión costaleros 55
5.3.3. Planificación de ensayos 55

5.3.4. Asignación automática de costaleros 56
5.3.5. Informes 56
5.3.6. Registro capataces 56
5.3.7. Gestión capataces ayudantes o auxiliares 57
5.4. Especificación de requisitos 57
5.4.1. Requisitos funcionales 57
5.4.3. Requisitos no funcionales 59
5.5. Diagrama de caso de uso 60
5.6. Planificación y priorización por sprints 61
5.6.1. Sprints 1,2 y 3 61
5.6.2. Sprints 4 y 5 61
5.6.3. Sprints 6,7 y 8 61
5.6.4. Sprints 9 y 10 62
5.6.5. Sprints 11, 12 y 13 62
5.6.6. Sprints 14 y 15 62
5.6.7. Sprints 16, 17 y 18 62
5.7. Prototipado de componentes 62
5.7.1. Menú Principal 63
5.7.2. Añadir Paso 63
5.7.3. Detalle de Paso 63
5.7.4. Añadir costalero 64
5.7.5. Búsqueda de costalero 64
5.7.6. Menú de ensayos 65
5.7.7. Creación de ensayo 65
5.7.8. Asignación de costaleros 66

6. Desarrollo e implementación 68
6.1. Tecnologías disponibles 68
6.1.1. Flutter 68
6.1.2. Apache Cordova 69
6.1.3. React Native 69
6.1.4. Ionic 69
6.2. Tecnologías elegidas y justificación 69
6.2.1. React Native 70
6.2.2. Expo 72
6.2.3. React Native Paper 72
6.2.4. Node.js 73
6.2.5. Firebase 73
6.3. Arquitectura de la aplicación 75
6.3.1. Diseño arquitectónico 76
6.3.2. Modelo de datos 77
6.4. Implementación del algoritmo de distribución 81
7. Conclusiones y trabajo futuro 82
7.1. Conclusiones del trabajo 82
7.2. Líneas de trabajo futuro 84
7.2.1. Líneas de trabajo futuro 84
7.2.2. Mejoras 84
7.3. Valoración personal 85
Referencias bibliográficas 87
Anexo A. Requisitos funcionales del sistema 92
Anexo B. Cuadrante Costaleros Semana Santa 2024 101

Índice de acrónimos 1

Índice de figuras

Figura 1. Esquema de una procesión en la calle. Fuente: Blog Güichi de Carlos. 10

Figura 2. Paso malagueño. Fuente: De museos por Málaga 12

Figura 3. Paso estilo sevillano. Fuente: Gente de Paz 12

Figura 4. Interior de un paso de Semana Santa. Fuente: CiberEcija.com 13

Figura 5. Costalero siendo tallado. Jorge Molina (Cedida) 14

Figura 6. Distribución de costaleros año 2025. Fuente: creación propia 17

Figura 7. Logo de CostalGo. Fuente: www.costalgo.com 21

Figura 8. CostalGo no soporta asignación automática de costaleros 22

Figura 9. Funcionamiento de GesHdad. Fuente: www.geshdad.com 23

Figura 10. Módulo directorio de GherCof. Fuente www.ghercof.com 24

Figura 11. Panel de control de HermandApp. Fuente: hermandapp.io 25

Figura 12. Grafo de un problema de transporte. Fuente: Manuel García Narváez - Universidad de Zaragoza 35

Figura 13. Esquema general de un algoritmo de back tracking. Fuente: Blog de Alberto Pascual 36

Figura 14. Iteraciones del algoritmo húngaro para un reparto de tareas. Fuente: Wikimedia Commons 39

Figura 15. Comparación agile vs waterfall. 43

Figura 16. Product backlog. Scrum.org 45

Figura 17. App de miro para este proyecto 50

Figura 18. Visual Studio Code 51

Figura 19. Zotero 51

Figura 20. Proyecto en MeuScrum 52

Figura 21. Página principal del proyecto en GitHub 52

Figura 22. Resultado del User Story Mapping. 54

Figura 23. Diagrama de caso de uso 60

Figura 24. Menú Principal de la aplicación 63

Figura 25. Prototipo de menú de paso, creación de paso y de creación de costalero 64

Figura 26. Prototipo de búsqueda de costaleros y de creación de ensayos 65

Figura 27. Asignación de costaleros 67

Figura 28. Logo de React Native 70

Figura 29. Estructura de una pantalla en React-Native. Fuente: https://reactnative.dev 71

Figura 30. Logo de Expo 72

Figura 31. Pantalla principal de Firebase 74

Figura 32. API keys de firebase en Visual Studio Code, ocultadas por seguridad 75

Figura 33. Consulta de datos a Firebase 75

Figura 34. Diseño arquitectónico de la aplicación. Fuente: creación propia 77

Figura 35. Resultado de distribución de costaleros automática 81

Índice de tablas

Tabla 1. Fechas de la Semana Santa en los próximos años. Fuente: calendario-365.es.....8

Tabla 2. Esquema de roles en un paso. 15

Tabla 3. Relación de costaleros del año 2024. Cedida por el palio de la Virgen de la Caridad de Cabra (Córdoba).....20

Tabla 4. Comparativa de aplicaciones de gestión de hermandades. 26

Tabla 5. Ejemplo de asignación de costaleros de forma simple 30

Tabla 6. Coste de asignación de cada costalero a una trabajadera 31

Tabla 7. Planificación del proyecto por sprints. 46

Tabla 8. Ejemplo de creación de Historia de usuario 47

Tabla 9. Definición de actores 55

Tabla 10. Listado de Historias de Usuario relativas a la gestión de pasos 55

Tabla 11. Listado de Historias de Usuario relativas a gestión de costaleros.....55

Tabla 12. Listado de Historias de Usuario relativas a gestión de ensayos 56

Tabla 13. Listado de Historias de Usuario relativas a la asignación de costaleros.....56

Tabla 14. Listado de Historias de Usuario de reporting. 56

Tabla 15. Listado de Historias de usuario referentes al registro en la aplicación. 57

Tabla 16. Listado de historias de usuario para la gestión de capataces auxiliares. 57

Tabla 17. Resumen de requisitos funcionales 58

Tabla 18. Requisitos no funcionales del sistema. 59

Tabla 19. Colección Usuarios de Firebase 78

Tabla 20. Colección pasos de Firebase 79

Tabla 21. Colección Ensayos de Firebase 79

Tabla 22. Colección Trabajaderas de Firebase 80

Tabla 23. Colección asignaciones de Firebase....81

Tabla 24. RF01 Creación de pasos. 92

Tabla 25. RF02 Añadir trabajaderas y huecos al paso. 92

Tabla 26. RF03 Editar - borrar paso 93

Tabla 27. RF04 Añadir costaleros 93

Tabla 28. RF05 Editar - borrar costalero....94

Tabla 29. RF06 Añadir foto a ficha de costalero....94

Tabla 30. RF07 Crear ensayo 94

Tabla 31. RF08 Ver histórico de ensayos....95

Tabla 32. RF09 Gestionar ensayos pasados 95

Tabla 33. RF10 Copiar ensayo....95

Tabla 34. RF11 Asignación automática de costaleros 96

Tabla 35. RF12 Asignar costaleros de forma manual 96

Tabla 36. RF13 Generación de plantilla de costaleros 97

Tabla 37. Generar informe de asistencia....97

Tabla 38. RF15 Exportar informes a PDF 98

Tabla 39. RF16 Notificar no asistencia vía WhatsApp 98

Tabla 40. RF17 Registro capataz en el sistema usando Google SSO 98

Tabla 41. RF18 Registro capataz en el sistema usando usuario y contraseña 99

Tabla 42. Registro de usuario usando otros SSO cómo Facebook o Apple....99

Tabla 43. RF20 Enviar invitación a capataces auxiliares....99

Tabla 44. RF21 Gestionar capataces auxiliares 100

Tabla 45. Cuadrante procesión Semana Santa 2024....101

1. Introducción

La Semana Santa es una celebración religiosa, profundamente arraigada en nuestra cultura, especialmente en el sur de España, Extremadura, Levante y las dos Castillas. Su relevancia, valor y aceptación popular ha sido reconocida oficialmente por el Ministerio de Turismo declarando la mayoría de Las semanas Santas de estas regiones cómo de interés turístico internacional. (*Ministerio de Industria y Turismo - Fiestas de Interés Turístico Internacional*, s. f.).

Uno de los elementos fundamentales en la Semana Santa son las llamadas procesiones, desfiles de Imágenes Sagradas donde se celebra la Pasión y Muerte de Cristo. Estas imágenes pueden ser portadas por costaleros o por portadores, según el estilo del paso procesional. En algunos lugares de la geografía española el paso descansa sobre los hombros mientras que en otros lugares descansa sobre la cerviz. Esto es lo que hace diferenciar entre costaleros y portadores u hombres de trono, que portan la Sagrada Imagen sobre su hombro.

Cuando se habla de costaleros, hay que mencionar que el paso por dentro se organiza en trabajaderas, unos travesaños horizontales situados en la parte inferior del paso. Fabricadas en madera, tienen la finalidad de servir como apoyo a los costaleros para sustentar el paso. Cada cargador contacta su región cervical inferior con la trabajadera, interponiendo entre ambos una protección que se llama costal, que amortigua la presión y distribuye uniformemente el peso por el cuello. (Colectivo Cabra en el Recuerdo, 2014)

Aquí se introduce la figura del capataz, que es la persona encargada de dirigir el paso en las procesiones, guiando a los costaleros desde el frontal del paso. A su vez, el capataz asume también el cometido de igualar a los costaleros según su altura. (Hortelano, Rosa, 2024).

La estatura de los costaleros es un factor muy importante ya que debe ser uniforme para evitar que el paso camine de forma inestable y sobre todo para preservar la salud de los cargadores. El objetivo trata de que todos los costaleros estén a la misma altura desde el cuello hasta la altura de la trabajadera. Lógicamente, si la trabajadera está a una altura determinada, y no todos los costaleros llegan a esa altura, hay que suplementar dicha diferencia con unos tacos de madera para hacer que todos estén a la misma altura y por tanto la carga se reparta de manera homogénea. Esta tarea no es fácil y siempre se ha hecho de forma manual siendo un

proceso bastante complejo y que requiere de bastante tiempo para llegar a una solución donde la mayoría de los costaleros vayan lo más cerca posible de la trabajadera. Esta ordenación, en la época actual donde la Semana Santa vive una edad de oro con multitud de aspirantes a costaleros, se vuelve ingobernable si para colocar 35 costaleros tienes 50 de diferentes alturas. (Hernández, 2010)

Es ahí donde se ha de poner encima de la mesa automatizar dicha tarea, garantizando que la solución obtenida sea la óptima.

1.1. Motivación

Este trabajo fin de grado (TFG) nace para intentar dar respuesta a una casuística que se repite en el ámbito de la Semana Santa, específicamente en el de las cuadrillas de costaleros: lo complejo que resulta organizar y distribuir a los costaleros debajo de un paso. Esta operación siempre se ha realizado de forma manual, probando diferentes combinaciones hasta encontrar una que se considere factible o que guste al capataz de la cuadrilla. Dicho procedimiento es realmente costoso en tiempo e implica bastantes quebraderos de cabeza para dar con un resultado satisfactorio. Además nadie asegura que el resultado obtenido sea el óptimo si no se puede corroborar con ninguna herramienta.

La elección de este tema se fundamenta en la experiencia acumulada en el ámbito del mundo del costal, permitiendo conocer de primera mano cómo de complejo puede ser la asignación de los costaleros en un paso de Semana Santa. Esta tarea resulta harto complicada debido a la disparidad de alturas de los costaleros, lo que exige una meticulosa adaptación a las alturas de las trabajaderas para lograr una distribución adecuada.

Este proceso, aplicando un algoritmo de asignación óptima, puede llevar a encontrar la solución del problema, siendo esta aquella cuya suma de las diferencias de cada costalero con respecto a su trabajadera sea la mínima de todas las combinaciones. Esta implementación además repercutirá en un mejor confort de los costaleros al ir colocados de la mejor forma posible.

Analizando el estado del arte, si bien se han encontrado algunas herramientas para gestión interna de cofradías, o para organizar fechas de ensayos u otro tipo de actos, no se ha encontrado ninguna para la colocación óptima de costaleros dentro de un paso de Semana

Santa, considerando que este trabajo puede tener un gran impacto en la comunidad, ya que, por un lado agilizará y ahorrará bastante tiempo a la hora de organizar cuadrillas de costaleros, y por otro lado, además, se reducirán al mínimo los suplementos de altura a colocar dentro del paso para igualar la altura de aquellos costaleros que no lleguen a la altura fijada por la trabajadera. Esto, asimismo, conllevará un ahorro a la cofradía y sobre todo facilitará la permuta de los costaleros dentro del paso.

Vivimos en una época en la que la tendencia es automatizar procesos, y la Semana Santa y sus procesos subyacentes no iban a ser menos, adaptándose a los nuevos tiempos con la implementación de un algoritmo de optimización que minimice el uso de tacos para mitigar la falta de altura con respecto a la trabajadera.

Por último, se considera una gran motivación para llevar a cabo esta empresa la oportunidad de aportar un legado de enjundia a la Semana Santa, contribuyendo a la optimización de procesos en un mundo tan inmovilista y tradicional, poniendo en valor cómo la tecnología puede mejorar la gestión de una cuadrilla de costaleros, además de democratizar la gestión de los costaleros, siendo esta herramienta accesible a todo el mundo, sin necesidad de conocimientos exhaustivos del mundo cofrade para para poder generar un reparto óptimo de costaleros.

1.2. Planteamiento del trabajo

El mundo cofrade es un mundo relativamente conservador y poco dado a innovaciones técnicas y tecnológicas. El presente trabajo se acota al contexto de las cuadrillas de costaleros y a la distribución de los costaleros de forma uniforme dentro de un paso procesional. Actualmente este proceso de distribución sigue los siguientes pasos:

- Celebración de la *igualá,* donde el capataz mide la altura desde el suelo hasta la cerviz de cada costalero.
- Recopilación y análisis de datos.
- Asignación manual de los costaleros a las trabajaderas. En base a estas alturas, se distribuyen los costaleros por las trabajaderas del paso, sabiendo que la altura nunca ha de superar la altura de la trabajadera, y en el caso de que la altura sea menor a la de la trabajadera objetivo, hay que compensar esa diferencia con unos tacos de madera o suplementos.

- Pruebas y ajustes de distintas combinaciones hasta alcanzar la que satisfaga al capataz.

Este proceso para ser finalizado con éxito requiere de mucho tiempo y esfuerzo, ya que puede haber muchas combinaciones válidas. Además, ¿cómo se puede garantizar que la opción elegida por el capataz sea la óptima? ¿hay otra combinación que permita minimizar el uso de tacos y por tanto que la inversión en estos sea la mínima?

La principal aportación del presente trabajo es dar respuesta a estas dos preguntas, creando una solución tecnológica que, mediante optimización matemática, pueda distribuir a los costaleros de forma óptima dentro de un paso de Semana Santa, y por tanto pueda serle útil al capataz de una cofradía para gestionar su cuadrilla, poniendo la tecnología al servicio de la tradición.

Tras entender la problemática existente, el desarrollo de esta aplicación supone un punto de inflexión a la hora de gestionar cuadrillas de costaleros, pues tras un exhaustivo análisis del estado del arte, la carencia de herramientas tecnológicas que cubran estas necesidades, indica la existencia de un nicho de mercado con oportunidades aún por explotar.

El eje fundamental de este TFG no es otro que la transformación de un mundo tan inmovilista como el cofrade, donde la gestión de una cuadrilla de costaleros siempre se ha realizado de forma artesanal, mediante anotaciones en papel y basada en la experiencia del capataz, siempre bajo criterios totalmente subjetivos. Reemplazando esta forma de trabajar por una aplicación móvil que realice esta ardua tarea, se optimizará la distribución de los costaleros, obteniendo la mejor combinación posible bajo criterios objetivos, y lo más importante: realizando una labor que otrora ocupaba días, en cuestión de segundos.

Como punto final, la realización de este proyecto ha supuesto una modernización de la gestión de costaleros, pero también una mejora de la eficiencia en la distribución de estos en el paso, influyendo en una mejor permutabilidad dentro del paso si se requiere, al minimizar el uso de tacos. Estas nuevas posibilidades de innovación nunca deben verse cómo una pérdida de la tradición, sino como una ventana abierta a la evolución de una tradición con tanto arraigo que puede beneficiarse de los avances tecnológicos de nuestros días.

1.3. Estructura del trabajo

Este proyecto se ha estructurado en diferentes capítulos que parten de una visión global del problema y acaba con las conclusiones una vez la problemática del proyecto ha sido resuelta. En este capítulo introductorio, se hace una breve pero concisa descripción del problema, que se irá desgranando en capítulos sucesivos.

En el capítulo 2, se acota el contexto del problema, realizando una labor pormenorizada del problema a tratar, describiendo cómo se organizan las procesiones de Semana Santa, definiendo términos propios del argot cofrade, para poco a poco ir estudiando los distintos métodos de distribución de costaleros y llegar al problema principal de este trabajo fin de grado: como solucionar el rompecabezas del capataz a la hora de distribuir a los costaleros de forma óptima dentro de un paso de Semana Santa. Además, se hace una búsqueda en profundidad de soluciones en el mercado que satisfaga el problema, lamentablemente sin encontrar nada en el estado del arte que cumpla con las necesidades requeridas. Estas necesidades son confrontadas y comparadas entre sí.

En el siguiente capítulo, se entra a analizar de forma metódica el problema de la distribución de costaleros de forma óptima, adjuntando ejemplos reales de cómo se hace este proceso de forma artesanal, posteriormente modelándolo matemáticamente, y finalmente eligiendo el algoritmo más apropiado para distribuir correctamente los costaleros, de entre varias opciones, que también se detallan, así como justificando la solución escogida cómo preferente.

Los objetivos y metodología de trabajo seguidos para solventar el problema son los protagonistas del capítulo 4, donde se definen unos objetivos generales y otros específicos que debe cumplir el sistema, la metodología usada para organizar el proyecto, así como las herramientas usadas para darle forma a la aplicación.

El capítulo 5 versa sobre el diseño del producto y la especificación de requisitos, partiendo de un lenguaje natural para transformarlo en historias de usuario propiamente definidas, definición de actores, requisitos funcionales y no funcionales, y el prototipado de componentes, así como una detallada planificación por sprints, en los que se ha organizado este proyecto ágil.

El siguiente capítulo trata sobre el desarrollo e implementación de la aplicación, describiendo qué tecnologías pueden ser utilizadas para resolver nuestro problema, la arquitectura de la aplicación, el modelo de datos y un ejemplo real de cómo la implementación del algoritmo de distribución ha mejorado un caso real.

Finalmente, se hace una reflexión sobre todo lo realizado, incidiendo en el legado que puede dejar este trabajo para el mundo de la Semana Santa y cómo podría cambiar una mecánica anquilosada hoy en día, con el mero hecho de usar las nuevas tecnologías. Se deja la puerta abierta a líneas de trabajo futuras con las que el proyecto se podría extender, y también, se sugieren mejoras que podrían aumentar la calidad del producto.

2. Contexto y Estado del Arte

La Semana Santa forma parte indisoluble de la sociedad andaluza. No se explica dicha sociedad sin su relación con la Semana Santa y viceversa. Trasciende lo puramente religioso y toca varias aristas como el arte, las relaciones sociales o la historia. En la mayoría de los rincones de Andalucía el movimiento cofrade es el movimiento asociativo que más gente congrega de cuantos tienen lugar en cualquier municipio. Se puede decir que el año gira en torno a la Semana Santa y que esta es símbolo de identidad cultural: bandas de música que ensayan la mayoría de los meses del año, cererías que trabajan para que las velas estén disponibles llegada la hora, bordadores, tallistas, juntas de gobierno de Hermandades y Cofradías que celebran sus actos para financiar proyectos… y un sinfín de elementos que constituyen el engranaje del mundo cofrade. (Briones Gómez, 1983)

En este capítulo se aporta una visión del contexto en el que tiene lugar este proyecto, tratando de identificar el problema al que se quiere dar solución, investigando diferentes vías de solución, además se hará una búsqueda exhaustiva de lo que ofrece el mercado por si hubiera algo disponible que se pueda mejorar o extender, o si por el contrario tenemos que crear una solución desde cero. Se partirá de una visión general de una Hermandad hasta describir cómo funciona una cuadrilla de costaleros, pormenorizando en el problema que nos ocupa: la gestión de alturas, y la solución propuesta.

Además, se aportará un estudio de las soluciones tecnológicas existentes en el mercado para resolver este problema, si las hubiera, lo cual ayudará a entender el porqué de esta temática para este proyecto.

2.1. Análisis del contexto

La Semana Santa es la celebración cristiana que conmemora la Pasión, Muerte y Resurrección de Jesucristo. Se celebra desde el Domingo de Ramos hasta el Domingo de Resurrección. Esta celebración no tiene una fecha fija en el calendario, sino que se establece en base a la fiesta de la Pascua judía que recordaba la liberación de la esclavitud de Egipto, el día de la primera luna llena de la primavera. Esta fecha se movía por el año lunar y no por el solar, de ahí que la Semana Santa cambie de fecha cada año. (Vallés, 2018).

Año	Fecha del Domingo de Ramos
Semana Santa 2025	**13 de abril 2025**
Semana Santa 2026	**29 de marzo de 2026**
Semana Santa 2027	**21 de marzo de 2027**
Semana Santa 2028	**9 de abril de 2028**
Semana Santa 2029	**25 de marzo de 2029**
Semana Santa 2030	**14 de abril de 2030**
Semana Santa 2031	**6 de abril de 2031**
Semana Santa 2032	**21 de marzo de 2032**
Semana Santa 2033	**10 de abril de 2033**
Semana Santa 2034	**2 de abril de 2034**
Semana Santa 2035	**18 de marzo de 2035**

Tabla 1. Fechas de la Semana Santa en los próximos años. Fuente: calendario-365.es

La Semana Santa tiene lugar en dos espacios físicos diferenciados: en el templo donde se celebran los actos litúrgicos asociados a la Semana Santa cómo el Triduo Pascual, besamanos y besapiés o misas de Hermandad, y por otro lado también se exalta o tiene su punto álgido en la calle, cuando las cofradías salen en procesión. En estos desfiles procesionales se representa la Pasión y Muerte de Cristo y el sufrimiento de la Virgen María. Esta representación se materializa en los llamados pasos, grandes estructuras o plataformas con imágenes sagradas. Las cofradías pueden contar con uno, dos o tres pasos en su procesión: de Cristo o de virgen, que generalmente va bajo palio. (Ugarte Pereira, 2019)

El recorrido de la procesión suele transitar lugares de culto o de fe como iglesias, catedrales, casas de hermandad, o lugares de importancia civil como la casa consistorial. La estructura de una procesión está sumamente cuidada y cada elemento que forma parte de ella, léase cruz

de guía, nazarenos, hermanos infantiles, cruces de penitencia, músicos, presidencia, etc. tienen su lugar específico en el discurrir de la procesión. (López-Guadalupe, 2018).

La Semana Santa se estructura en Hermandades o Cofradías, las cuales tienen el encargo y la responsabilidad no solo de procesionar su sagrado titular en Semana Santa, sino de mantener una vida de hermandad todo el año. Una Hermandad es una asociación pública de fieles cristianos que se unen para promover el culto en torno a un misterio de la pasión, muerte y resurrección del Señor en nombre de la Iglesia. (Obispo Segorbe, 2019).

Los hermanos de una cofradía o hermandad están registrados en el archivo de esta y cómo tales tienen derechos y obligaciones. Uno de los derechos es ser partícipes de la procesión en Semana Santa de cualquiera de las maneras que se les requiera cómo por ejemplo de costalero o de nazareno. En cuanto a las obligaciones, se puede decir que una de ellas es estar al corriente del pago de la cuota anual o semestral.

A su vez, dichas hermandades y cofradías se agrupan para fines representativos, organización interna y de defensa del bien común ante entes superiores, que gestionan los desfiles procesionales, garantizando cumplimiento de horarios, organizando los itinerarios de los desfiles procesionales y resolviendo los conflictos que pudiera haber entre hermandades. (Consejo General de Hermandades y Cofradías de la Ciudad Sevilla, 2012)

2.1.1. Procesión de Semana Santa

Las procesiones de Semana Santa son desfiles religiosos en los que se representa la Pasión y Muerte de Cristo. (*Las procesiones de Semana Santa - Arguments*, 2019).

Estas representaciones nacen en España de mano de las cofradías y hermandades de penitencia en el siglo XII, aunque toman especial relevancia a partir del siglo XVII, cuando estos desfiles añaden al carácter devocional un propósito de súplica colectiva ante desastres como la epidemia de la peste negra que asoló Sevilla en 1649. (Gómez, 2018).

En la actualidad estas procesiones se siguen celebrando de una manera muy similar, aunque adaptadas a los nuevos tiempos, incluyendo en los desfiles procesionales elementos como bandas de música, cámaras de vídeo o faroles, inimaginables en sus orígenes por las limitaciones propias de la época en cuestión.

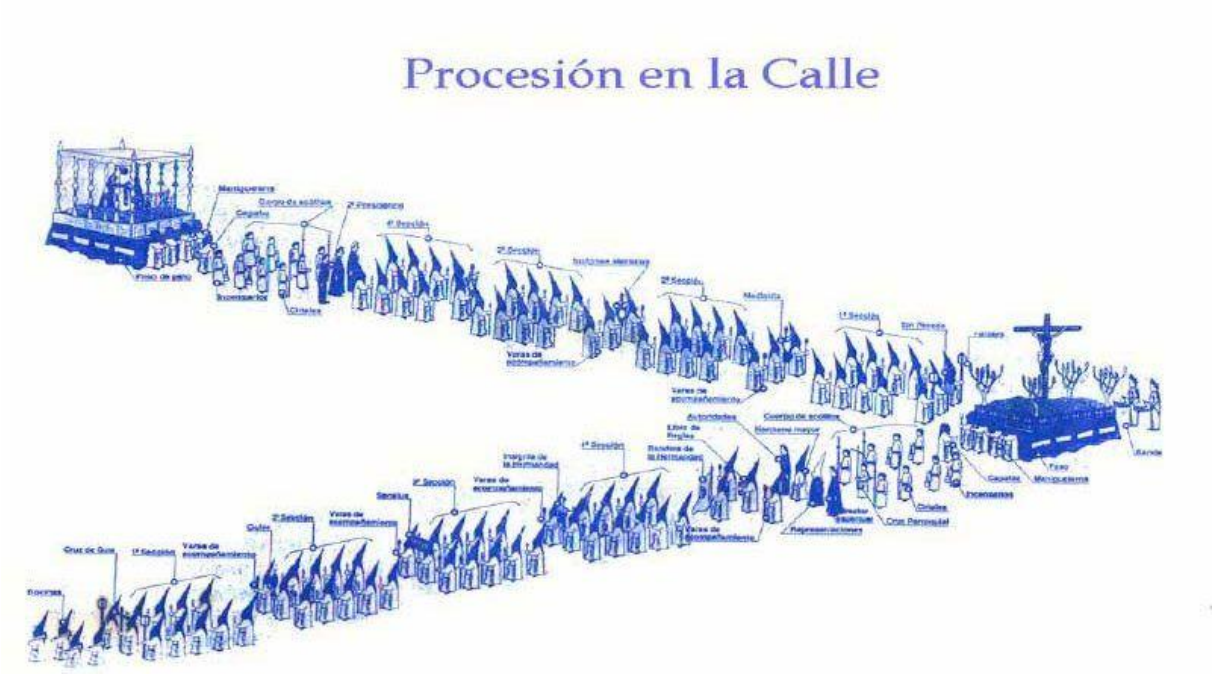


Figura 1. Esquema de una procesión en la calle. Fuente: Blog Güichi de Carlos.

A grandes rasgos una procesión consta, en orden de paso, por la cruz de guía, flanqueada por dos faroles. A continuación comienzan los tramos de nazarenos, separados por el diputado de tramo que siempre se encuentra en el centro del desfile. Estos nazarenos generalmente suelen portar cirios. Su indumentaria es la más característica de la Semana Santa, constando de una túnica del color de la Hermandad, un cíngulo de esparto o de tela a la altura de la cintura, y el elemento más identificativo: un capirote que cubre el rostro en señal de recogimiento y proporciona anonimato para hacer de la procesión una estación de penitencia libre de protagonismo. (Peñalver, 2022).

Entre estos tramos de nazarenos, y antes del paso sobre el que se posa la imagen a procesionar, se portan las insignias de más valor de la hermandad, cómo un simpecado, una bandera representativa o el libro de reglas de la hermandad, entre otras.

Justo antes del paso, se colocan los acólitos que acompañan al paso portando cirios de gran altura que forman parte del acto litúrgico. Delante del paso, se coloca el capataz, que es la figura que tiene el encargo de dirigir el paso por las calles y que va dando instrucciones a los costaleros que están situados dentro del paso procesional y que portan el paso con gran pericia. Por último y tras el paso se sitúa la banda de música que acompaña al paso

interpretando marchas procesionales. No todas las hermandades llevan banda de música, ya que algunas son de riguroso luto y por tanto el único sonido que tienen es el silencio de todo el cortejo procesional. (Rodríguez, 2022).

En lo que respecta a la organización interna del cortejo procesional, se hace de la forma más justa posible; esto es, respetando el orden de antigüedad en la hermandad. Este orden de antigüedad viene marcado por la fecha en la que un hermano se dio de alta en la cofradía. Es decir, un hermano más antiguo irá más cerca del sagrado titular que un hermano más nuevo, estando los nazarenos ordenados por riguroso orden de antigüedad en todo el desfile procesional. (Carlos López Bravo, 2023)

Algunas cofradías también trabajan con el denominado orden de antigüedad en el puesto, que se usa para aquellos puestos que no son filas de nazarenos cómo la cruz de guía, faroles u otras insignias, estableciendo que una vez que se ha asumido esta responsabilidad tendrá el derecho adquirido de permanecer en el puesto siempre que manifieste la voluntad de participar en la procesión, reforzando la continuidad y estabilidad del cortejo procesional.

2.1.2. Cuadrilla de Costaleros

Si bien la organización de un cortejo procesional es responsabilidad única de la Junta de Gobierno de la Hermandad en cuestión, la gestión interna de una cuadrilla de costaleros reside única y exclusivamente en la figura del capataz. El capataz es la persona designada por la Hermandad para guiar al paso procesional en la calle, y sirve de ojos de los costaleros. Dicha figura es fundamental ya que, ayudada por sus auxiliares, reparte a los costaleros debajo de las trabajaderas según su experiencia, para llevar a cabo la empresa repercutiendo en el mayor bienestar posible para la cuadrilla de costaleros. (Ismael Figueroa, 2018).

Además, el capataz gestiona todo lo que tiene que ver con la cuadrilla de costaleros, y hace de nexo entre estos y la cofradía. El capataz, cómo se menciona más arriba, es designado por la cofradía, pero este tiene potestad para designar a su equipo de trabajo, que, eso sí, debe ser ratificado por la Junta de Gobierno de la hermandad. Normalmente el equipo de trabajo se suele componer de uno o dos capataces auxiliares, también llamados contraguías.

Figura 2. Paso malagueño. Fuente: De museos por Málaga

Hay que discernir aquí entre los dos estilos más arraigados de portar imágenes sagradas en Andalucía. Por un lado está el estilo malagueño, dónde hay que hablar de **tronos**, que son mayores en tamaño y peso, y donde no se hacen llamar costaleros sino **hombres de trono** colocados de manera longitudinal por el exterior del trono, bajo unos salientes llamados varales, portando el trono con el hombro, yendo a cara descubierta y vistiendo la túnica de la hermandad.

Figura 3. Paso estilo sevillano. Fuente: Gente de Paz

Por otro lado, existe el estilo sevillano, uno de los más populares, dónde no se habla de tronos sino de **pasos**, y a su vez se habla de **costaleros** y no de hombres de trono. En este estilo, los costaleros se colocan debajo del paso, siendo invisibles para el público. No se organizan en varales sino en trabajaderas, una especie de vigas horizontales de lado a lado del paso, dónde los costaleros portan el paso con su cerviz usando para ello una prenda llamada costal, un saco grande de tela, en el que comúnmente se transportan granos, semillas u otras cosas.

La denominación de **costalero** proviene de la palabra "costal", una prenda que los costaleros colocan en su cuello y hombros para amortiguar el peso de lo que cargan. En la Sevilla antigua, los costaleros eran trabajadores del puerto, que ofrecían sus servicios para transportar los pasos de Semana Santa. Con el tiempo y debido al auge de la Semana Santa, esta labor pasó de ser una labor profesional a ser una labor voluntaria o de fe, asumiendo este encargo los fieles y devotos de las cofradías de Semana Santa. (Almería noticias, 2014)

Figura 4. Interior de un paso de Semana Santa. Fuente: CiberEcija.com

Cómo se aprecia en la imagen superior, un paso de Semana Santa se compone de un conjunto de trabajaderas, que se sitúan de forma horizontal soportando el peso de todo el paso procesional. En el ejemplo de la imagen, se aprecian 7 trabajaderas con 5 huecos cada una, necesitando 35 costaleros cómo mínimo para poder mover el paso de forma óptima.

¿Cómo se colocan los costaleros dentro de un paso? Este es un proceso bastante meticuloso en el que cada capataz intenta implantar un estilo propio basado en su trayectoria. Todo parte de la celebración de la ***igualá***, un evento en el que el capataz y su equipo miden la altura de los aspirantes. La altura de cada trabajadera es fija, por lo que hay que ser muy preciso colocándolos, ya que si el costalero no llega a la altura de la trabajadera hay que suplementarlo con tacos de madera para compensar esa falta de altura.

Figura 5. Costalero siendo tallado. Jorge Molina (Cedida)

Este proceso busca poder asignar a cada costalero una posición óptima dentro del paso, ya que un error en la colocación de un costalero puede ser fatal en términos de confort dentro del paso, o incluso producir lesiones físicas.

Siguiendo con el proceso, dentro de un paso de Semana Santa se pueden desempeñar diferentes roles. Por un lado están los **pateros**, que son aquellos costaleros situados en las cuatro esquinas del paso, cuya importancia reside en ser aquellos que marcan los movimientos durante los giros, que corrigen la trayectoria del paso y a quiénes van dirigidas las órdenes del capataz. Los **costeros** son los costaleros que igualan en los lados del paso. Es decir, su número duplica el número de trabajaderas, y se llamarán a ellos mismos costeros izquierdos o derechos según en el flanco en el que vayan situados, exceptuando a los pateros, citados anteriormente. Los **fijadores**, a su vez, son los costaleros que se sitúan al lado de los

costeros y pateros, y su función principal es ayudar a éstos en lo que se requiera. Por último están los corrientes, que son aquellos que se colocan en la parte central del paso, entre los fijadores de ambos flancos, también llamados costeros. Este nombre proviene de la situación antaño de los desagües en las calles, que se situaban en el centro de estas. Este rol cuando la trabajadera tiene menos de cinco huecos, no se usa, habiendo solo costeros y fijadores. (P. Coto, 2009)

Delantera del paso				
Patero	**Fijador**	**Corriente**	**Fijador**	**Patero**
Costero	**Fijador**	**Corriente**	**Fijador**	**Costero**
Costero	**Fijador**	**Corriente**	**Fijador**	**Costero**
Costero	**Fijador**	**Corriente**	**Fijador**	**Costero**
Costero	**Fijador**	**Corriente**	**Fijador**	**Costero**
Patero	**Fijador**	**Corriente**	**Fijador**	**Patero**
Trasera del paso				

Tabla 2. Esquema de roles en un paso.

Además de estos roles, según la tipología del paso, también puede existir la figura del vocero, que es aquel costalero que implementa los cambios de paso según los cambios en la música, y a los que toda la cuadrilla responde aplicando el paso correspondiente. (Figueroa, 2018)

2.1.3. Métodos de distribución de costaleros

Cuando llega la hora de colocar costaleros debajo de un paso de Semana Santa, existen diversos patrones que se usan según el tipo de paso dependiendo de cómo esté repartida la carga. Si es un paso de palio, generalmente la carga pese a ir repartida proporcionalmente, incide más en la delantera del paso por la colocación del juego de velas del paso procesional.

Si es un paso de cristo donde la imagen está colocada en el centro del trono, las trabajaderas centrales serán las que tengan que reforzarse un poco más. Si es un misterio con figuras

repartidas por todo el trono, habría que ver específicamente dónde incide más el peso para reforzar esa trabajadera.

Por esto, existen varios sistemas de distribución de costaleros, ya que los casos de uso pueden ser diversos, interviniendo variables cómo la altura de los costaleros, el tipo de paso, el movimiento de este e incluso las dinámicas físicas que se forman.

A continuación se detallan los tres métodos más arraigados a la hora de distribuir los costaleros debajo del paso. (Ignacio Díaz Pérez, 2012)

2.1.3.1. Método descendente

Es el método más arraigado, dónde los costaleros de mayor altura se sitúan en la delantera del paso, mientras que los más bajos se sitúan en la trasera de este. Esta organización en recorridos con pendientes es más eficiente

2.1.3.2. Método en "V"

En este método los costaleros de menor altura se sitúan en las trabajaderas centrales y los de mayor altura en la delantera y la trasera del paso, creando equilibrio y distribuyendo de forma homogénea el peso. Bien es cierto que para llevar a cabo este tipo de reparto se requiere de una gran cantidad de costaleros de gran altura, y es uno de los más usados.

2.1.3.3. Método en "W"

Evolución del sistema explicado anteriormente, en el que las trabajaderas de los costaleros de menor altura se sitúan intercaladas entre los de mayor, estando estos en la delantera, trasera y posiciones intermedias del paso. Es el método dónde se consigue la distribución del peso más uniforme, además de sostener el paso por diferentes puntos, aportando la mayor estabilidad posible.

En contrapartida, este método es difícil de aplicar ya que necesita de muchos costaleros de gran altura, siendo difícil llevar a cabo esta empresa al tener la exigencia de que las alturas sean homogéneas en su mayoría.

2.1.4. El rompecabezas del capataz

Cómo se ha explicado en las secciones anteriores, dentro de un paso de Semana Santa se representan diferentes roles, siendo los más importantes los cuatro pateros, pues son los de

más importancia al ser aquellos que tienen la misión de determinar el giro de un paso o de corregir la trayectoria de este. Obviando los pateros, es decisivo garantizar que las posiciones restantes asignadas cumplan con el supuesto de altura; esto es, que la altura de cada costalero nunca exceda la altura de la trabajadera correspondiente. Es la única restricción posible en todo este proceso. Dicho proceso, puede llegar a generar innumerables combinaciones que cumplan con el requisito de altura, pero también es muy complicado determinar cuál es la combinación óptima, al no efectuarse ningún tipo de verificación de coste mínimo. Se puede afirmar que cuando una combinación es del gusto del capataz, el proceso termina, aunque ello conlleve incurrir en errores o mejor dicho, en no elegir la mejor solución.

A su vez, el problema se agranda cuando las alturas de los costaleros son heterogéneas, recurriendo al uso de unas herramientas de compensación de alturas llamadas tacos, para mitigar esta diferencia de altura y lograr la altura adecuada. Aquí entra en la ecuación otro axioma fundamental en este proceso de colocación de costaleros, que no es otro que minimizar el uso de tacos, ya que mejorará la movilidad interna de los costaleros durante el recorrido, además de reducir costes, ya que los tacos requieren de un desembolso económico.

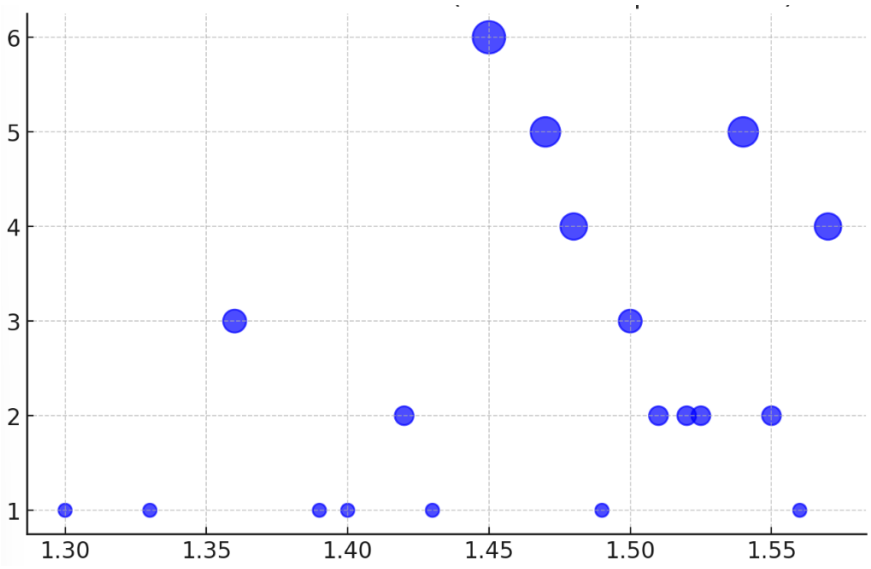


Figura 6. Distribución de costaleros año 2025. Fuente: creación propia

En la figura de arriba se puede ver gráficamente el problema de la distribución de los costaleros. Asumiendo que un paso puede tener entre 6 y 9 trabajaderas, solo se aprecia uniformidad de alturas en aquellos costaleros que miden 1.46, 1.45 y 1.54 dónde el número

de costaleros con dicha altura es mayor o igual a 5, pudiendo completar una trabajadera. El resto de las alturas está bastante disperso, teniendo que ajustar muy bien la distribución en las trabajaderas para que vayan lo más cerca posible de una trabajadera.

Con respecto a la movilidad interna de los costaleros, mencionada en párrafos anteriores, esto se hace para preservar la salud de estos, debido a que no todas las calles están al mismo nivel, presentando pendientes hacia el centro o hacia los lados, dependiendo de la ubicación del desagüe. Al minimizar el uso de tacos, y por tanto la movilidad interna de los costaleros, se puede, por ejemplo, hacer que un costalero que está situado en el costero en una calle con el desagüe en el centro, y por tanto, cargando más peso que uno situado en los huecos centrales del mismo, pueda intercambiarse con otro que vaya situado en los huecos centrales, para así compensar la carga en el total de la procesión, y en consecuencia, impactar positivamente en la salud y bienestar físico del costalero, evitando el riesgo de sobrecargas o lesiones.

De vuelta a este proceso, se puede afirmar que es un ejercicio de paciencia y dedicación por parte del equipo de capataces. No es algo excesivamente complejo pero sí muy tedioso al tener tantas posibles combinaciones en cada posición, interviniendo variables cómo la altura de costaleros y trabajaderas, pero también otras variables más del gusto del capataz o costalero cómo las preferencias personales (costalero que prefiere costero o costalero que prefiere ir de fijador o corriente, capataz que confía en ciertos costaleros para la delantera o la trasera del paso, etc.), además de las variables intrínsecas del entorno, también comentadas anteriormente cómo el desnivel de la calle o la nivelación de los desagües.

Hay otro punto que merece la pena resaltar, que son los contratiempos, de los que nadie está exento. Si en una procesión un costalero sufre un percance, hay que conocer la altura de cada miembro de la cuadrilla de aspirantes para reemplazar al costalero lesionado. Esto se hace de manera intuitiva, pero no parece ser el procedimiento más adecuado. Además, durante los ensayos que se realizan antes de la procesión para amoldar el cuerpo a la Semana Santa, y hacer probaturas, puede haber asistencias dispares, y en cada ensayo habría que hacer este reordenamiento dependiendo de la asistencia para manejar la fluctuación de costaleros entre ensayos.

Este proceso de asignación se ha demostrado que es efectivo, ya que los pasos de Semana Santa llevan procesionando desde hace décadas a la calle, pero sí es cierto que pese a ser

efectivo, el reparto casi nunca es el óptimo al no verificarse los criterios de coste mínimo. Un reparto será óptimo cuando la suma de las diferencias de las alturas de cada costalero con respecto a su trabajadera sea la menor de todas las combinaciones. Lógicamente, para validar esto hacen falta conocimientos matemáticos o tener una herramienta que te verifique si el reparto es óptimo.

Por tanto, se echa de menos un proceso automático que asigne los costaleros de forma automática en base a las restricciones citadas con anterioridad, todas cuantificables y fácil de integrar en cualquier algoritmo para obtener el resultado objetivo. Además, las variables de gusto personal, también descritas anteriormente, pueden ser incluidas en el algoritmo de reparto cómo condiciones a aplicar durante el mismo.

Este reparto automático para colocar costaleros dentro de un paso de Semana Santa es una solución viable y podría suponer un avance destacado en la gestión de las cuadrillas, ahorrando un tiempo muy valioso que podría ser dedicado a otros quehaceres de la gestión de la cuadrilla.

Relación de costaleros de costaleros año 2024							
Costalero 1	1.57	Costalero 10	1.52	Costalero 19	1.48	Costalero 28	1.48
Costalero 2	1.55	Costalero 11	1.52	Costalero 20	1.48	Costalero 29	1.48
Costalero 3	1.55	Costalero 12	1.51	Costalero 21	1.45	Costalero 30	1.45
Costalero 4	1.55	Costalero 13	1.51	Costalero 22	1.45	Costalero 31	1.45
Costalero 5	1.54	Costalero 14	1.5	Costalero 23	1.42	Costalero 32	1.42
Costalero 6	1.54	Costalero 15	1.5	Costalero 24	1.42	Costalero 33	1.42
Costalero 7	1.54	Costalero 16	1.5	Costalero 25	1.4	Costalero 34	1.4
Costalero 8	1.54	Costalero 17	1.49	Costalero 26	1.39	Costalero 35	1.39
Costalero 9	1.525	Costalero 18	1.48	Costalero 27	1.36	Costalero 36	1.36
Costalero 37	1.43	Costalero 38	1.42	Costalero 39	1.41	Costalero 40	1.43

Costalero 41	**1.41**	**Costalero 42**	**1.45**	**Costalero 43**	**1.48**	**Costalero 44**	**1.44**

Tabla 3. Relación de costaleros del año 2024. Cedida por el palio de la Virgen de la Caridad de Cabra (Córdoba)

2.2. Estado del arte

En esta sección se van a enumerar las diferentes aplicaciones tecnológicas que existen en el mercado, tanto de aplicaciones móviles cómo de páginas web para el sector comercial de la gestión de una cuadrilla de costaleros. Como se ha remarcado en la sección anterior, no se busca una aplicación que redescubra lo evidente, ni hacer nada que ya estuviera hecho con anterioridad.

La aplicación de la tecnología al ámbito de la gestión de cofradías se encuentra aún lejos del nivel de desarrollo de otros sectores tradicionales que sí se han sabido adaptar al mundo 3.0. No hay mucho software disponible y el que sí, no lo está de forma gratuita, ni siquiera para poder ver sus características principales o una demostración de este. Si esta gestión de cofradías está en una fase incipiente de desarrollo, un proceso tan artesanal y con tanto arraigo como el de la colocación de costaleros en un paso de Semana, aún está en una fase embrionaria, apoyándose en herramientas tradicionales cómo un papel y un bolígrafo.

Cómo se ha mencionado con anterioridad, realizar este proceso de forma manual y artesanal es un proceso romántico y de amor a las tradiciones, pero que no es perfecto al haber múltiples combinaciones, y en ciertos años dónde hay tallas muy dispares puede ser un verdadero rompecabezas colocar a todos los costaleros en el puzle porque cuando asignas un costalero a un sitio debajo del paso, probablemente el sitio que queda libre no se adapta igual de bien a otro candidato. Y así sucesivamente.

Con la premisa de identificar una solución que se adapte a las necesidades descritas, se van a evaluar una serie de herramientas que se han encontrado en una primera búsqueda. Estas herramientas son dispares: algunas son sólo web, otras son exclusivamente para dispositivos móviles y hay también alguna que tiene un objetivo más de red social que de gestión propiamente dicha.

2.2.1. CostalGo

Figura 7. Logo de CostalGo. Fuente: www.costalgo.com

CostalGo es "la herramienta definitiva para capataces y costaleros de Semana Santa" según se puede extraer de su página web, www.costalgo.com.

Es una aplicación móvil compatible con iOS y Android, en la que se pueden gestionar cuadrillas de costaleros asumiendo dos roles fundamentales: capataz o costalero. Además, ofrece un amplio reporting, un calendario de eventos, notificaciones personalizadas y una especie de red social, mediante la cual se puede buscar un paso y postularse para costalero tan sólo mandando una solicitud, similar a cuándo se solicita amistad en Facebook o se sigue a alguien en Instagram.

Poniendo el foco en lo que nos ocupa, la gestión de costaleros, la aplicación se desenvuelve bastante bien, el diseño es atractivo y la herramienta es funcional. Quizás un poco compleja de usar porque apunta muy alto, como se menciona más arriba está más orientada a ser una red social de costaleros y capataces que a una herramienta de gestión interna de tu propia cuadrilla de costaleros.

El registro es un registro clásico de usuario y contraseña, teniendo que esperar un e-mail de confirmación para validar el usuario.

Tampoco permite crear un paso desde cero sino que tienes que solicitarlo y los creadores de la aplicación te tienen que aprobar, por lo que se pierde agilidad al tener que hacer ese tipo de comprobaciones.

Volviendo a la gestión de costaleros, el punto débil viene cuando al empezar a usarla, leyendo las preguntas frecuentes, vemos que hoy en día no soporta una asignación de costaleros, ya que dice lo siguiente: "La primera *igualá* lleva más tiempo hacerla, pero una vez realizada, puedes guardarla y cargarla posteriormente en otro relevo".

Se puede considerar entonces que esta aplicación está más pensada en red social de capataces y costaleros que en gestión de una cuadrilla propiamente dicha, por lo que igual es apuntar demasiado alto ya que las necesidades que cubren este TFG son las inherentes a la colocación

de costaleros en un paso de Semana Santa de forma automática, eligiendo la opción óptima de todas las posibles. Es gratuita.

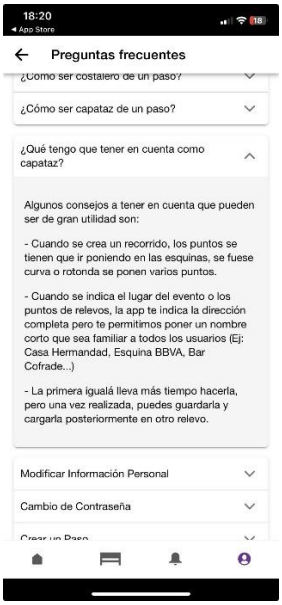


Figura 8. CostalGo no soporta asignación automática de costaleros

2.2.2. GesHdad

GestHdad, según reza en su página web, https://www.geshdad.com/productos/geshdad/, es el "software líder en la provincia de Sevilla y con presencia en toda España". Se ofrece cómo una solución global para las Hermandades y Cofradías. Inicialmente se concibió cómo mero gestor de hermanos en los años noventa, pero ha ido evolucionando hasta manejar todos los aspectos de una cofradía:

- ✓ Secretaría: gestiona todo lo relacionado con la Secretaría, léase base de datos de hermanos, listados, archivo histórico de actos celebrados por la hermandad, o una agenda de hermandades para el uso de correspondencia.
- ✓ Mayordomía: se ocupa de los aspectos económicos de la cofradía, gestionando las cuotas, y otros ingresos fijos y extraordinarios. También se engloba el uso de inventario.
- ✓ Salida procesional: Se va a trabajar con todo aquello que engloba la salida a la calle de la Hermandad, ya sea definición de los tramos que tendrá la procesión, el número de

penitentes o nazarenos por tramo, incluyendo la impresión y generación de papeletas de sitio.

Una vez revisada toda la documentación de la herramienta, la impresión es que no sirve para el cometido de este TFG, al centrarse única y exclusivamente en la gestión de una hermandad, sin hacer mención en el apartado de Salida Procesional a la gestión de costaleros de los diferentes pasos que conforman la Hermandad.

Otra potencial desventaja es que es una aplicación instalable, no en la nube, por lo que tampoco cumpliría con otro de los puntos fuertes de este cometido, la interoperabilidad.

Es de destacar, también, que esta aplicación no hace mejoras continuas, sino que todos los meses de octubre actualiza la versión del producto. Es decir, una actualización por año.

No es gratuita.

Figura 9. Funcionamiento de GesHdad. Fuente: www.geshdad.com

2.2.3. GHerCof

Ghercof, https://ghercof.com/, es una herramienta para la gestión integral de Hermandades y Cofradías. Funcionalmente es muy parecida a la aplicación anterior, pero esta tiene la mejora de ser multiplataforma, es decir, accesible a través de dispositivos móviles, tablets u ordenadores, basándose en tecnología web. Esta herramienta está pensada para que sea fácil de manejar, facilitando usarla en cualquier lugar con solo conexión a internet.

Se divide en seis módulos principales:

- ✓ Directorio. Gestiona el libro de hermanos y el contacto con otras hermandades.
- ✓ Económico. Abarca todo el apartado financiero de la hermandad como el cobro de cuotas o la generación de recibos.

- ✓ Procesión. Control de la salida procesional, incluyendo puestos en dicha procesión, gestión de papeletas de sitio.
- ✓ Comunicaciones. Conexión con los hermanos para el envío de correspondencia
- ✓ Otros. Tesorería o inventario.
- ✓ El futuro. Eventos o acción social.

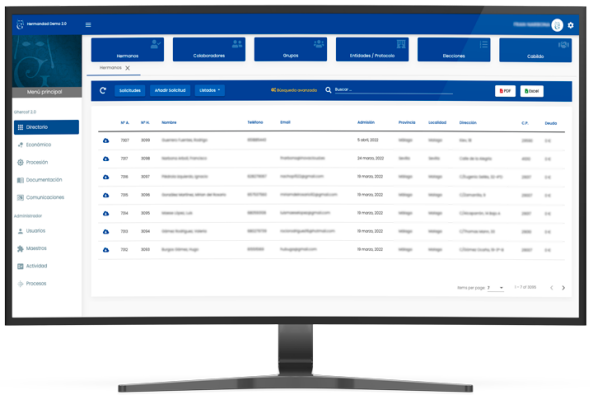


Figura 10. Módulo directorio de GherCof. Fuente www.ghercof.com

De nuevo, no tiene nada referente a la gestión de costaleros, solo se centra en la gestión integral de la hermandad, pero sin profundidad en los aspectos de la cuadrilla de costaleros.

Por otro lado, no tiene aplicación para dispositivos móviles sino que está basada en una web, estando disponible desde cualquier navegador de cualquier dispositivo.

No es gratuita.

2.2.4. HermandApp

HermandApp es el software de gestión de hermandades, asociaciones, agrupaciones y cofradías.

Se organiza en seis módulos principales:

- ✓ Secretaría. Gestiona la información de los hermanos. Además también en esta área se maneja la información referente a entidades colaboradoras, hermandades amigas y toda entidad.
- ✓ Tesorería. Gestión de cuotas y pagos automáticos.

- ✓ Salidas procesionales. Imprime y gestiona papeletas de sitio y solicitudes de cambio. Organización del cortejo procesional.
- ✓ Inventario. Gestión de enseres y túnicas de la hermandad.
- ✓ Utilidades. Panel de control de la aplicación, asignando permisos a los distintos usuarios que la utilizan.
- ✓ Miembros. Perfil del usuario que está conectado a la aplicación

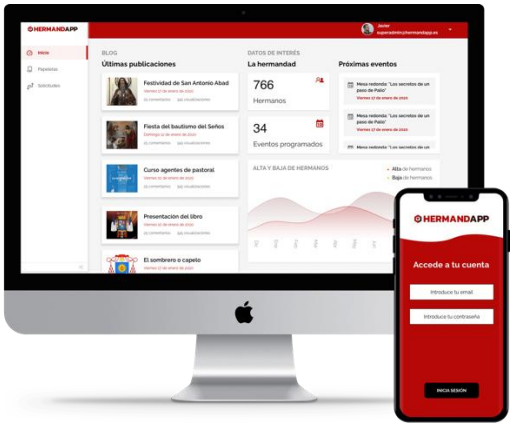


Figura 11. Panel de control de HermandApp. Fuente: hermandapp.io

De nuevo, nos encontramos en la tesitura de otra aplicación de gestión integral de hermandades y cofradías, que no tiene en consideración la gestión de cuadrillas de costaleros, por lo que tampoco cumple las expectativas depositadas

Dispone de un plan gratuito con acceso limitado a algunas características cómo gestión de la salida procesional, inventario o gestión de túnicas, además de limitar la base de datos de hermanos dependiendo del plan que se contrate.

2.2.5. Comparativa entre las diferentes aplicaciones

A continuación se presenta un resumen de las características principales de cada aplicación desgranada en la sección anterior. Cómo nota a destacar, se puede afirmar que pee a que todas abarcan grandes departamentos de una hermandad, casi ninguna llega al punto del objetivo que nos ocupa, la gestión de costaleros, y la única que los gestiona, carece de asignación automática, teniendo que colocar costalero a costalero en su hueco, por lo que no satisface el requerimiento principal.

	CostalGo	GesHdad	GHerCof	HermandApp
Gestión de costaleros	**Sí**	**No**	**No**	**No**
Asignación automática	**No**	**No**	**No**	**No**
Multiplataforma	**iOS, Android**	**No. Solo Pc**	**Sí. WebApp**	**Sí**
Actualizaciones	**Sí. Desatendidas**	**Anual**	**Sí**	**Sí**
Coste	**Gratuito**	**No Gratuita**	**No Gratuita**	**Gratuita con funcionalidad limitada**
Ventaja principal	**Red Social**	**Años de experiencia**	**Multiplataforma**	**Abarca muchísimos campos de la hermandad.**
Desventaja	**Requiere aprobación para crear un paso**	**No es multiplataforma.** **No se actualiza con frecuencia**	**No gestión costaleros**	**Limitación del plan gratuito**

Tabla 4. Comparativa de aplicaciones de gestión de hermandades.

2.3. Conclusión

De todas las herramientas investigadas, y siendo alguna de ellas bastante potente tanto funcional como visualmente, desafortunadamente ninguna se adapta a nuestras necesidades, debido a que ninguna provee la colocación de costaleros de forma automática.

Tras un análisis exhaustivo, es evidente que ninguna satisface las necesidades del problema propuesto. Estas soluciones, increíbles en lo técnico y en lo visual, no ofrecen el realizar una asignación automática de costaleros en los pasos de Semana Santa, punto principal del proyecto que nos ocupa.

Centrándonos en *CostalGo*, que es la aplicación que se queda más cerca de satisfacer las necesidades, permite crear fichas de costaleros registrando posiciones históricas dentro del paso procesional, pero se queda ahí, sin avanzar hacia el objetivo de la automatización total y completa. Por tanto, el proceso de colocación de costaleros en un paso de Semana Santa sigue siendo un proceso manual, ineficiente y costoso en tiempo. Es por esto por lo que se abre un hueco en el mercado de aplicaciones dónde con esta herramienta se puede llenar este vacío, resolviendo esta necesidad, y poniendo el valor el beneficio de que tener un reparto óptimo de costaleros repercutirá en beneficios cómo el menor desembolso en tacos para mitigar la altura, facilitar la movilidad o intercambio de posiciones dentro del paso, y la facilidad de hacer ajustes sobre la marcha, al estar las alturas repartidas de forma óptima.

Para finalizar, el contenido de este TFG está sobradamente justificado en el momento en el que una vez implementado el reparto de costaleros automáticos, este proceso se vea reducido a segundos en vez de a horas. Del mismo modo, la herramienta que se pretende crear es genérica y no abarca solo la realidad de un paso específico sino que se adapta a la generalidad, tan solo definiendo el número de trabajaderas, la altura de estas, y los huecos que tiene cada una, haciendo de la herramienta una herramienta flexible y que puede ser usada en cualquier contexto cofrade.

3. Análisis del problema y justificación del algoritmo elegido

En esta sección se explica el problema descrito en el análisis del contexto, mediante un sencillo ejemplo reduciendo el número de trabajaderas, huecos y costaleros a asignar, ya que la mecánica de asignación será la misma independientemente de la magnitud del problema. De ahí se irá desarrollando el problema desde lo específico hasta la generalidad, para, por último, ver qué algoritmos existen que pueden dar solución al problema de asignación de costaleros.

3.1. Resolución manual del problema

El ejemplo contempla 3 trabajaderas de 2 huecos cada una, con las siguientes alturas:

1. 170 cm
2. 165 cm
3. 160 cm

Siendo por tanto el número de huecos objetivos a rellenar 6.

Además, los aspirantes a costaleros que se disponen son los siguientes:

1. Antonio. 169cm
2. Pedro. 166cm
3. José. 162cm
4. Juan. 159cm
5. Fede. 155cm
6. Beatriz. 150cm
7. Javier. 170cm
8. Rafael. 165cm
9. Julián. 160cm
10. Manuel. 158cm

Con esta lista de aspirantes, y siguiendo el proceso detallado en el análisis del contexto, lo primero que habría que hacer es descartar, si hubiera, a aquellos aspirantes cuya altura sea mayor que la trabajadera de más altura, ya que no habría manera de que se les pudiera acomodar dentro del paso. Siendo la altura de la trabajadera más alta 170, el corte lo

tendríamos en esta altura por lo que de la lista especificada, todos serían aptos para ser asignados.

El siguiente paso es ordenar las alturas de los aspirantes de forma descendente, para una mayor facilidad a la hora de asignarlos de forma manual:

1. Javier. 170cm
2. Antonio. 169cm
3. Pedro. 166cm
4. Rafael. 165cm
5. José. 162cm
6. Julián. 160cm
7. Juan. 159cm
8. Manuel. 158cm
9. Fede. 155cm
10. Beatriz. 150cm

Para asignar la primera trabajadera, cuya altura es de 170cm, primero se buscan aspirantes con altura igual a dicha cota, encontrando a Javier, con 170cm y a Antonio con 169cm suplementado con 1cm.

Para la segunda, la cota está fijada en 165cm, por lo que habría que descartar a aquellos que ya están asignados y además aquellos que superen dicha altura. De la lista de arriba, Pedro quedaría excluido ya que su altura supera la de dicha trabajadera, además de ser incapaz de acomodarlo tampoco en la tercera, cuya altura es aún menor. La segunda trabajadera queda conformada de la siguiente manera: Rafael y José, este último suplementado con 3cm para igualar la altura.

La última trabajadera está a una altura de 160cm, descartando los ya asignados y aquellos cuya altura sea superior, nos quedarían Julián y Juan, este último con 1cm de suplemento.

El reparto final sería:

Delantera del paso	
Javier	**Antonio (+1)**
Rafael	**José (+3)**
Julián	**Juan (+1)**

Tabla 5. Ejemplo de asignación de costaleros de forma simple

Quedando fuera de la asignación:

1. Pedro. 166cm
2. Manuel.158cm
3. Fede. 155cm
4. Beatriz. 150cm

Pedro sería un buen candidato para la primera trabajadera con +4 si hubiera que darle descanso a alguno de los costaleros asignados. Manuel, por su parte, lo sería para la tercera con +2, mientras que Fede y Beatriz, con +5 y +10 respectivamente serían candidatos también para la tercera trabajadera pero ya con un número de centímetros a suplementar que se volvería difícil de gestionar.

Este proceso también se puede resumir en una tabla de coste de asignación de cada costalero con respecto a cada trabajadera.

Costalero	Diferencia T1 (170)	Diferencia T2 (165)	Diferencia T3 (160)
Javier (170)	**0**	**No válido**	**No válido**
Antonio (169)	**1**	**No válido**	**No válido**
Pedro (166)	**4**	**No válido**	**No válido**
Rafael (165)	**5**	**0**	**No válido**
José (162)	**8**	**3**	**No válido**

Julián (160)	**10**	**5**	**0**
Juan (159)	**11**	**6**	**1**
Manuel (158)	**12**	**7**	**2**
Fede (155)	**15**	**10**	**5**
Beatriz (150)	**20**	**15**	**10**

Tabla 6. Coste de asignación de cada costalero a una trabajadera

Observando la tabla es fácil deducir que la asignación óptima será aquella cuyo coste total sea el mínimo. Por tanto se puede asegurar que la asignación manual realizada es la óptima ya que el coste por trabajadera es:

- Trabajadera 1 (170cm): Javier (0) + Antonio (1) = 1.
- Trabajadera 2 (165cm): Rafael (0) + José (3) = 3.
- Trabajadera 3 (160cm): Julián (0) + Juan (1) = 1.

Siendo el coste total: 1 + 3 + 1 = 5.

Esta asignación es satisfactoria, debido también a los datos simplificados utilizados para una mejor comprensión del problema. Sin embargo, en un caso real la situación sería mucho más compleja. A pesar de su simplicidad, este ejemplo es válido para entender la mecánica del problema y su solución.

Cabe reseñar que esta asignación simple ha llevado alrededor de una hora de trabajo, sin haber probado todas las combinaciones, ya que muchas se podan al haber restricciones de altura o al haber sido ya asignados a otra trabajadera. Si extrapolamos este escenario a un caso real de 7 trabajaderas con 5 huecos cada una y un listado de aspirantes de 45, y de alturas más similares, el número de combinaciones sería bastante mayor, siendo un trabajo arduo y costoso en tiempo encontrar la solución óptima, justificando la implementación de un algoritmo de optimización automatizado para resolver el problema de forma eficiente.

3.2. Modelado matemático

El ejemplo detallado en el punto anterior se debe generalizar de forma matemática para automatizarlo y buscar algún tipo de algoritmo que lo resuelva de manera óptima o al menos se aproxime a una solución válida.

Primero se enumeran los parámetros que actúan en este proceso de asignación de costaleros:

- ***N***: número de trabajaderas.
- ***M***: número de huecos disponibles en cada trabajadera.
- ***K***: número total de costaleros.
- H_i: altura de la trabajadera *i*, donde $i \in \{1, 2, \ldots, N\}$.
- C_j: altura del costalero *j*, donde $j \in \{1, 2, \ldots, K\}$.
- A_{ij}: variable de asignación que especifica si el costalero *j* está asignado al hueco *i* de la trabajadera, siendo 1 si el costalero *j* está asignado y 0 si el costalero *j* no está asignado.
- d_{ij}: diferencia de altura entre la trabajadera *i* y el costalero *j* si la asignación es válida: $d_{ij} = H_i - C_j$, si $C_j \leq H_i$

El objetivo es minimizar la suma total de las diferencias entre las alturas de las trabajaderas y los costaleros asignados, tal que:

$$\min \sum_{i=1}^{N} \sum_{j=1}^{K} A_{ij} \cdot (H_i - C_j)$$

Donde:

- $H_i - C_j$ es la diferencia de altura entre la trabajadera *i* y el costalero *j*
- A_{ij}, dada su condición binaria, asegura que solo se consideren las opciones válidas, de lo contrario la multiplicación sería 0 y por tanto neutro para la suma.

Restricciones:

- Altura válida. Un costalero no puede sobrepasar la altura de la trabajadera objetivo

$$C_j \leq H_i \, , \forall_{i,j}$$

- Capacidad máxima. Una trabajadera tiene capacidad fija, es decir máximo M costaleros.

$$\sum_{j=1}^{K} A_{ij} \leq M, \qquad \forall \mathrm{i} \in \{1, 2, \ldots, \mathrm{N}\}$$

- Asignación unívoca. Un costalero solo puede estar en un hueco asignado al mismo tiempo.

$$\sum_{i=1}^{N} A_{ij} \leq 1, \qquad \forall j \in \{1, 2, \ldots, K\}$$

- Binariedad. Una asignación puede ser no asignada (0) o asignada (1).

$$A_{ij} \in \{0, 1\}, \qquad \forall i, j$$

La complejidad ciclomática es una métrica del software en ingeniería del software que proporciona una medición cuantitativa de la complejidad lógica de un problema. Por tanto, en este problema, asumiendo un paso con 7 trabajaderas, 5 huecos, y 40 aspirantes, se puede obtener partiendo de la formula básica de la complejidad ciclomática:

$$\boldsymbol{M} = E - N + 2P$$

Siendo ***M*** la citada complejidad ciclomática, ***E*** el número de aristas del grafo, ***N*** el número de nodos del grafo correspondientes a sentencias del programa y ***P*** el número de componentes conectados. (Ortiz, 2022)

Primero se calcula el número de iteraciones en el peor de los casos. Para ello se definen las diferentes iteraciones sobre las trabajaderas y costaleros:

1. Cada trabajadera tiene 5 huecos disponibles.
2. 40 costaleros disponibles.

Por tanto, $7 \cdot 5 \cdot 40 = 1400$ iteraciones como máximo.

Con respecto a las restricciones:

- Altura válida. (1400), obtenido del total de iteraciones.
- Capacidad máxima (7). Una restricción por trabajadera al ser la altura de esta fija.
- Asignación única (40), dado que es una restricción por costalero.

Ahora se obtienen los valores de la fórmula de la complejidad, que partiendo del nodo inicial, queda así:

$$\boldsymbol{N} = 1 + (1400 + 7 + 40) = 1448$$

Las aristas quedarían, incluyendo iteraciones y condiciones:

$$\boldsymbol{E} = 1400 + 1447 = 2847$$

P, al ser un algoritmo continuo en el que el flujo de control nunca se detiene, quedaría tal que $\boldsymbol{P} = 1$, que sustituyendo en la fórmula original:

$$M = E - N + 2P$$
$$M = 2847 - 1448 + 2 = 1401$$

La complejidad para 40 costaleros, 7 trabajaderas de 5 huecos cada una es de 1401, lo que infiere que hay 1401 soluciones distintas, siendo prácticamente imposible que una persona sea capaz de obtener la óptima, o si la obtuviera, no ser capaz de validar que esta sea la correcta.

3.3. Programación lineal

La programación lineal se puede resumir en la lucha o disputa de una cantidad de actividades por unos recursos de carácter limitado, de tal forma que se obtenga un máximo rendimiento, entendiéndose el rendimiento de dos formas:

1. Maximización, cuando lo que se persigue es maximizar el uso o los ingresos.
2. Minimización, cuando se intenta minimizar el coste de un objetivo.

Esta técnica es una de las más útiles para las casuísticas de optimización de funciones (Humberto Guerrero, 2022). Hay circunstancias en las que la complejidad del algoritmo que optimiza dicha función es tan alta que solo es abarcable mediante algoritmos genéticos de aproximación. (Profesor Hossein Arsham, 1994).

Se puede establecer una analogía entre uno de los problemas más famosos de programación lineal, el **problema del transporte**, y la asignación automática de costaleros en una trabajadera. En el problema del transporte se busca minimizar el coste de llevar productos desde un origen a un destino según las restricciones establecidas de oferta y demanda, siendo estas restricciones que no se puedan enviar más elementos de los que se tiene o que la

demanda sea satisfecha. Por su parte, el tema que nos ocupa en este TFG busca colocar un conjunto de costaleros en una posición específica, optimizando el uso de tacos, buscando que la diferencia de la suma de las alturas de cada costalero con respecto a su trabajadera sea la mínima, respetando las restricciones descritas en la sección anterior. (Bryan Salazar López, 2019).

Esta problemática, si lo que se busca es encontrar una solución válida asumiendo que esta es válida sí y solo sí se han cumplido todas las restricciones, puede pertenecer a la clase **NP**, ya que encontrar una solución que cumpla todas las restricciones se puede realizar en tiempo polinómico. (Leonardo Martínez Sandoval, 2022).

El problema se puede calificar como un problema de asignación óptima, fundamentales en la optimización combinatoria, dónde hay que distribuir de forma eficiente unos recursos en base a unas necesidades, minimizando el coste y maximizando las preferencias, sin incumplir ninguna restricción ni duplicar asignaciones. (Ávila Herrera, 1997).

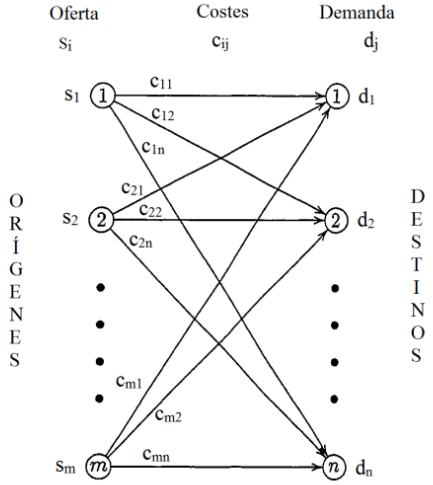


Figura 12. Grafo de un problema de transporte. Fuente: Manuel García Narváez - Universidad de Zaragoza

En contrapartida, se ha hablado de que lo que se pretende con este trabajo no es resolver el problema sino buscar una asignación óptima que minimice el uso de tacos, y por tanto la diferencia de alturas. Por ello, se define la optimización de un problema como la solución de este teniendo en cuenta un conjunto de restricciones y una función objetivo, buscando que esta solución minimice el valor de la función objetivo.

3.4. Algoritmos que dan solución al problema

En este punto se van a explorar diversos algoritmos que dan solución a la casuística expuesta anteriormente. Básicamente se pretende optimizar la asignación de costaleros para minimizar el coste sujeto a unas restricciones, garantizando una solución óptima, que no todos los algoritmos pueden obtener.

3.4.1. Back tracking o vuelta atrás

Los algoritmos de vuelta hacia atrás son algoritmos que encuentran soluciones que tienen una solución completa, donde el orden de los elementos no importa y en los que existen ciertas variables, asignándole a estas un valor teniendo en cuenta las restricciones impuestas por el propio problema. Dicho en otras palabras, es una estrategia en la que se investigan todas las posibles soluciones partiendo de un conjunto de valores inicial, para encontrar el resultado definido por el problema. Esta técnica se ayuda de la recursividad para hacer búsqueda exhaustiva. Suelen ser algoritmos muy ineficientes y costosos, ya que se usan para resolver problemas para los que no existe ningún algoritmo eficiente (José Javier Peleato, 2020).

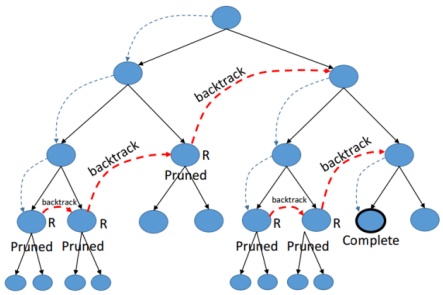


Figura 13. Esquema general de un algoritmo de back tracking. Fuente: Blog de Alberto Pascual

El back tracking garantiza una solución óptima, ya que explora todas las combinaciones. No se recomienda su uso para casos de uso dónde los problemas sean extremadamente grandes y donde se tenga que dar respuesta en un tiempo limitado, cómo se denota de la complejidad de este tipo de algoritmos: $O(n!)$.

Estos algoritmos de vuelta hacia atrás pueden ser mejorados con condiciones de poda, descartando aquellos caminos que no vayan a mejorar la solución actual.

3.4.2. Algoritmos voraces (Greedy)

Los algoritmos voraces son aquellos que obtienen soluciones a un problema a base de hacer elecciones localmente óptimas. Suelen usarse para resolver problemas de optimización, como el que nos ocupa, y en general son algoritmos eficientes y fáciles de implementar. (F. Moreno, 2019).

Este algoritmo, si bien no garantiza la solución óptima, sí se puede garantizar que el algoritmo da una solución que se queda muy cerca de la óptima. El porqué de no garantizar que la solución es la óptima es porque puede darse el caso de que el algoritmo se quede bloqueado en una solución localmente óptima, descartando otras soluciones que pudieran ser mejores.

En contrapartida, su complejidad $O(n \log n)$ lo hace muy eficiente para problemas de gran envergadura. Es fácilmente comprensible y de implementar.

Llevado a la práctica, consiste en ordenar las alturas de las trabajaderas, la lista de costaleros y el coste de ir de cada costalero a cada trabajadera. Después se asignan costaleros a las trabajaderas a los que se puedan asignar con menor coste, y se van llenando los huecos, descartando el costalero una vez esté asignado. Así sucesivamente hasta que no queden huecos que asignar.

3.4.3. Programación dinámica

La programación dinámica es un método para reducir el tiempo de ejecución de un algoritmo dividiéndolo en subproblemas más pequeños y almacenando sus soluciones para evitar repetir cálculos. Este tipo de programación se puede aplicar a problemas cuya solución óptima de un problema se puede construir a partir de las soluciones de sus subproblemas. (Javier Campos, 2021).

Uno de los mejores ejemplos para comprender la aplicación de la programación dinámica, es minimizar el coste del cálculo de N en la serie de Fibonacci. Manejando una lista auxiliar con el resultado de la serie para cada valor, se pueden evitar cálculos redundantes, y teniendo que calcular sólo el número actual, ya que el número Fibonacci es la suma de sus dos previos elementos (Shuheng.Ma, 2024).

La aplicación de la programación dinámica, en la práctica no sería muy distinta a la realizada para los algoritmos voraces, con la salvedad de que cada vez que un costalero sea asignado a

una trabajadera hay que guardar en la tabla cuál es el mejor resultado hasta este momento y evitar duplicar estas comprobaciones ya hechas.

El coste computacional es muy alto, debido a la cantidad de subproblemas que maneja, de ahí que su complejidad se asemeje a $O\ (n^2)$ o superior.

3.4.4. Algoritmo Kuhn - Munkres

El algoritmo de Kuhn – Munkres, también conocido como el método húngaro, es un algoritmo que encuentra el coste mínimo de un conjunto de tareas que deben ser realizadas por las personas más adecuadas. Se basa en la teoría de grafos y emplea una combinación de diversas técnicas de búsqueda exhaustiva y optimización para lograr la solución óptima. La raíz de este método es encontrar una matriz de costes reducida que dibuje las condiciones de asignación. (Mario Pastrana Moreno, 2012)

Para llevarlo a cabo, los pasos son los siguientes:

1. Inicio.
2. Construcción de la matriz.
3. Desarrollo de la matriz de costes.
4. Prueba de optimalidad
   4.1. Si es óptima, se acaba el algoritmo.
   4.2. Si no es óptima, se revisa la matriz y volvemos al punto 4.
5. Fin

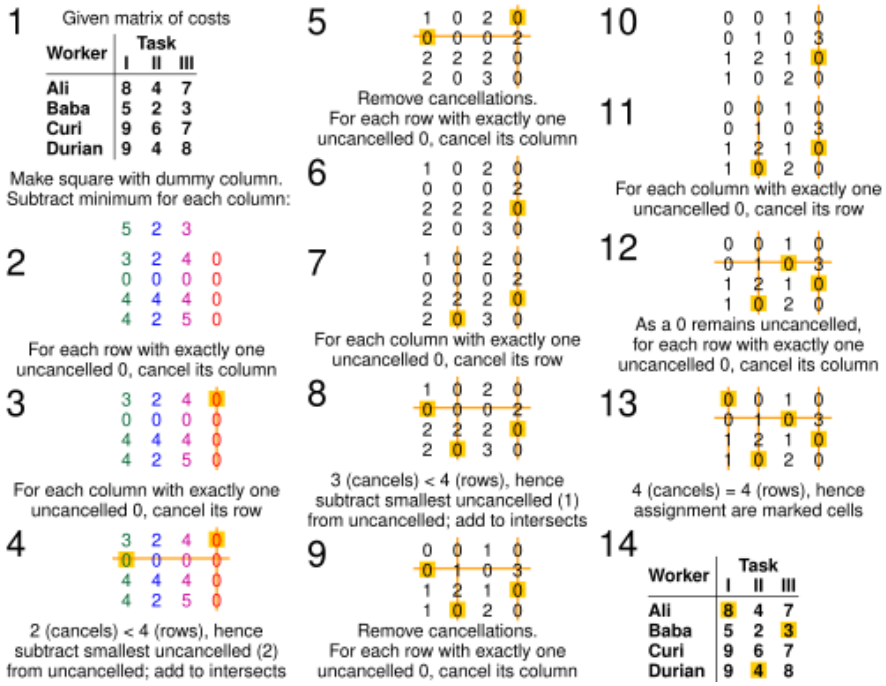


Figura 14. Iteraciones del algoritmo húngaro para un reparto de tareas. Fuente: Wikimedia Commons

Este algoritmo, para que sea óptimo requiere que la matriz de costes sea cuadrada, es decir, el mismo número de filas que de columnas, por lo que inicialmente no sería válido para la problemática que ocupa este trabajo. Si se tuvieran más aspirantes que huecos, la matriz no sería de N x N sino de N x M. Sin embargo, esto se puede mitigar convirtiendo esta matriz en cuadrada añadiendo huecos de coste infinito si se quiere buscar el coste mínimo o de coste 0 si se busca maximizar una función objetivo.

La complejidad de este algoritmo es de $O\ (n^3)$ y suele ser bastante eficiente para situaciones donde no hay muchos elementos.

3.5. Decisión final

Una vez evaluados algunos algoritmos que pueden resolver la complicación de la asignación de costaleros, se decide implementar el algoritmo de asignación húngara o de Kuhn – Munkres, debido principalmente a que obtiene una solución óptima y es eficiente por sí mismo.

Identificado el problema, y sabiendo que la muestra de datos con la que se va a trabajar no es excesivamente grande, el algoritmo de asignación húngara debe funcionar bien para el

propósito de este TFG. La programación dinámica se descarta por costosa y por lo inmanejable de las listas de asignación.

Se prescinde también de implementar back tracking, debido a su baja eficiencia, al tener que manejar todas las combinaciones posibles, pese a incluir podas que evitarían revisitar caminos que no lleven a la solución correcta.

Los algoritmos voraces, teniendo una eficiencia muy alta y siendo fáciles de comprender y de implementar, no se consideran tampoco debido a que no obtienen la solución óptima en todos los casos, aunque se acerquen, debido a lo mencionado más arriba sobre que se pueden bloquear en soluciones óptimas localmente.

Por tanto, se ha decidido resolver la parte nuclear de este TFG mediante el algoritmo húngaro o de Kuhn – Munkres.

4. Objetivos y metodología de trabajo

Este proyecto propone una herramienta tecnológica innovadora que viene a satisfacer una demanda creciente en lo que respecta a la gestión y organización de costaleros, poniendo énfasis en su colocación dentro del paso de una manera óptima. Tradicionalmente, este ha sido un proceso manual que se realizaba mediante ensayo y error, buscando una solución que gustara al equipo de capataces, pero que en ningún momento este método se probaba que garantizaba el mejor reparto final.

El objetivo principal de este proyecto no es otro que transformar lo tradicional en algo automático y moderno, dándole forma a esta demanda desde la concepción primigenia de la idea hasta verla materializada en algo tangible y medible cómo puede ser una aplicación multiplataforma.

Con el propósito de conseguir este objetivo general se definen a su vez unos objetivos específicos, que cumplidos uno a uno, guiarán a la solución correcta.

4.1. Objetivo general

El objetivo general de este TFG no es otro que el desarrollar una aplicación multiplataforma para la gestión de cuadrillas de costaleros de Semana Santa. Dicha aplicación debe ser capaz de crear, modificar, y borrar pasos de Semana Santa, dar de alta a costaleros que pertenezcan a una cuadrilla, así como colocarlos en el trono de una forma óptima minimizando el uso de tacos o de apoyos suplementarios. Debe permitir gestionar ensayos de forma eficiente, es decir, que se puedan mitigar los posibles imprevistos por falta de asistencia generando una disposición dentro del paso de forma automática en base al capital humano del que se disponga, así como otras funcionalidades clave que se detallan a continuación:

- Registro de capataces
- Creación y gestión de pasos
- Configuración de trabajaderas
- Mantenimiento de costaleros
- Proceso de asignación automática o manual, según se quiera, de costaleros en un paso.

4.2. Objetivos específicos

Para lograr el objetivo general, se establecen varios objetivos específicos, que cumplidos uno a uno, darán paso a la solución completa:

- **Conocer la demanda** exacta de diferentes equipos de capataces para que la solución propuesta sea aplicable no solo a la especificación sino a la generalidad del problema.
- **Investigar el estado del arte** del sector, justificando la temática de este trabajo en base a la existencia o no de soluciones que cubran esta problemática.
- **Desacoplar los datos del dispositivo** y guardarlos en la nube, estando accesibles para los usuarios de la aplicación sin importar del dispositivo que usen, mejorando la seguridad y el rendimiento de la aplicación.
- **Desarrollar una aplicación multiplataforma** para la gestión de costaleros de forma eficiente.
- **Crear una arquitectura descentralizada**, aprovechando la computación en la **nube** siempre que se pueda para reducir el acoplamiento entre sistema, permitiendo una arquitectura modular e independiente, mejorando la eficiencia al manejar información.
- **Prototipado**. Se diseña el producto en base a las necesidades, para que a la hora de acometer la implementación de la solución se sepa el qué y el cómo.
- **Planificación** del producto. El tiempo del que se dispone es limitado, para ello, siguiendo una planificación ágil dispuesta en *sprints* de una semana, se han estimado las diferentes historias de usuario en base a las 277 horas en las que está definido este TFG según la guía docente de la asignatura. (UNIR, Grado en Informática, 2021)
- **Trabajo futuro**. Detallar aquellas funcionalidades que, o bien no han sido tenidas en cuenta a la hora de desarrollar este proyecto, o no ha habido tiempo material de acometerlas.
- **Verificación del trabajo**. Adjuntar prueba de que el trabajo está funcionando correctamente para una casuística real. Para ello se comprobarán disposiciones de costaleros de años precedentes contra la asignación realizada por esta aplicación.
- **Conclusión**. Justificación de lo realizado, comprobando si los objetivos inicialmente planteados se han cumplido o no.

4.3. Metodología de trabajo

Existen dos principales paradigmas de desarrollo de software: desarrollo ágil y desarrollo en cascada. Mientras una metodología tradicional se centra en la planificación de una actividad de principio a fin, subdividiéndola en cómo definición de requerimientos, análisis, diseño, aprobación de diseños, implementación, pruebas y entrega, Scrum por su parte se basa en iteraciones cortas que entregan una parte del producto y no su completitud, para que a partir de esta el producto evolucione. La gran diferencia es que mientras que las metodologías tradicionales son rígidas ante cambios o imprevistos, las metodologías ágiles siempre están dispuestas al cambio al trabajar con iteraciones cortas, y por tanto, con compromisos a no tan largo plazo. (Rodríguez & Dorado Vicente, 2015)

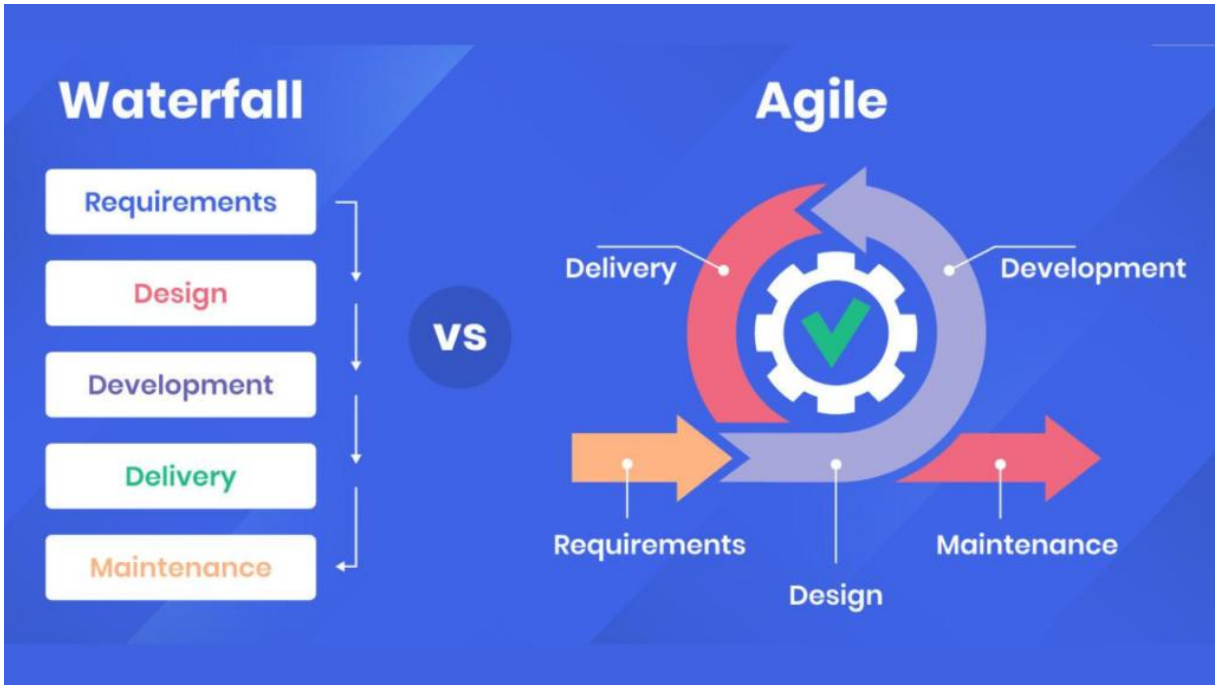


Figura 15. Comparación agile vs waterfall.

Nota. En la imagen se comparan ambos modelos mencionados. Centrándonos en el modelo de desarrollo ágil podemos ver cómo se destacan en las diferentes iteraciones el proceso de entrega de producto, ya que en cada iteración se irá entregando una versión del producto para ir obteniendo feedback del usuario final y por tanto poder anticiparse a los futuros problemas y mejoras. (Hardik, 2023).

4.3.1. Organización del proyecto

Se definen *sprints* de duración semanal, de lunes a domingo, con el compromiso de ir mejorando el producto iteración a iteración y sobre todo que en cada sprint el producto se mejore y funcione. Los *sprints* tienen duración fija, antes de iniciarlo hay que fijar cuál es el objetivo de ese sprint y a qué se va a comprometer el equipo. Este objetivo es invariable durante el transcurso del sprint. El *sprint* se compone de una serie de historias de usuario, basado en la capacidad de la que disponga el equipo. Esta capacidad no es fija, sino que puede variar entre sprints dependiendo de factores cómo disponibilidad de los recursos, prioridades, etc.

Los *sprints* contienen historias de usuario que han sido extraídas del *product backlog,* uno de los artefactos más importantes al trabajar en Scrum. El *product backlog,* es una lista ordenada de todo lo necesario para mejorar un producto, y sirve cómo la única fuente de trabajo que el equipo Scrum debe realizar. Los elementos que componen este producto backlog han de ser refinados y mejorados constantemente, añadiéndole granularidad y detalles cómo descripción, orden y tamaño. Este proceso de refinamiento es una actividad capital para la mejora continua del producto y ser capaz de tener tareas listas para ser desarrolladas. (Schwaber, s. f.)

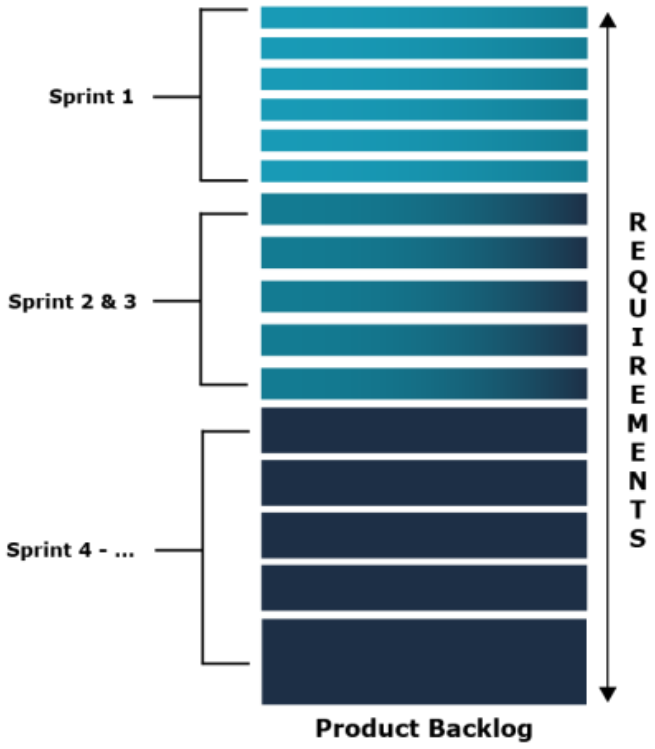


Figura 16. Product backlog. Scrum.org

Además, se establecen tres hitos en forma de mínimo producto viable, (de aquí en adelante MVP), que no es otra cosa que una versión básica y funcional de un producto, interfaz o servicio que se desarrolla incluyendo solo las funcionalidades esenciales para satisfacer a los primeros usuarios y obtener feedback valioso para el desarrollo final. Este enfoque permite validar nuestra idea y ver si se va en la dirección apropiada. (Hill, 2023).

Estos tres MVP que se van a desarrollar y que coincidirán con las entregas periódicas de este TFG darán lugar a nuestro primer mínimo producto comercializable (MMP), siendo este el primer lanzamiento del producto real, que puede satisfacer la necesidad de un usuario con un conjunto mínimo de funcionalidades. (Hernández, 2021).

Iteración	Fecha	Hito
Sprint 1	11 de noviembre / 17 de noviembre	
Sprint 2	18 de noviembre / 24 de noviembre	

Sprint 3	**25 de noviembre / 1 de diciembre**	
Sprint 4	**2 de diciembre / 8 de diciembre**	
Sprint 5	**9 de diciembre / 15 de diciembre**	**MVP1**
Sprint 6	**16 de diciembre / 22 de diciembre**	
Sprint 7	**23 de diciembre / 29 de diciembre**	
Sprint 8	**30 de diciembre / 5 de enero**	
Sprint 9	**6 de enero / 12 de enero**	
Sprint 10	**13 de enero / 19 de enero**	**MVP2**
Sprint 11	**20 de enero / 26 de enero**	
Sprint 12	**27 de enero / 2 de febrero**	
Sprint 13	**3 de febrero / 9 de febrero**	
Sprint 14	**10 de febrero / 16 de febrero**	
Sprint 15	**17 de febrero / 23 de febrero**	**MVP3**
Sprint 16	**24 de febrero / 2 de marzo**	
Sprint 17	**3 de marzo / 9 de marzo**	
Sprint 19	**10 de marzo / 12 de marzo**	**MMP**

Tabla 7. Planificación del proyecto por sprints.

Una vez se establece la duración de los sprints, como equipo de desarrollo toca también decidir los diferentes estados por los que va a pasar una tarea o historia de usuario. Para no

complicar demasiado se trabaja con la solución más simple posible, que no es otra que manejar tan solo tres estados:

- ✓ **Pendiente**. Tareas que aún no se han comenzado a desarrollar, pero que han sido priorizadas y comprometidas para finalizarse en el sprint.
- ✓ **En progreso**. Historias de usuario que han sido asignadas a un usuario y que están siendo desarrolladas en este momento.
- ✓ **Terminado**. Tareas completadas y que cumplen con lo acordado.

Las historias de usuario antes mencionadas son descripciones breves y simples de una funcionalidad del sistema, redactadas siempre desde la perspectiva del usuario final. El objetivo es capturar qué necesita el usuario y por qué, facilitando la comunicación entre el equipo de desarrollo y el representante del cliente en el equipo. (Alexander Menzinsky et al., 2022).

Patrón de redacción de Historia de Usuario	Ejemplo
Cómo [rol de usuario] **Quiero [objetivo]** **Para [beneficio]**	**Cómo gerente,** **Quiero generar informes** **Para ver el estado del producto**

Tabla 8. Ejemplo de creación de Historia de usuario

Estas historias de usuario una vez definidas, hay que estimarlas y priorizarlas. Estimarlas no es más que asociar a la historia de usuario una unidad de tiempo que determine de forma objetiva su complejidad. Normalmente se suele usar la serie de Fibonacci (0, 1, 2, 3, 5, 8, 13, 21, etc.) ya que esta serie es relativa al tamaño, es decir, ayuda a los equipos a evaluar la complejidad de las tareas en relación con otras. Al reflejar un crecimiento exponencial captura perfectamente la creciente incertidumbre y complejidad a medida que las tareas se van volviendo más grandes (Scrum Alliance, s. f.)

En lo que respecta a la priorización, esto se realiza en una ceremonia de *Scrum* llamada *Sprint Planning*, que por lo general se realiza el primer día del *sprint*, donde se revisa la historia de usuario y si todo está claro, se añade *al Sprint Backlog*, siendo el *sprint backlog* la cantidad de

elementos que van a estar disponibles para trabajar en el sprint actual y que por tanto el equipo se compromete a finalizar durante dicho *sprint*.

4.3.2. Fase de Discovery

Antes de comenzar a desarrollar el producto, hay que acometer una fase de *Discovery* o de descubrimiento de producto. Esta práctica, extendida en muchas organizaciones hoy en día, se realiza para cerciorarse de que el equipo de desarrollo sea plenamente consciente de las necesidades del usuario y a dónde quiere llegar. Al finalizar esta actividad el equipo será capaz de saber si merece la pena crear el producto y sobre todo, si es viable antes de empezar a trabajar en él. (Silvia U, 2022)

Para ello hay que ser capaces de responder a las siguientes preguntas:

- **Hipótesis de valor.** ¿necesita el cliente el producto que vamos a desarrollar?
- **Usabilidad**. ¿es usable el producto que vamos a crear?
- **Viabilidad técnica.** ¿tenemos los recursos necesarios para crear el producto?
- **Viabilidad de negocio.** ¿encaja este producto con la visión de negocio de la empresa?

El resultado de esta fase de Discovery no es otro que tener un producto propiamente dicho

4.3.3. Fase de especificación de requisitos

Esta fase se puede definir como el proceso del estudio de las necesidades de los usuarios para llegar a una definición de requisitos del sistema, así como el proceso inherente al estudio y refinamiento de dichos requisitos. («IEEE Recommended Practice for Software Requirements Specifications», 1998).

En esta etapa del desarrollo del software no solo interviene el equipo de desarrollo del producto, sino que hay que involucrar de manera activa a los usuarios finales, siendo estos los que mejor conocen las necesidades. El objetivo final no es otro que crear un documento de especificación de requisitos dónde se describa de forma legible y clara lo qué se requiere, sin ambigüedad y sin entrar a valorar en el cómo. Estos requisitos deben ser fácilmente comprobables para que puedan ser validados por el cliente. Asimismo, el resultado de esta fase actúa cómo contrato entre el equipo de desarrollo y el cliente, sirviendo además de guía para todo el proceso de desarrollo de software.

4.3.4. Fase de desarrollo

En esta fase se comienza a trabajar con lo que realmente es el producto tangible, dividido en dos partes bien diferenciadas, pero complementarias entre sí:

1. Diseño de arquitectura
2. Desarrollo del producto

En la fase de diseño de arquitectura, se toman decisiones críticas que van a definir la base tecnológica del proyecto, como la elección de la plataforma en la nube adecuada para trabajar con el resultado del desarrollo, o la elección del sistema de autenticación de usuarios de la aplicación garantizando un acceso seguro, así como la elección de la tecnología apropiada para este proyecto teniendo en cuenta las especificaciones y los objetivos.

Todas estas decisiones son sumamente importantes ya que dependiendo de estas se podrán determinar distintos patrones de diseño, las herramientas a utilizar, repercutiendo en un sistema más flexible, escalable, seguro y sobre todo mantenible a futuro.

Con respecto al desarrollo del producto, este se lleva a cabo siguiendo un desarrollo ágil, incrementando poco a poco el producto. Cómo cada historia de usuario ha de ser testeada y validada para ser cerrada, se ven ventajas cómo la no necesidad de una fase propiamente dicha de pruebas, ya que éstas son parte del desarrollo de cada historia de usuario. A su vez, estas entregas facilitan un feedback más temprano por parte del cliente, y por tanto, una detección precoz de cualquier punto que sea susceptible de ser cambiado o mejorado, bien porque no cumpla las especificaciones o porque se quiera ir en otra dirección.

4.4. Herramientas usadas

En esta sección se detallan las diferentes herramientas que se han utilizado para el desarrollo de este TFG. He puesto el foco en herramientas online para poder tener interconectividad al no trabajar siempre en el mismo ordenador, lo que me ha permitido despreocuparme de tener que usar siempre el mismo dispositivo físico para este trabajo fin de grado.

4.4.1. Miro

Para la fase de Discovery y definición de lo qué es lo que se necesita el cliente, se ha usado Miro, que es una plataforma de trabajo online para diseñar y crear flujos de trabajo en equipo

de forma remota. Esta aplicación se configura a base de proyectos, dónde se ha creado un proyecto para este trabajo. (*¿Qué es Miro? – Centro de ayuda de Miro*, s. f.)

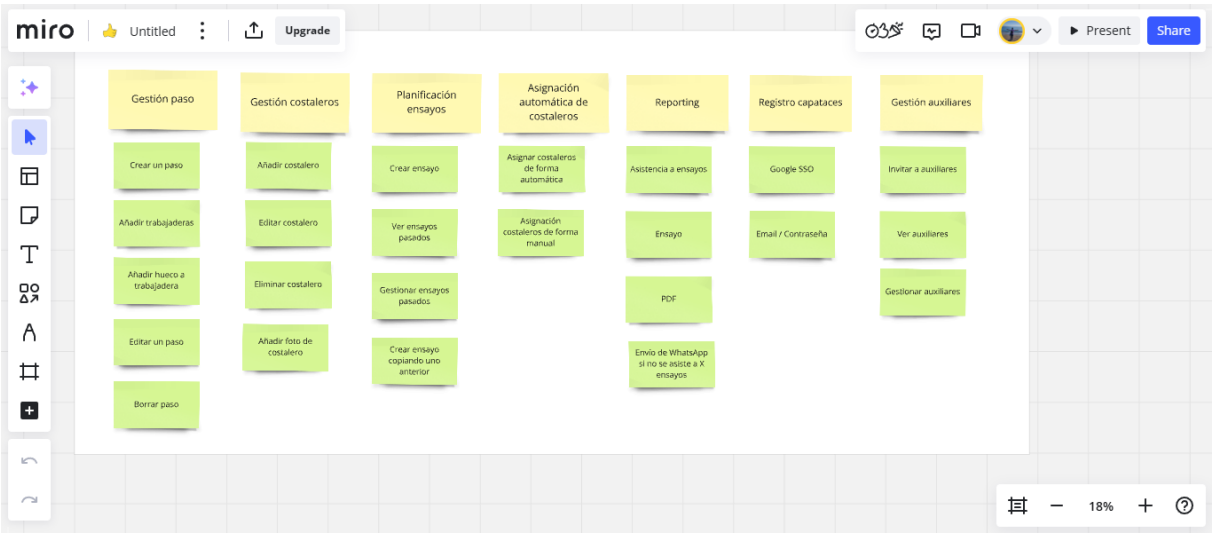


Figura 17. App de miro para este proyecto

Url: https://miro.com/app/

4.4.2. Visual Studio Code

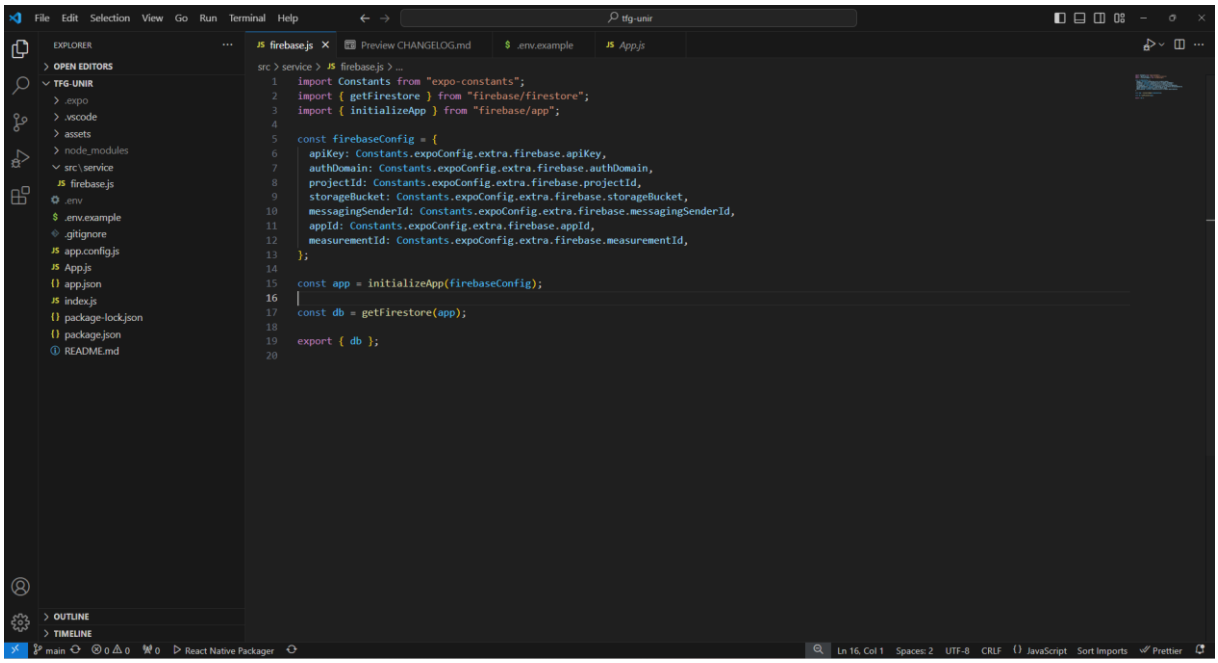


Figura 18. Visual Studio Code

4.4.3. Zotero

Zotero para la gestión de la bibliografía

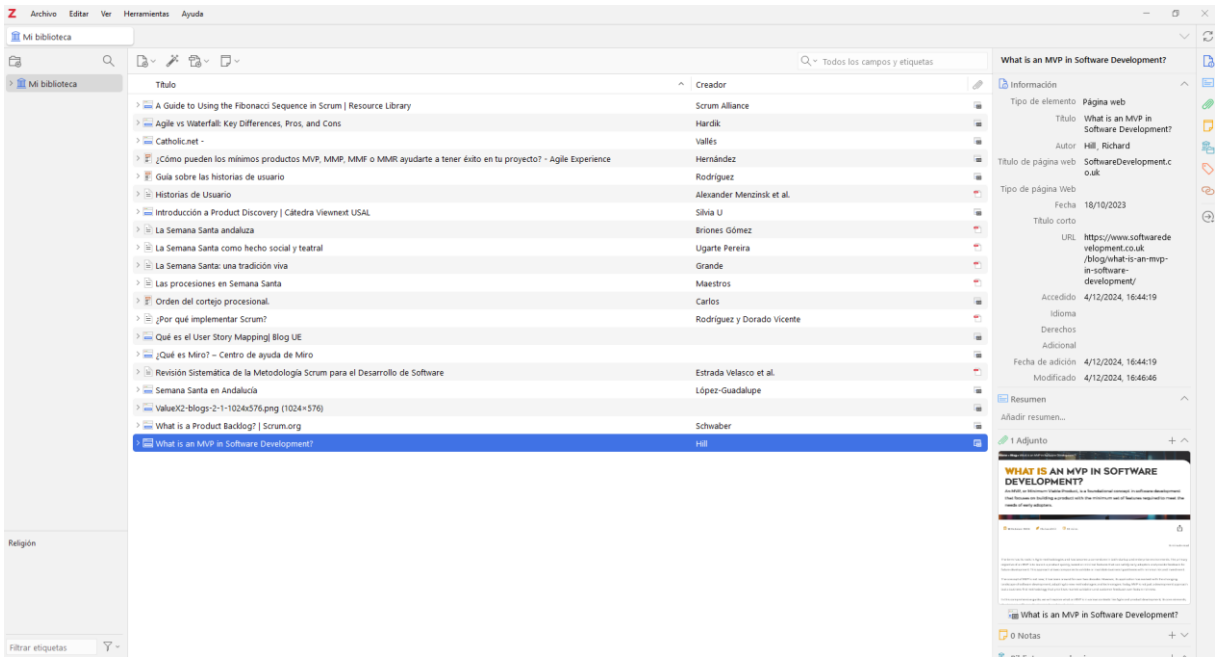


Figura 19. Zotero

4.4.4. MeuScrum

Esta plataforma online gratuita permite crearnos proyectos *Scrum* y gestionarlos de una forma bastante intuitiva y versátil.

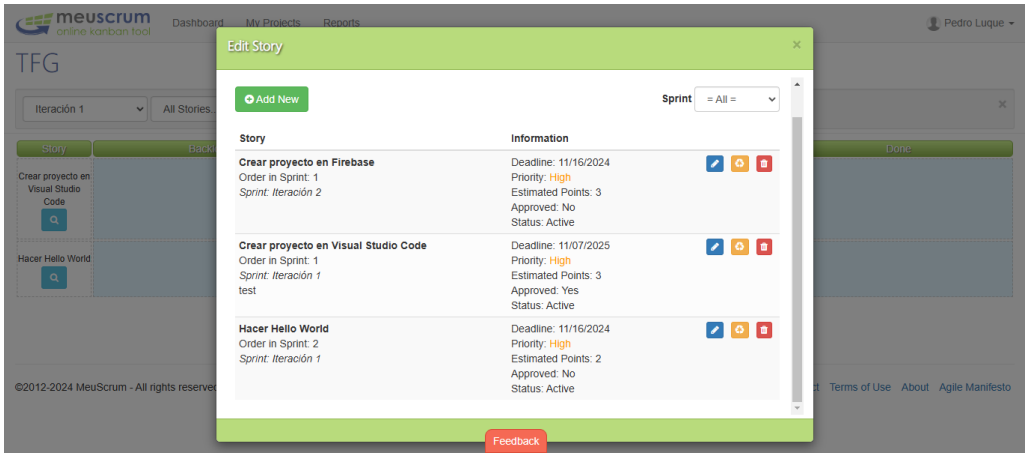


Figura 20. Proyecto en MeuScrum

Url: https://www.meuscrum.com/

4.4.5. GitHub

Fundamental para la gestión y mantenimiento del código fuente, GitHub es a día de hoy la plataforma más usada para alojar código fuente, añadiendo además potentes capas de personalización cómo la creación de proyectos online o páginas web.

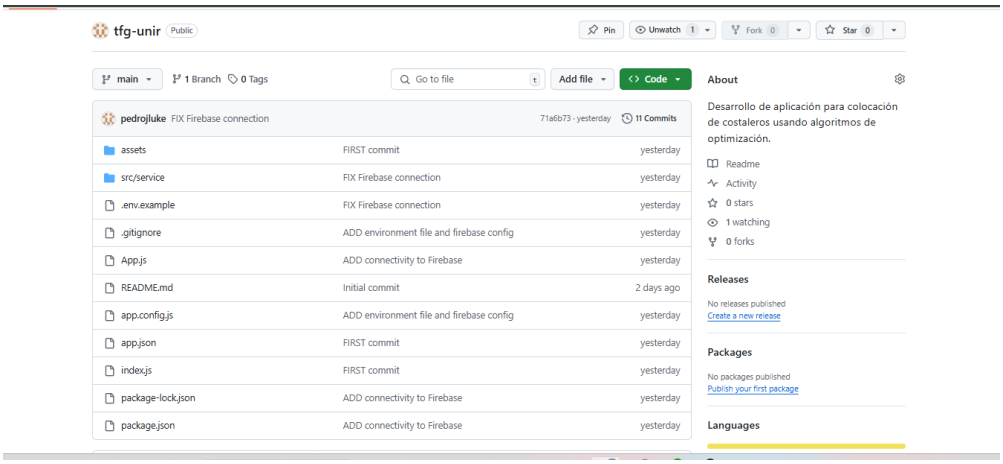


Figura 21. Página principal del proyecto en GitHub

Url: https://github.com/pedrojluke/tfg-unir

5. Diseño y especificación de requisitos

En esta sección se enumera el proceso de transformación de una idea hasta que esta se plasma en elementos tangibles y definidos que permiten su desarrollo. Comienza por la parte de *discovery* o descubrimiento de producto para saber qué es lo que se requiere, para a partir de aquí tener una visión global del producto, desglosándose en historias de usuario, casos de uso y los requisitos funcionales y no funcionales.

5.1. Resultado de la fase de Discovery

Cómo resultado de la fase de Discovery se ha obtenido una visión global y genérica de lo que el cliente espera, identificando las áreas principales del producto a desarrollar. A esta tarea se le llama *User Story Mapping*. Promovida por Jeff Patton, esta metodología permite a los equipos visualizar el flujo de trabajo del usuario a través de un mapa, aportando claridad al proceso de desarrollo, organizando el backlog de forma visual, lo que ayuda a la posterior creación de historias de usuario. (*Qué es el User Story Mapping| Blog UE*, 2023)

Podemos observar en amarillo las áreas sobre las que va a pivotar nuestro producto, y en verde las potenciales historias de usuario ordenadas por prioridad de forma descendente. Este paso no es definitivo ni significa que todo lo que hay descrito en el mapa se vaya a desarrollar ya que este trabajo está acotado por una fecha fija de entrega y por tanto va a depender de la capacidad de desarrollo y sobre todo del tamaño y complejidad de las historias de usuario. Cómo la idea es hacer un MMP algunas funcionalidades pueden no ser requeridas para la primera versión del producto y ser añadidas en futuras versiones.

Por otro lado, hay algunas potenciales historias de usuario descritas que son bastante complejas y que por tanto una vez se refinen, pueden descomponerse en varias historias de usuario, o viceversa; algunas características son tan simples que pueden ser combinadas en una única historia de usuario.

Para salir de dudas, vamos a definir las historias de usuario una a una y estimarlas, para luego ver la capacidad total y priorizar en consecuencia. Y sobre todo, ver hasta dónde se puede llegar.

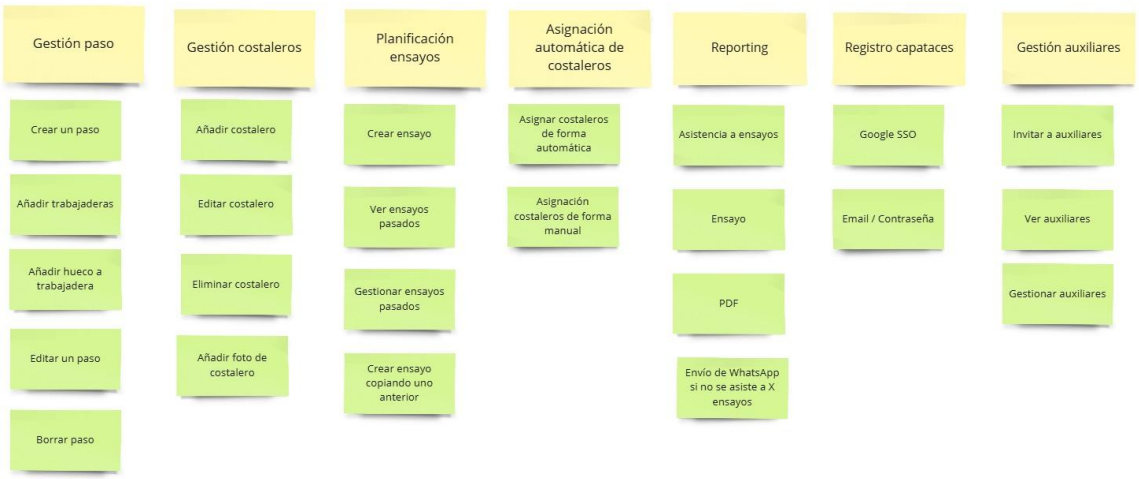


Figura 22. Resultado del User Story Mapping.

5.2. Definición de actores

Actor	Requerimientos
Capataz	• **Crear y gestionar pasos.** • **Añadir y borrar costaleros.** • **Planificar ensayos** • **Asignar costaleros de forma manual** • **Gestionar ayudantes** • **Consultar informes**
Capataz auxiliar	• **Consultar costaleros** • **Consultar ensayos** • **Asignar costaleros**
Sistema	• **Asignar costaleros de forma automática**
Módulo de reporting	• **Generar informes**

Tabla 9. Definición de actores

5.3. Creación de historias de usuario

5.3.1. Gestión de pasos

Número	Nombre	Puntos	Prioridad
US1	Crear paso	5	Alta
US2	Añadir trabajadera	5	Alta
US3	Editar trabajadera	3	Alta
US4	Editar paso	3	Alta
US5	Borrar paso	2	Media

Tabla 10. Listado de Historias de Usuario relativas a la gestión de pasos

5.3.2. Gestión costaleros

Número	Nombre	Puntos	Prioridad
US6	Añadir costalero	5	Alta
US7	Editar costalero	5	Alta
US8	Borrar costalero	3	Media
US9	Añadir foto de costalero	5	Baja

Tabla 11. Listado de Historias de Usuario relativas a gestión de costaleros

5.3.3. Planificación de ensayos

Número	Nombre	Puntos	Prioridad
US10	Crear ensayo	5	Alta
US11	Consultar histórico de ensayos	3	Alta
US12	Copiar ensayo	5	Baja

Tabla 12. Listado de Historias de Usuario relativas a gestión de ensayos

5.3.4. Asignación automática de costaleros

Número	Nombre	Puntos	Prioridad
US13	Investigar sobre algoritmos de ordenación con poda y coste mínimo	5	Alta
US14	Asignación automática de costaleros	5	Alta
US15	Asignación manual de costaleros	5	Alta

Tabla 13. Listado de Historias de Usuario relativas a la asignación de costaleros

5.3.5. Informes

Número	Nombre	Puntos	Prioridad
US16	Asistencia a ensayos	3	Media
US17	Ensayo	3	Media
US18	Exportación a PDF	5	Baja
US19	Integración con WhatsApp para notificar asistencias	8	Baja

Tabla 14. Listado de Historias de Usuario de reporting.

5.3.6. Registro capataces

Número	Nombre	Puntos	Prioridad
US20	Registro usando Google SSO	5	Alta
US21	Registro usando email y contraseña	3	Media
US22	Registro usando otros SSO cómo Facebook o Apple	5	Baja

Tabla 15. Listado de Historias de usuario referentes al registro en la aplicación.

5.3.7. Gestión capataces ayudantes o auxiliares.

Número	Nombre	Puntos	Prioridad
US23	Enviar invitación a auxiliares	8	Media
US24	Ver capataces auxiliares	5	Media
US25	Gestionar capataces auxiliares	3	Baja

Tabla 16. Listado de historias de usuario para la gestión de capataces auxiliares.

5.4. Especificación de requisitos

En esta sección se van a enumerar los diferentes requisitos funcionales y no funcionales del sistema o sus componentes. Un requisito funcional (RF) definen las funcionalidades y características que un software objetivo debe ofrecer. Un requisito no funcional (RNF) aborda características de funcionamiento que pueden ser tan críticos cómo la seguridad, rendimiento o usabilidad, calidad del software, tiempo de respuesta, etc. (webmaster, 2023).

5.4.1. Requisitos funcionales

La siguiente tabla enumera los diferentes requisitos del sistema que han sido extraídos de los resultados de las diferentes fases anteriores. Esta tabla que se muestra incluye la mayoría de las historias de usuario que salieron como parte del proceso de *discovery*, si bien otras han sido combinadas ya que funcionalmente tienen el mismo alcance y pueden ser desarrolladas bajo el mismo caso de uso.

Resumen requisitos funcionales del sistema		
Tabla 24	RF01	Creación de paso de Semana Santa
Tabla 25	RF02	Añadir trabajaderas y huecos al paso
Tabla 26	RF03	Editar / Borrar paso

Tabla 27	**RF04**	**Añadir costaleros**
Tabla 28	**RF05**	**Editar / Borrar costalero**
Tabla 29	**RF06**	**Añadir foto de costalero**
Tabla 30	**RF07**	**Crear ensayo**
Tabla 31	**RF08**	**Ver histórico de ensayos**
Tabla 32	**RF09**	**Gestionar ensayos pasados**
Tabla 33	**RF10**	**Crear ensayo copiando uno anterior**
Tabla 34	**RF11**	**Asignación automática de costaleros a sus respectivas trabajaderas**
Tabla 35	**RF12**	**Asignación de costaleros de forma manual**
Tabla 36	**RF13**	**Generación de plantilla de costaleros**
Tabla 37	**RF14**	**Generación de informe de asistencia**
Tabla 38	**RF15**	**Exportación de informes a PDF**
Tabla 39	**RF16**	**Notificación vía WhatsApp en caso de no asistencia**
Tabla 40	**RF17**	**Registro de usuario en el sistema por Google SSO**
Tabla 41	**RF18**	**Registro de usuario en el sistema por usuario o email**
Tabla 42	**RF19**	**Registro de usuario por otros métodos de SSO**
Tabla 43	**RF20**	**Enviar invitación a ayudantes**
Tabla 44	**RF21**	**Gestionar ayudantes**

Tabla 17. Resumen de requisitos funcionales

5.4.3. Requisitos no funcionales

Estos requisitos funcionales muestran las características deseadas del sistema, poniendo el foco en la calidad y el rendimiento.

Resumen de requisitos no funcionales del sistema		
RNF01	**Web**	**La aplicación debe permitir al usuario usarla desde cualquier lugar**
RNF02	**Disponibilidad**	**El sistema debe de estar disponible en todo momento.**
RNF03	**Portabilidad**	**La aplicación debe poder usarse desde cualquier dispositivo.**
RNF04	**Usabilidad**	**La interfaz debe ser intuitiva, permitiendo una buena experiencia de usuario.**
RNF05	**Rendimiento**	**La aplicación debe manejar un buen número de peticiones sin perjuicio del rendimiento.**
RNF06	**Rendimiento**	**El algoritmo de asignación automática de costaleros se ejecuta correctamente y en un tiempo razonable.**
RNF07	**Flexibilidad**	**La asignación automática de costaleros puede ser revertida para ajustar manualmente los costaleros que se deseen**
RNF08	**Seguridad**	**La autenticación de usuarios en Google debe realizarse con la API oficial de Google, usando la última versión de OAuth.**
RNF10	**Protección de datos**	**Los datos que se guarden deben de cumplir todos los estándares de protección de datos de nuestro ordenamiento jurídico.**
RNF11	**Mantenibilidad**	**La aplicación debe ser fácilmente mantenible y extensible para añadir nuevas funcionalidades o corregir aquello que se pueda mejorar.**
RNF12	**Calidad**	**El código desarrollado cumple con los estándares de buenas prácticas de desarrollo software.**

Tabla 18. Requisitos no funcionales del sistema.

5.5. Diagrama de caso de uso

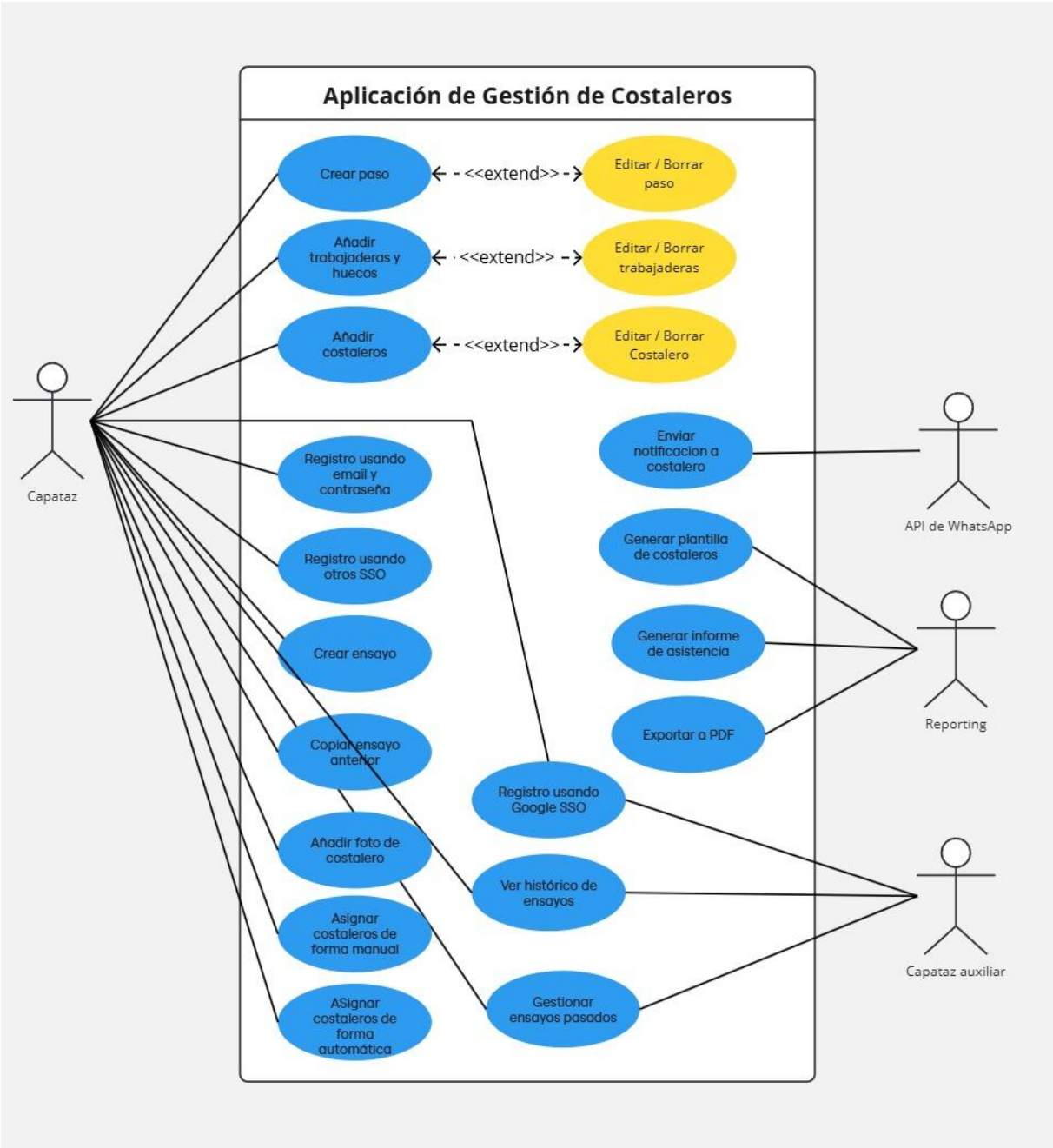


Figura 23. Diagrama de caso de uso

El presente diagrama representa la aplicación que permite a un capataz gestionar los costaleros de un paso de Semana Santa, pudiendo además gestionar los ensayos, la asignación de costaleros en la canastilla de forma manual o automática. También se incluye el reporting

o la interacción con la API de WhatsApp para enviar recordatorios. Igualmente entra la figura del capataz auxiliar para delegar ciertos menesteres relativos a la gestión de costaleros.

Con respecto a las relaciones entre actores y las funcionalidades, el capataz interactúa con casi todos los casos de uso, siendo el principal usuario de la aplicación. El capataz auxiliar, como se menciona en el párrafo anterior tiene acceso a algunas funcionalidades pero no a la totalidad de estas.

5.6. Planificación y priorización por sprints

5.6.1. Sprints 1,2 y 3

Estas tres primeras semanas se dedican casi en exclusiva a la toma de requisitos. Se clarifica qué es lo que de desea y se hace una búsqueda exhaustiva en el mercado por si hubiera alguna herramienta tecnológica que diera solución a la problemática planteada. Esta toma de requisitos ocupa los dos primeros sprints.

El sprint tercero se dedica en exclusividad a comenzar la documentación del proyecto, coincidiendo con la primera entrega de este.

5.6.2. Sprints 4 y 5

Durante estos dos sprints, se configura el proyecto en Visual Studio Code, y se finalizan las primeras historias de usuario:

- Crear Paso
- Añadir y editar trabajadera
- Editar y borrar paso.
- Añadir y Editar Costaleros

5.6.3. Sprints 6,7 y 8

En estos sprints, muy cerca de la segunda entrega de este proyecto, se prosigue desarrollando la aplicación, siendo el turno en este periodo para todo lo referente a los ensayos: creación, edición o borrado.

Además, se comienza a investigar sobre algoritmos que solucionen el problema planteado, exponiendo pros y contras de cada uno antes de tomar una decisión.

5.6.4. Sprints 9 y 10

Coincidiendo con la segunda entrega del trabajo, estos sprints se dedican en exclusiva a la continuación de la memoria del TFG, en este caso poniendo el foco en una definición acotada del contexto de nuestro problema, y enumerando los diferentes algoritmos que pueden distribuir costaleros de forma automática.

Se modela de forma matemática la situación actual de colocación de costaleros, tras un breve ejemplo de cómo es el proceso de forma manual.

5.6.5. Sprints 11, 12 y 13

Sprints cruciales estos, pues en ellos se implementan las funciones fundamentales de este TFG, la asignación manual de costaleros y por supuesto se implementa el algoritmo de Kuhn-Munkres para la asignación automática de los mismos.

5.6.6. Sprints 14 y 15

En estas dos últimas semanas antes de la entrega del borrador final del trabajo, se mejoran aquellos aspectos del mismo susceptibles de ser mejorados, así como se termina el documento de memoria introduciendo los aspectos más técnicos del producto, como es el diseño arquitectónico, el modelo de datos o las tecnologías usadas, así cómo finalizando con las conclusiones finales, líneas de mejora y una valoración personal del proyecto.

5.6.7. Sprints 16, 17 y 18

Estos sprints se han dejado como sprints de contingencia por si hubiera que hacer algún tipo de cambio tras la entrega del borrador final. Durante estas semanas se actualiza el documento con imágenes reales de la aplicación en funcionamiento, y se hacen pequeñas mejoras a dicha memoria.

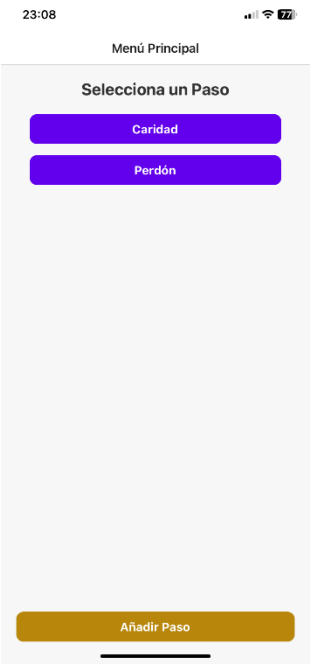
Con respecto al código fuente, se implementa algún tipo de refactorización y se crea el archivo .**md** antes de publicarlo en gitHub.

5.7. Prototipado de componentes

A continuación se detallan los componentes que dan forma a esta aplicación. Se ha optado por un diseño minimalista dando protagonismo al color morado, pues es el color cofrade por excelencia, ya que este representa la penitencia y la preparación espiritual, y está plenamente

aceptado cómo uno de los colores representativos de la Semana Santa. Complementando a dicho color púrpura existen también de forma excepcional algunos elementos en azul claro y también se ha fomentado el uso de iconos que aporten contexto de forma visual sin necesidad de saturar demasiado la visión añadiendo texto. (Aragón, 2017)

5.7.1. Menú Principal



Pantalla principal de la aplicación. En ella se cargan desde la base de datos los pasos guardados con anterioridad y además permite, pulsando el botón situado en el fondo de la pantalla, crear un paso nuevo. Si se pulsa en cualquiera de los pasos ya creados, se dirigirá al usuario a una nueva ventana de detalle del paso dónde podrá dar de alta costaleros, crear ensayos o asignar costaleros a trabajaderas.

Se puede observar cómo se han elegido colores asociados a la Semana Santa cómo el purpura y el dorado.

Figura 24. Menú Principal de la aplicación

5.7.2. Añadir Paso

En esta pantalla se cumplimentarán los datos relativos a un paso de Semana Santa cómo el nombre de este y una descripción. Además, se han de añadir trabajaderas sin límite, especificando, el orden en el trono y el número de huecos respectivamente. Además se le da al usuario la posibilidad de borrar trabajaderas o editar los detalles previamente insertados.

5.7.3. Detalle de Paso

Menú principal del paso, en él se le da al usuario la posibilidad de editar los detalles de este, añadir costaleros, ver los costaleros que pertenecen al paso incluyendo una búsqueda por nombre, así como la posibilidad de acceder a los ensayos. Ventana de carácter simple,

consistente en cuatro botones de color púrpura, consistente con el diseño del resto de componentes.

5.7.4. Añadir costalero

Ventana bastante simple donde se requieren una serie de campos de carácter obligatorio como el nombre, apellidos, teléfono y la altura especificada en centímetros. Todos los campos son de tipo texto. Al pulsar en guardar, se consolida el costalero en la base de datos, asociado al paso creado con anterioridad.

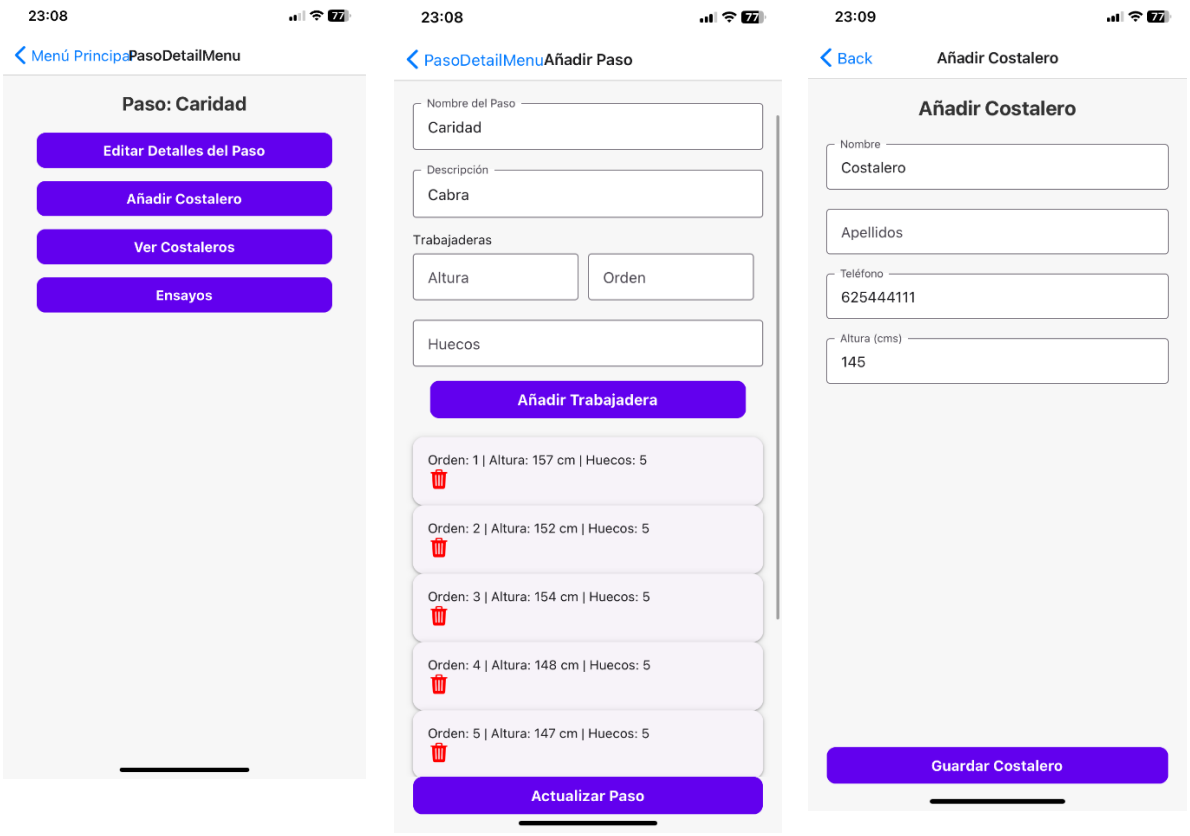


Figura 25. Prototipo de menú de paso, creación de paso y de creación de costalero

5.7.5. Búsqueda de costalero

Búsqueda de costaleros asignados al paso actual, se traen desde la base de datos todos los registros de costaleros pertenecientes al paso y se ordenan por altura para una mejor facilidad de búsqueda. En la lista de resultados, a su vez se muestra el teléfono del costalero, su altura, además de su nombre y apellidos. Se incluye también un icono de una persona para una mayor usabilidad. Además, desde esta ventana de búsqueda también se permite crear un nuevo

costalero, algo útil por si al hacer una búsqueda el costalero deseado no aparece en los resultados.

5.7.6. Menú de ensayos

Menú principal referente a los ensayos, dónde se permite la creación de nuevos ensayos, visualización de ensayos pasados o edición de ensayos ya creados.

Los ensayos aparecen en una lista debajo del botón de creación de nuevos ensayos ordenados por fecha de celebración. Si el ensayo es un ensayo pasado, no se permite asignar costaleros al no tener utilidad. Para una mejor experiencia de usuario, los ensayos pasados aparecerán en rojo, alertando al usuario de que es un ensayo que ya ha sido celebrado.

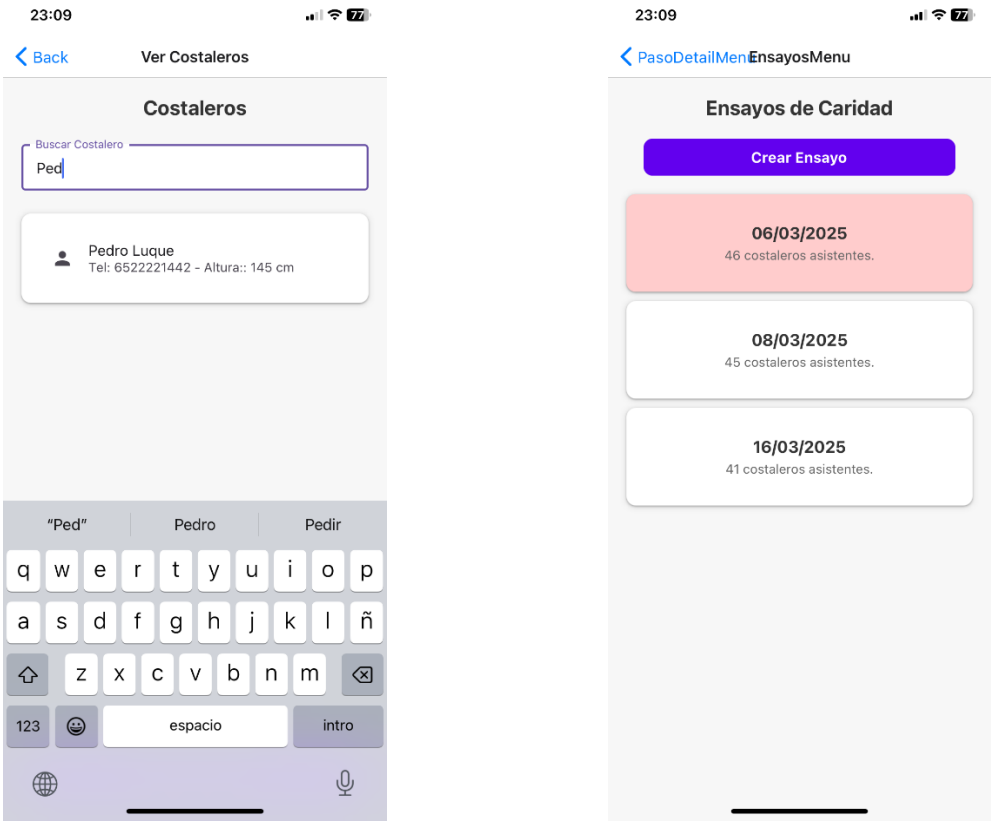


Figura 26. Prototipo de búsqueda de costaleros y de creación de ensayos

5.7.7. Creación de ensayo

En esta pantalla se planifica un ensayo de costaleros. Para ello, se requiere una fecha de celebración que a su vez dará título al ensayo. Tras elegir la fecha de celebración hay que verificar qué costaleros van a asistir al mismo. Para ello, se muestran todos los costaleros

pertenecientes a la cuadrilla y mediante casillas de verificación se seleccionan aquellos que asisten. Debido a que la mecánica de este proceso puede ser laboriosa, se facilita la labor añadiendo una opción de seleccionar todos, siendo más fácil deseleccionar que seleccionar, pues estadísticamente son menos loso que faltan que los que asisten. Debajo de la lista de costaleros, se coloca un contador de costaleros para en todo momento estar informado de cuántos costaleros pueden ensayar en la fecha planificada. Por último, dos botones. En azul claro el botón de asignación de costaleros, mientras que en púrpura, el botón de guardado del ensayo.

5.7.8. Asignación de costaleros

La ventana más compleja de la aplicación. En color púrpura se muestran las diferentes trabajaderas del paso ordenadas según su situación en el paso. Estos botones permiten asignar costaleros a cada trabajadera de forma manual, ya que uno de los requisitos indispensables de este proyecto es que se puedan asignar costaleros de forma automática pero también manual, debido a que hay algunos roles dentro del paso que tienen puestos fijos cómo por ejemplo los pateros. La mecánica es sencilla: elegir los puestos fijos y luego asignar de forma automática el resto.

Al expandir una trabajadera, se verá cómo los huecos de la trabajera están dispuestos de forma vertical, siendo la parte superior la correspondiente al costero derecho y la inferior la del izquierdo. Si no hay costaleros asignados aún, se muestra un botón de asignar que selecciona todos los costaleros candidatos a ese hueco: se excluyen aquellos costaleros cuya altura supere a la de la trabajadera y además, aquellos cuya altura sea diez centímetros inferiores a la de la trabajadera, pues se da por supuesto que siempre habrá un candidato mejor estos casos, facilitando así la labor del capataz.

Asimismo, una vez que un costalero se asigna a un hueco, aparece un botón de borrar por si se buscase otra posición para dicho costalero. Una vez la trabajadera esté completamente asignada, aparecerá a la derecha un tick de validación, para informar de que una trabajadera ha sido validada con éxito.

En la parte inferior de la pantalla aparecen dos botones: Asignar de forma automática, el cual asigna de forma automática los costaleros **no asignados de forma manual** al resto de huecos,

y otro botón para ver aquellos costaleros que no han sido asignados y que parten como aspirantes en el ensayo.

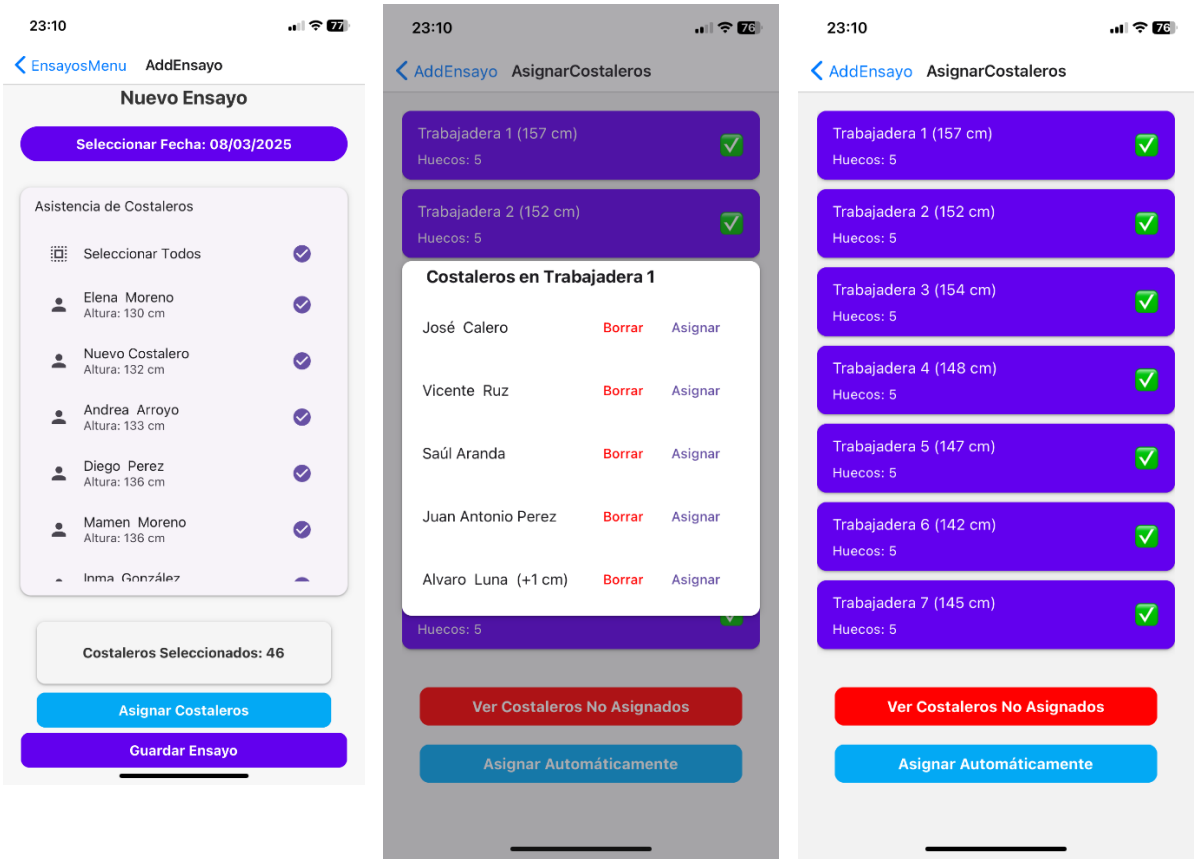


Figura 27. Asignación de costaleros

6. Desarrollo e implementación

Una vez finalizada la exposición del diseño y la especificación de requisitos, se pasa a detallar el proceso de desarrollo e implementación, empezando por la crucial elección de la tecnología utilizada, pasando por la arquitectura ideada, y finalizando por el diseño de componentes.

6.1. Tecnologías disponibles

Como se ha hablado en secciones precedentes, se pretende crear una aplicación multiplataforma y multidispositivo. Para ello, primeramente se va a justificar esta decisión, fundamentada principalmente en el ahorro de tiempo de desarrollo en comparación con las aplicaciones nativas. Además, la creación de una aplicación multiplataforma repercute en una menor mantenibilidad del código fuente creado, evitando código duplicado y por tanto, en aras de una mayor sostenibilidad a largo plazo se opta por crear la aplicación multiplataforma. Bien es cierto que el desarrollo de una aplicación multiplataforma tiene ciertas desventajas como el rendimiento, ya que una aplicación nativa siempre va a aprovechar mejor los recursos del dispositivo que una aplicación realizada en una herramienta de terceros dependiendo de capas intermedias para renderizar la aplicación en el dispositivo. (Stanislav Naborshchikov, 2023).

Pese a estas limitaciones, las ventajas superan ampliamente estas posibles desventajas, sobre todo cuando hoy en día existen herramientas en el mercado muy eficientes y solventes como React Native, y con una gran comunidad detrás.

En las siguientes líneas se van a describir algunas tecnologías multiplataforma que se han considerado relevantes para la implementación de este proyecto.

6.1.1. Flutter

Flutter es un SDK desarrollado por Google pare crear aplicaciones móviles tanto para Android como para IOS. Este fue desarrollado como un software cómo uso interno dentro de la compañía pero al ver el potencial, se liberó su código en el año 2017, siendo desde entonces *open source.* El lenguaje de programación usado en Flutter es lenguaje Dart, que requiere una curva más grande que otros lenguajes. Tiene su propio motor gráfico por lo que su UI es la más cercana a la nativa del dispositivo.

6.1.2. Apache Cordova

Cordova es un framework desarrollado por Apache que permite desarrollar aplicaciones mediante *WebViews*, usando componentes web como html, css y JavaScript. La aplicación desarrollada en estas tecnologías web se ejecuta en un contenedor nativo que es envuelto en la citada *WebView*. Su rendimiento es bastante inferior al de las soluciones nativas, pero en contrapartida hace muy fácil la conversión de aplicación web a aplicación móvil.

6.1.3. React Native

React Native, según se extrae de su página web, es un *framework* de código abierto creado por Facebook (Meta) para desarrollar aplicaciones para Android, iOS, MacOS, Android TV, Web y Windows. La diferencia con respecto a React radica en que sustituye los componentes HTML para ser ejecutados en un navegador, por otros que se ejecuten directamente sobre las plataformas móviles nativas. (Álvaro Jiménez Martín, 2019).

El lenguaje principal de esta tecnología es JavaScript, aunque también se pueden desarrollar las aplicaciones usando TypeScript. Una de sus mayores ventajas es la gran comunidad de usuarios que posee y lo más importante, que el rendimiento, pese a no ser nativo, es el que más se acerca a lo nativo.

6.1.4. Ionic

Ionic es un framework híbrido que usa también tecnologías webs cómo html, css y Javascript para construir aplicaciones móviles. Tiene un enfoque basado en webviews cómo Cordova, por lo que no obtiene el rendimiento de una aplicación nativa. Es una buena tecnología para desarrollar aplicaciones de forma rápida debido en gran parte, a su diversidad de plugins disponibles

6.2. Tecnologías elegidas y justificación

En esta sección se describen las tecnologías que son parte fundamental de este proyecto. Se enumeran tanto las tecnologías de frontend, como de backend, así como aquellos frameworks o herramientas necesarias para llevar a cabo este proyecto.

6.2.1. React Native

Figura 28. Logo de React Native

Una vez valorados todos los pros y contras de cada una de las tecnologías para la realización de este TFG se ha optado por **React Native,** debido básicamente a la gran comunidad y soporte disponibles, y a la posibilidad de usar innumerables bibliotecas de terceros mediante paquetes de *Node.js*. Además, es compatible con Android, iOS y además con aplicaciones web usando el mismo código fuente, sin tener que diferenciar código dependiendo de la capa de presentación, por lo que solo habrá un único código fuente. Asimismo, con respecto al rendimiento, de las cuatro tecnologías mencionadas anteriormente es la única que, sin ser nativa, se acerca prácticamente a un rendimiento nativo.

React – Native usa una arquitectura basada en componentes, piedra fundamental de su escalabilidad, promoviendo la reutilización del código al dividir los componentes en otros a su vez más pequeños e independientes, que encapsulan datos y funcionalidad, haciéndolos fáciles de mantener. (Root Stack, s. f.).

Otra ventana de usar React Native radica en la utilización de JavaScript cómo lenguaje, siendo este uno de los más populares hoy. Esto permite que una persona con nociones de JavaScript puede aprender React native rápidamente, reduciendo la curva de aprendizaje.

En cuanto al entorno de desarrollo, React Native usa la llamada *hot reloading*, (recarga en caliente) una característica de este lenguaje que hace que una vez que se guarda el código fuente, este es reflejado en el ejecutable en menos de un segundo, sin tener que desplegar ni compilar sin tener que hacer nada más. (*Introducing Hot Reloading · React Native*, 2016).

Figura 29. Estructura de una pantalla en React-Native. Fuente: https://reactnative.dev

El vasto ecosistema de bibliotecas disponibles es un punto para considerar favorablemente a la hora de usar React, pues un desarrollador tan solo tiene que focalizarse en crear la solución principal, sin inventar la rueda, ya que siempre habrá disponible una biblioteca que integre funcionalidades específicas cómo acceso a sensores de dispositivos, notificaciones, cámara, etc.

React-Native descarga la responsabilidad al usuario final de crear interfaces de usuario que sean compatibles con todos los dispositivos, proporcionando componentes nativos cómo <**View**>, <**Text**>, <**Image**> o <**Button**>, equivaliendo estos a <**div**>, <**p**>, <**img**> o <**button**> en lenguaje html. En cuanto al estilo, React proporciona una hoja de estilos muy similar a css pero usando la sintaxis de JavaScript llamada **StyleSheet**, eliminando cualquier referencia de css en este framework de desarrollo. (*StyleSheet · React Native*, s. f.)

Una de las características fundamentales del lenguaje React-Native son los *Hooks,* que permiten el manejo de estados del sistema, fundamental a la hora de crear y trabajar con componentes. Hay dos tipos de propiedades que controlan un componente:

1. **Prop**. Son definidos por el padre y no varían durante toda la vida útil de un componente.
2. **State**. Los datos que van a cambiar durante el ciclo de vida de un componente tienen que ser definidos en el *state*.

6.2.2. Expo

Figura 30. Logo de Expo

Expo es un ecosistema dentro de React Native que facilita el desarrollo, la compilación y el despliegue de proyectos en React Native. Una de las desventajas del uso de React Native puro es precisamente la complejidad de la configuración de un entorno de desarrollo, ya que es todo manual, sin ningún tipo de apoyo en herramientas de terceros. Expo es compatible con iOs, Android y web, permitiendo, con un mismo código fuente, que una aplicación funcione en tres plataformas totalmente antagónicas. Además, Expo proporciona una opción fácil y rápida para desarrollar en React-Native, limitando el control y la flexibilidad sobre React Native. En este caso, como la aplicación es una aplicación sencilla, sin acceso a sensor

6.2.3. React Native Paper

La creación de interfaces de usuario es uno de los hitos más importantes de este trabajo, pues se requiere una interfaz sencilla, intuitiva y fluida para garantizar una experiencia de usuario óptima. Los componentes nativos de React, al requerir de bastante customización para ajustarse a lo que se requiere en este proyecto, se descartan, por lo que tras un minucioso estudio sobre las bibliotecas de interfaz de usuario disponibles en el mercado, se opta por React Native Paper, basada en Material Design, debido básicamente a su facilidad de implementación y apariencia simple pero moderna.

Material Design es un estilo de diseño creado por Google con el lanzamiento de Android Lollipop. Su éxito derivó en un lenguaje propio de diseño estableciendo cómo ha de ser la interacción persona – dispositivo. El nombre radica en que para la consecución de este estilo de diseño se usaron objetos materiales para estudiar la luz y adecuarlos a las aplicaciones, analizando superficie y movimiento de estos dependiendo de la luz, con la finalidad de crear un diseño basado en el mundo real, para trasladar estos al diseño digital. El propósito que tuvo Google cuando creó este lenguaje no fue otro que unificar el diseño entre web y móviles, para tener la percepción de que todo sea coherente funcional y estéticamente. (Bustos, 2022).

React Native Paper es una librería de componentes de interfaz de usuario específicamente diseñada para React Native, alineada con los principios de Material Design, siendo compatible con modo claro y oscuro de un teléfono móvil, con React Navigation o permitiendo establecer

colores o estilos globales dentro de una aplicación de forma sencilla. Su instalación se realiza mediante paquetes de Node.js, y estos se pueden usar directamente importándolos al proyecto, reduciendo considerablemente el tiempo de desarrollo.

6.2.4. Node.js

Node.js es el entorno que permite ejecutar JavaScript fuera del navegador web, proporcionando las herramientas necesarias para el desarrollo de aplicaciones modernas. Con la instalación de Node.js también se instala npm (gestor de paquetes de Node.js), siendo fundamental en cualquier aplicación, gestionando las librerías que se añaden al proyecto, además de facilitar la configuración de un proyecto cuando se trabaja con repositorios.

En este trabajo, son diversos los paquetes de Node.js que se utilizan, destacándose los siguientes:

- Expo. Mencionando anteriormente, este framework facilita mucho el trabajo con React Native.
- Prettier. Fundamental en desarrollo web, esta herramienta formatea el código para garantizar que éste cumpla con los estándares de diseño.
- React Navigation. Sistema de navegación que usa React.
- Firebase. También explicado anteriormente, sirve de conexión con el backend cloud de Google.
- React Native Paper. Biblioteca de components UI usada en este proyecto, basada en Material Design.

En definitiva, Node, es esencial para el desarrollo de la aplicación permitiendo no solo la instalación y gestión de bibliotecas sino también proporcionando un entorno para el desarrollo y despliegue de aplicaciones de forma eficiente.

6.2.5. Firebase

Firebase es una plataforma en la nube para el desarrollo de aplicaciones web, perteneciente a Google. Su propósito es simplificar las labores de gestión y de mantenimiento de la infraestructura, permitiendo al usuario final centrarse únicamente en el qué y no en el cómo.

La elección de Firebase como motor de backend ha sido fundamentada básicamente en la necesidad de tener una base de datos en tiempo real y que no esté acoplada a un teléfono

físico. También se ha querido desvincular la creación de la base de datos del uso de esta. Es decir, Firebase al ser una base de datos no relacional no requiere de una creación de tablas propiamente dichas, sino que al trabajar con estructuras de datos plenamente conocidas en lenguajes web como son los JSON, no se necesita de ningún tipo de tabla física para guardar los datos.

En la siguiente imagen se puede apreciar cómo una tabla de pasos tiene asignados una serie de costaleros, y además estos costaleros están asociados a una trabajadera dentro de un ensayo.

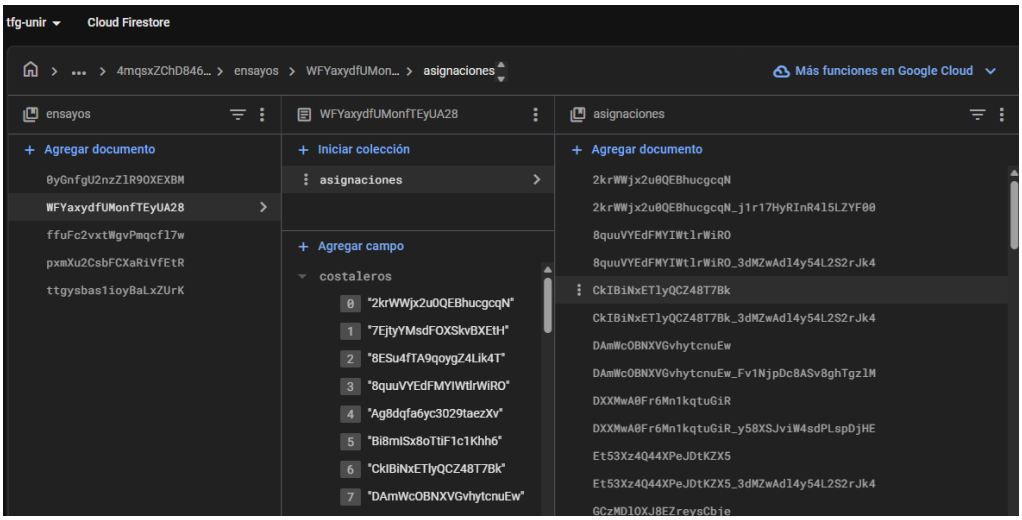


Figura 31. Pantalla principal de Firebase

En lo que respecta a la configuración, Firebase no es excesivamente difícil de configurar, tan solo hay que crear una aplicación en su web, y acto seguido generará un archivo JavaScript para importarlo al proyecto de Visual Studio. Este archivo contiene las API keys de la aplicación, por lo que no es conveniente copiar y pegar, sino enmascarar dichas claves en constantes, y crear un archivo local .env donde se peguen las claves para el uso del proyecto, sin compartir estas por motivos obvios de seguridad. Con respecto a este tema, sin ser una tecnología propiamente dicha, sí es conveniente usar un paquete de npm llamado dotenv que hace uso de este archivo **.env** para el manejo de las claves de seguridad.

```
app.config.js > ...
1   import "dotenv/config";
2
3   export default ({ config }) => ({
4     ...config,
5     extra: {
6       firebase: {
7         apiKey: process.env.FIREBASE_API_KEY,
8         authDomain: process.env.FIREBASE_AUTH_DOMAIN,
9         projectId: process.env.FIREBASE_PROJECT_ID,
10        storageBucket: process.env.FIREBASE_STORAGE_BUCKET,
11        messagingSenderId: process.env.FIREBASE_MESSAGING_SENDER_ID,
12        appId: process.env.FIREBASE_APP_ID,
13        measurementId: process.env.FIREBASE_MEASUREMENT_ID,
14      },
15    },
16  });
17
```

Figura 32. API keys de firebase en Visual Studio Code, ocultadas por seguridad

Por último, como se ha comentado al principio de este punto, Firebase no es una base de datos relacional por lo que las consultas a realizar contra la base de datos son distintas a las que se podrían realizar en cualquier entorno relacional. En esta consulta que se muestra, se están trayendo todos los costaleros ordenados por altura.

```
const fetchCostaleros = async () => {
  try {
    setLoading(true);
    const costalerosRef = collection(db, collections.usuarios.name);
    const q = query(
      costalerosRef,
      where(collections.pasos.id, collections.pasos.equals, pasoId),
      where(
        collections.usuarios.rolField,
        collections.pasos.equals,
        collections.usuarios.rol
      )
    );
    const costalerosSnap = await getDocs(q);
    let costalerosList = costalerosSnap.docs.map((doc) => ({
      id: doc.id,
      ...doc.data(),
    }));
    costalerosList = costalerosList.sort((a, b) => a.altura - b.altura);
    setCostaleros(costalerosList);
  } catch (error) {
    console.error(TEXTS.error, error);
  } finally {
    setLoading(false);
  }
};
```

Figura 33. Consulta de datos a Firebase

6.3. Arquitectura de la aplicación

En esta sección se detalla la arquitectura de este TFG. La herramienta tecnológica se crea para gestionar cuadrillas de costaleros, empezando por la definición del paso, la estructura de sus

trabajaderas (altura y huecos disponibles), asignar costaleros a cada paso creado, y una vez haya costaleros suficientes, crear ensayos dónde los costaleros puedan ser asignados de forma automática o de forma manual por el usuario, también llamado capataz. La información, en ningún caso va a estar almacenada en el dispositivo del usuario sino en la nube.

Para lograr este cometido, se usará una arquitectura llamada *serverless,* un método de modelado de software mediante el cual se pueden crear aplicaciones sin gestionar la arquitectura por sí misma. El desarrollador tan solo tiene que preocuparse de escribir un buen código y ejecutarlo correctamente, despreocupándose del aprovisionamiento, escalabilidad o mantenimiento del servidor. Es decir, las aplicaciones *serverless* se ejecutan en los servidores pero los recursos asignados a la aplicación no dependen de la aplicación sino del proveedor de servicios en la nube. (Google Inc., s. f.)

Para este proyecto se ha elegido la plataforma de Google por su facilidad de uso, su gratuidad por la gran comunidad de desarrollo que tiene a su alrededor, además de ser compatible con React Native + Expo de una forma sencilla, punto a favor de Google en este sentido.

6.3.1. Diseño arquitectónico

Se decide estructurar la arquitectura del sistema en tres capas bien diferenciadas entre sí:

- Capa cliente. Aplicación desarrollada en React Native con Expo, interfaz de usuario creada usando una librería de terceros llamada React Native Paper – Material UI, así como las dependencias de paquetes de node necesarias para que la aplicación funcione. Al estar la aplicación implementada en React Native, es compatible con cualquier plataforma o dispositivo.
- Capa de seguridad. Quizás la más importante de la aplicación, ya que ha de garantizar la integridad y seguridad de los datos. Para ello, se delega en el *single sign-on* de Google, que a su vez está integrado en **Firebase,** permitiendo el acceso o no a los datos almacenados en esta aplicación en la nube. Esto se logra gracias a la autorización basada en token, verificando en todo momento la identidad del usuario ante cualquier interacción con los datos almacenados.
- Capa de *cloud computing*. Configuración de Firebase, Google *Firestore* y Google *FireStorage* para el procesamiento y almacenamiento de los datos en la aplicación.

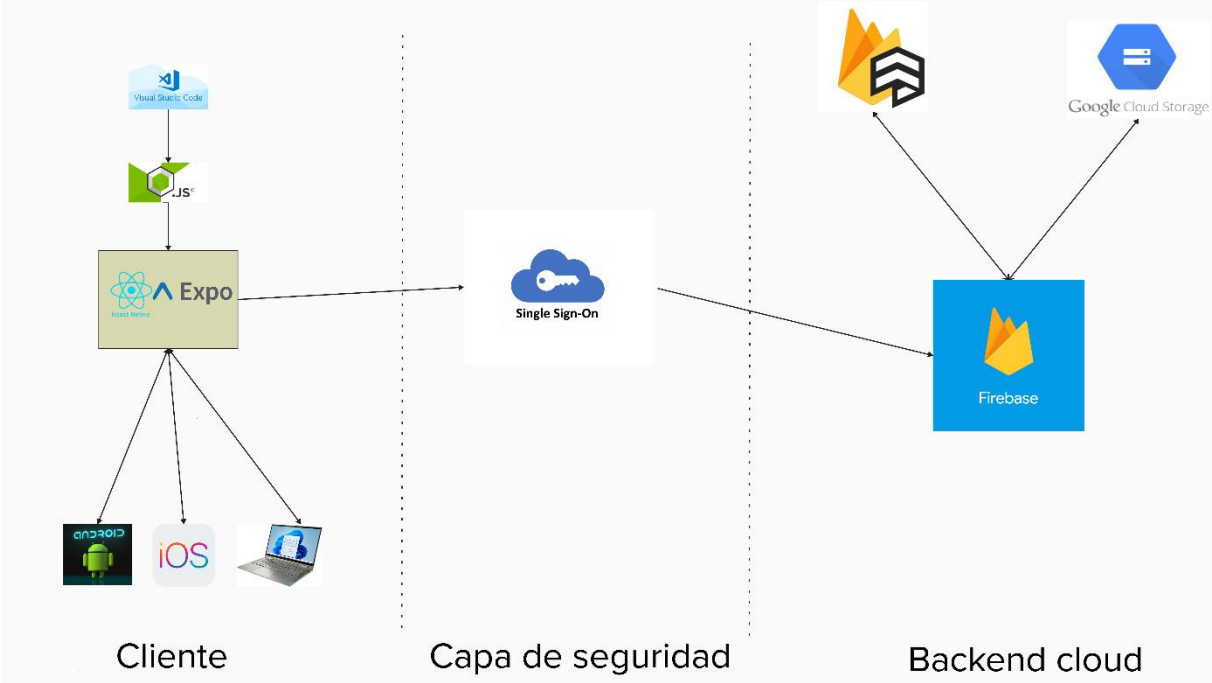


Figura 34. Diseño arquitectónico de la aplicación. Fuente: creación propia

6.3.2. Modelo de datos

Como se menciona en el diseño arquitectónico, se ha elegido Google Firebase como backend. Este backend proporciona una base de datos no relacional llamada Firebase. Firebase, es una base de datos NoSQL, flexible, escalable y en la nube, creada por Google para almacenar y sincronizar datos. Mantiene los datos sincronizados en tiempo real a través de objetos escucha y ofrece soporte sin conexión para dispositivos que no se puedan conectar a la web. Tiene una altísima disponibilidad y escala de forma automática, adaptándose a cualquier casuística o carga de trabajo. (Google, Inc., s. f.)

6.3.2.1. Usuarios

Guarda información de los usuarios que forman parte de la aplicación. Pueden desempeñar tres roles fundamentales: capataz, auxiliar y costalero.		
Firestore: usuarios/*usuarioId*		
usuarioId	**String**	**Id del usuario, asignado automáticamente por firebase**
Nombre	**String**	**Nombre del usuario**

Apellidos	**String**	**Apellido del usuario**
Altura	**Integer**	**Altura del costalero**
Telefono	**String**	**Número de teléfono del costalero**
Rol	**Enum**	**Capataz, Costalero, Auxiliar**
Pasos	**Lista**	**Contiene los identificadores de aquellos pasos a los que pertenece el usuario**

```
{
"usuarioId": "id1",
"nombre": "José Luis",
"apellidos": "Martínez",
"telefono": "666555444",
"rol": "costalero",
"altura": 145,
"pasos": ["paso1", "paso2", "paso3"]
}
```

Tabla 19. Colección Usuarios de Firebase

6.3.2.2. Pasos

Representa un paso procesional, con sus respectivos costaleros.		
Firestore: pasos/*pasoId*		
pasoId	**String**	**Id del paso, asignado automáticamente por firebase**
Nombre	**String**	**Nombre del paso**
CapatazId	**String**	**Id del capataz del paso**
Auxiliares	**Lista**	**Lista de Ids de los capataces auxiliares del paso si los hubiera**
Trabajaderas	**Int**	**Total de trabajaderas en el paso**

Costaleros	Lista	Lista de ids de los costaleros que pertenecen al paso

<pre>{
"pasoId": "id1",
"nombre": "Paso TFG",
"capatazId": "id1",
"trabajaderas": 8,
"costaleros": [“cost1”, “cost2”, “cost3”]
}</pre>		

Tabla 20. Colección pasos de Firebase

6.3.2.3. Ensayos

Subcolección bajo la colección paso para almacenar los ensayos previstos.		
Firestore: pasos/*pasoId*/ensayos/*ensayoId*		
ensayoId	String	Id del ensayo, asignado automáticamente por firebase
Fecha	Date	Fecha de celebración del ensayo
Costaleros	Lista	Lista de ids de los costaleros que asisten al ensayo

<pre>{
"ensayoId": "ensayoId1",
"fecha": "2025/03/08",
"costaleros": [“cost1”, “cost2”, “cost3”]
}</pre>		

Tabla 21. Colección Ensayos de Firebase

6.3.2.4. Trabajaderas

Subcolección bajo la colección paso para almacenar las trabajaderas que determinan la posición de los costaleros.		
Firestore: pasos/*pasoId*/trabajaderas/*trabajaderaId*		
trabajaderaId	String	Id de la trabajadera, asignado automáticamente por firebase

Orden	**Int**	**Orden de la trabajadera desde delante hacia atrás**
Altura	**Int**	**Altura de la trabajadera**
Costaleros	**Lista**	**Lista de costaleros asignados a la trabajadera**
{ "trabajaderaId": "trabajaderaId1", "orden": 1, "altura": 145, "costaleros": [“cost1”, “cost2”, “cost3”] }		

Tabla 22. Colección Trabajaderas de Firebase

6.3.2.5. Asignaciones

Subcolección bajo la colección paso para almacenar la asignación a una trabajadera en un ensayo determinado.		
Firestore: pasos/*pasoId*/ensayos/ensayoId/asignaciones/*asignacionId*		
asignacionId	**String**	**Id de la asignación, asignado automáticamente por firebase**
EnsayoId	**String**	**Id del ensayo al que pertenece esta asignación**
CostaleroId	**Int**	**Id del costalero asignado**
TrabajaderaId	**Int**	**Id de la trabajadera asignada al costalero**
Taco	**Int**	**Diferencia de altura entre la trabajadera y el costalero**
{ "asignacionId": "asign01", "ensayoId": "ens1", "costaleroId": "cost1", "trabajaderaId": "tr1": "taco": 0, }		

Tabla 23. Colección asignaciones de Firebase

6.4. Implementación del algoritmo de distribución

Figura 35. Resultado de distribución de costaleros automática

Una vez implementado el algoritmo de distribución, se ha comparado contra una distribución real de costaleros (anexo B), cedida por el equipo de capataces del palio de la Virgen de la Caridad de Cabra (Córdoba), y cómo se puede apreciar, en la distribución manual se obtiene un coste total de 40 centímetros, mientras que en la distribución automática el coste se reduce a 37 pero porque algunas posiciones son fijas por mandato de dicho equipo de capataces: pateros o gente que puede actuar de pateros van asignados de forma fija a la primera o última trabajadera y el algoritmo no tiene posibilidad de mejorar dichas posiciones.

La gran ventaja de esta distribución es que se hace en menos de un segundo, mientras que la manual requiere horas o días para encontrar la combinación correcta, permutando costaleros hasta dar con una disposición que satisfaga al equipo de capataces.

7. Conclusiones y trabajo futuro

En esta última sección del documento se muestran las conclusiones obtenidas tras la finalización de este, y aquello que es susceptible de ser mejorado o extendido en versiones futuras. Se procederá a evaluar el cumplimiento o no de aquellos objetivos planteados al principio del proyecto. Asimismo, se justificará con evidencias reales la consecución o no de los objetivos específicos inherentes al objetivo general de este trabajo: crear una aplicación multiplataforma que permita gestionar cuadrillas de costaleros de una forma eficiente y óptima.

7.1. Conclusiones del trabajo

La ejecución de este TFG ha demostrado ser un avance significativo en la modernización de un ámbito sumamente tradicional e inmovilista como la Semana Santa. El objetivo principal de este proyecto ha sido crear una herramienta para darle soporte a los capataces de pasos de Semana Santa a la hora de distribuir los costaleros dentro de un paso procesional.

El resultado de las pruebas realizadas refrenda que este objetivo se ha cumplido holgadamente, pues en el ejemplo real adjuntado como parte de este trabajo, se ha demostrado que con la distribución automática realizada implementando el método de asignación húngara o de Kuhn – Munkres, se obtienen mejores resultados comparados con el caso real del año 2024, mejorando en 8 centímetros la asignación realizada de forma manual por el equipo de capataces. Para lograr el objetivo principal del proyecto, implícitamente se han tenido que cumplir los otros objetivos generales del mismo, pues son pasos necesarios para lograr la asignación óptima, a saber:

1. Creación de pasos procesionales.
2. Configuración de las trabajaderas, indicando su altura y los huecos disponibles.
3. Mantenimiento de costaleros: Alta, Baja o modificación de datos.

El proyecto comenzó con la definición del **qué,** importante a la hora de conocer qué se quería realizar, conociendo los problemas que estaban teniendo los capataces de diversas cofradías a la hora de gestionar una cuadrilla de costaleros, y sobre todo, de asignarlos de forma óptima en el paso procesional, pues no existía certeza de si la solución elegida era la mejor o no.

El segundo objetivo específico consistió en un minucioso análisis del estado del arte del sector, valorando pros y contras de las aplicaciones existentes, la mayoría focalizadas en cofradías en general, pero sin referirse en ningún momento a la gestión de una cuadrilla de costaleros. Tan solo se encontró una aplicación que ponía énfasis en el mundo del costal, pero era más una red social que una herramienta para capataces. Básicamente, esta aplicación ponía en contacto a capataces y a costaleros, sin organizar la cuadrilla internamente. Por tanto, el objetivo se concluyó resaltando la necesidad de crear una aplicación que aprovechase esa oportunidad sectorial para crear algo único.

Una vez que se decidió realizar una aplicación nueva a no existir nada en el mercado que cumpliera con los requisitos, se propuso realizar una aplicación móvil para que el capataz pudiera llevarla consigo, y hacer y deshacer cambios en la cuadrilla de forma rápida y sencilla, sin necesidad de seguir los procedimientos tradicionales, más mecánicos y costosos pero menos efectivos. Se propone que los datos de esta aplicación no residan en el teléfono del usuario sino en la nube para desacoplar dichos datos del dispositivo.

Antes de proceder a la implementación del producto, se diseñaron una serie de prototipos de los componentes para su validación previa al desarrollo. En base a estos prototipos, se planifica el trabajo por sprints de una semana de duración, siguiendo un desarrollo escrupulosamente ágil, con entregas cortas y sencillas para ir recibiendo feedback lo más pronto posible.

Otro objetivo específico fue la verificación del producto. En este caso, como se ha mencionado anteriormente, se comparó la asignación de costaleros hecha para la Semana Santa del año 2024, con la realizada automáticamente por la aplicación, resultando 3cm mejor la asignación automática que la manual.

Como conclusión final, en base a todo lo mencionado durante este punto y a lo expuesto en este TFG se concluye que este trabajo ha cumplido con éxito aquello que se propuso cómo objetivo general, así como aquellos objetivos específicos descritos líneas arriba, aportando una mejora considerable a la hora de asignar costaleros a un paso de Semana Santa, pero también dando una mayor versatilidad a los capataces ante posibles imprevistos que puedan ocurrir durante una procesión o ensayo, y que con la ayuda de la aplicación ya no serán tales, pues se proveerá una solución inmediata a dichos imprevistos.

7.2. Líneas de trabajo futuro

Pese a que la mayoría de los objetivos inicialmente planteados han sido cumplidos de forma exitosa, cómo se justifica en el estado del arte, en esta área de conocimiento sigue habiendo aún mucho trabajo por hacer al ser un entorno generalmente poco proclive a la modernización tecnológica. Esto permite que se allane el camino para la proliferación de nuevas soluciones que automaticen algunas tareas cómo la gestión de una cuadrilla de costaleros

Hay que discernir aquí entre mejoras del proyecto actual y las líneas de trabajo futuro que evolucionarían el producto.

7.2.1. Líneas de trabajo futuro

- Duplicación de pasos. Sería de gran utilidad añadir una nueva funcionalidad que permita a los usuarios reutilizar pasos existentes. Es una casuística poco dada, pero puede haber capataces que gestionen más de un paso de Semana Santa y que por tanto el proceso de crear un trono desde cero sea un tanto repetitivo, ayudando esta funcionalidad de copiar un paso y cambiar solo el número de altura de las trabajaderas.
- Reutilización de costaleros. Permitir que un costalero esté asignado a más de un paso sin tener que registrarlo una vez en cada paso que procesione. Esto impactaría directamente en la consistencia de los datos.
- Añadir nuevas variables de asignación. Añadir restricciones al algoritmo utilizado para tener en cuenta cómo el rendimiento histórico en un determinado puesto, la disponibilidad de ensayos o si su perfil se adapta más a la delantera o a la trasera del paso.

7.2.2. Mejoras

- Posibilidad de que el algoritmo automáticamente determine la altura de las trabajaderas, ya que en este trabajo se está asumiendo que la altura de éstas es fija, pero también podrían ser variable. Dado que la viga tiene unos centímetros de movilidad dentro del trono, este ajuste puede permitir mejorar la adaptabilidad de la altura de los costaleros, repercutiendo en una mejor distribución reduciendo la necesidad de tacos compensando la diferencia de altura.

- Soporte para SSO adicional. Ampliar los proveedores de inicio de sesión, permitiendo otros proveedores populares como Facebook o Apple. Esto podría beneficiar en una mayor apertura a los usuarios, no forzándolos al uso de un único proveedor de SSO.
- Integración con WhatsApp. Mencionado en la fase de discovery pero no priorizado por falta de tiempo, esta integración permitiría enviar recordatorios a los costaleros mediante WhatsApp.
- Uso de Redux para el manejo de los estados del sistema. El uso de esta librería permitiría desacoplar la lógica del manejo de estados de los componentes, proporcionando una solución más eficiente y modular.

7.3. Valoración personal

Este trabajo finaliza una etapa académica. Ha supuesto un desafío, que en determinados momentos ha requerido de un gran esfuerzo e implicación. La temática elegida ha sido un motor de motivación en los momentos más difíciles haciendo que el compromiso con el trabajo permaneciera intacto. La proximidad de la entrega de este proyecto con el inicio de una nueva Cuaresma ha añadido una presión positiva impulsando la finalización de la aplicación a tiempo para su puesta en práctica en la Semana Santa del presente año.

Uno de los aspectos más importantes de este trabajo ha sido la investigación y selección de las tecnologías a aplicar. Pese a que en un principio se planteó la idea de desarrollarlo en una tecnología de sobra conocida para el autor, la decisión de ir finalmente con una tecnología nueva como React Native ha conseguido que se apliquen los conocimientos adquiridos durante el Grado. Desde las entrevistas con el cliente para determinar los requisitos, pasando por el modelado del producto y su diseño arquitectónico, hasta la implementación final, todo ha sido un proceso de aprendizaje enorme, ya que uno de los objetivos de este trabajo era salir de la zona de confort y no utilizar tecnologías con las que hubiera trabajado anteriormente, un reto cumplido con creces.

Dejando de lado el aspecto técnico, este trabajo representa la unión de dos de mis grandes pasiones: la Semana Santa y el desarrollo de Software. La idea primigenia surgió para solucionar un problema recurrente en muchas cuadrillas de costaleros y el saber que con este trabajo se va a contribuir a mejorar el estado del arte, me satisface enormemente. Además, también se cumple el propósito de que este TFG no solo sea un mero ejercicio académico sino

que tenga un impacto real en el mundo cofrade, reforzando el valor de este trabajo realizado durante meses.

Por último, las líneas de trabajo que se han descrito más arriba dejan la puerta abierta a una extensión del producto, esperando que sean tomadas en consideración para que esta herramienta puede seguir creciendo y extendiéndose, ayudando a costaleros y capataces en su día a día.

Referencias bibliográficas

Alexander Menzinsk, Gertrudis López, Juan Palacio, Miguel Ángel Sobrino, Rubén Álvarez, & Verónica Rivas. (2022). Historias de Usuario. *Agosto 2022*, *3*, 62.

Almería noticias. (2014, abril 13). *Historia de los costaleros*. https://www.almerianoticias.es/historia-de-los-costaleros/

Álvaro Jiménez Martín. (2019, junio 18). *React Native: ¿Qué es y para que sirve? | OpenWebinars*. https://openwebinars.net/blog/react-native-que-es-para-que-sirve/

Aragón, H. de. (2017, marzo 23). *El hábito del cofrade, símbolo y tradición de la Semana Santa*. heraldo.es. https://www.heraldo.es/noticias/aragon/zaragoza/2017/03/23/el-habito-del-cofrade-simbolo-tradicion-semana-santa-1165731-2261126.html

Ávila Herrera, J. F. (1997). Asignación óptima de recursos. *Revista de Matemática: Teoría y Aplicaciones*, *4*(2), 75-84.

Briones Gómez, R. (1983). La Semana Santa andaluza. *Gazeta de Antropología*. https://doi.org/10.30827/Digibug.6734

Bryan Salazar López. (2019, junio 11). *Problema del transporte o distribución » Ingenieria Industrial Online*. https://www.ingenieriaindustrialonline.com/investigacion-de-operaciones/problema-del-transporte-o-distribucion/

Bustos, J. L. (2022, abril 5). *¿Qué es el material design? | KeepCoding Bootcamps*. https://keepcoding.io/blog/que-es-material-design/

Carlos López Bravo. (2023, marzo 22). *Antigüedad*. El correo de Andalucía. https://www.elcorreoweb.es/semana-santa/2023/03/22/antiguedad-104461889.html

Colectivo Cabra en el Recuerdo. (2014). *Trabajaderas*. https://www.cabraenelrecuerdo.com/wiki/trabajadera.php

Consejo General de Hermandades y Cofradías de la Ciudad Sevilla. (2012, agosto 31). Historia. *Consejo General de Hermandades y Cofradías de la Ciudad Sevilla*. https://www.hermandades-de-sevilla.org/inicio/consejo/institucion/historia-del-consejo-de-cofradias/

F Moreno. (2019, septiembre 20). *Algoritmos Voraces | Aprende Programación Competitiva*. https://aprende.olimpiada-informatica.org/algoritmia-voraz

Figueroa, I. (2018, marzo 20). *El capataz y el vocero marcan el andar de los pasos en Semana Santa*. COPE. http://www.cope.es/religion/hoy-en-dia/semana-santa/noticias/capataz-vocero-marcan-andar-los-pasos-semana-santa-20180320_175687

Gómez, J. (2018, marzo 15). La Gran Peste de Sevilla, 1649. *Desperta Ferro Ediciones*. https://www.despertaferro-ediciones.com/2018/la-gran-peste-de-sevilla-1649/

Google, Inc. (s. f.). *Firestore*. Firebase. Recuperado 22 de febrero de 2025, de https://firebase.google.com/docs/firestore?hl=es-419

Google Inc. (s. f.). *¿Qué es la arquitectura sin servidor? Google Cloud*. Google Cloud. Recuperado 22 de febrero de 2025, de https://cloud.google.com/discover/what-is-serverless-architecture

Hardik, C. (2023, mayo 29). *Agile vs Waterfall: Key Differences, Pros, and Cons*. Agile vs Waterfall – Know Meaning, Differences, Pros and Con. https://www.valuex2.com/agile-vs-waterfall-methodology-differences-pros-cons/

Hernández, J. (2010, octubre 12). *A la medida de la Semana Santa*. Información. https://www.informacion.es/alicante/2010/10/12/medida-semana-santa-7106201.html

Hernández, J. (2021, junio 2). ¿Cómo pueden los mínimos productos MVP, MMP, MMF o MMR ayudarte a tener éxito en tu proyecto? - Agile Experience. *Agile Experience - Curso Scrum Master - Formación con Certificación Oficial*. https://agileexperience.es/2021/06/02/como-pueden-los-minimos-productos-mvp-mmp-mmf-o-mmr-ayudarte-a-tener-exito-en-tu-proyecto/

Hill, R. (2023, octubre 18). *What is an MVP in Software Development?* SoftwareDevelopment.co.uk. https://www.softwaredevelopment.co.uk/blog/what-is-an-mvp-in-software-development/

Hortelano, Rosa. (2024, mayo 24). *Diccionario cofrade | Rosa Hortelano*. http://www.rosahortelano.com/diccionario-cofrade/

Humberto Guerrero. (2022). *Programación lineal aplicada*. ECOE Ediciones. https://doi.org/10.59979/ECOE.9789585033825.9789585033832

IEEE Recommended Practice for Software Requirements Specifications. (1998). *IEEE Std 830-1998*, 1-40. IEEE Std 830-1998. https://doi.org/10.1109/IEEESTD.1998.88286

Ignacio Díaz Pérez. (2012, marzo 31). *Todos por igual*. https://www.elmundo.es/elmundo/2012/03/31/andalucia_sevilla/www.elmundo.es/elmundo/2012/03/31/andalucia_sevilla/1333212511.html

Introducing Hot Reloading · React Native. (2016, marzo 24). https://reactnative.dev/blog/2016/03/24/introducing-hot-reloading

Ismael Figueroa. (2018, marzo 20). *El capataz y el vocero marcan el andar de los pasos en Semana Santa*. https://www.cope.es/religion/hoy-en-dia/semana-santa/noticias/capataz-vocero-marcan-andar-los-pasos-semana-santa-20180320_175687

Javier Campos. (2021, marzo 25). *Algoritmia básica (AB) » Programación dinámica: Algoritmos de abajo arriba iterativos versus de arriba abajo con memorialización*. https://webdiis.unizar.es/asignaturas/AB/?p=2598

José Javier Peleato. (2020, agosto 24). *2.4 Backtracking | Programación, refactoriza tu mente*. https://docs.jjpeleato.com/algoritmia/backtracking

Las procesiones de Semana Santa—Arguments. (2019, abril 10). https://www.arguments.es/liturgia/las-procesiones-de-semana-santa/

Leonardo Martínez Sandoval. (2022). *Clases de complejidad y P vs. NP — Matemáticas Discretas para Ciencia de Datos*. https://madi.nekomath.com/P5/ReduccionesPvsNP.html

López-Guadalupe, M. L. (2018, enero 31). *Semana Santa en Andalucía*. Identidad e Imagen de Andalucía en la Edad Moderna. https://www2.ual.es/ideimand/semana-santa-en-andalucia/

Mario Pastrana Moreno. (2012, octubre 31). MÉTODO HUNGARO. *Pastranamoreno Blog*. https://pastranamoreno.wordpress.com/2012/10/31/metodo-hungaro/

Ministerio de Industria y Turismo—Fiestas de Interés Turístico Internacional. (s. f.). Recuperado 25 de diciembre de 2024, de https://turismo.gob.es/desarrollo-

sostenibilidad/fiestas/Paginas/Fiestas-de-Inter%C3%A9s-Tur%C3%ADstico-internacional.aspx

Obispo Segorbe. (2019, abril 6). *Las cofradías y procesiones de Semana Santa*. https://obsegorbecastellon.es/las-cofradias-y-procesiones-de-semana-santa/

Ortiz, J. E. (2022, julio 28). *Complejidad ciclomática*. JohnDev. https://www.johndev.co/posts/ciclomatic-complexity/

P. Coto. (2009, mayo 26). DENOMINACION DEL SITIO QUE OCUPAN LOS COSTALEROS Y SUS FUNCIONES... *DENOMINACION DEL SITIO QUE OCUPAN LOS COSTALEROS Y SUS FUNCIONES... ~ Costaleros de Los Ángeles*. https://costalerosvirgendelosangeles.blogspot.com/2009/05/denominacion-del-sitio-que-ocupan-los.html

Peñalver, M. (2022, marzo 24). *Qué significa cada prenda de los nazarenos de Semana Santa*. revista misión. https://www.revistamision.com/nazarenos-de-semana-santa/

Profesor Hossein Arsham. (1994, febrero 25). *Modelos Deterministas: Optimización Lineal*. https://home.ubalt.edu/ntsbarsh/business-stat/opre/spanishd.htm#rlp

Qué es el User Story Mapping| Blog UE. (2023, noviembre 14). Universidad Europea. https://universidadeuropea.com/blog/user-story-mapping/

¿Qué es Miro? – Centro de ayuda de Miro. (s. f.). Recuperado 4 de diciembre de 2024, de https://help.miro.com/hc/es/articles/360017730533--Qu%C3%A9-es-Miro

Rodríguez, C. (2022, febrero 20). Orden del cortejo procesional. *El Güichi de Carlos*. https://www.elguichidecarlos.com/orden-del-cortejo-procesional/

Rodríguez, C., & Dorado Vicente, R. (2015). ¿Por qué implementar Scrum? *Revista ONTARE*, *3*(1), 125-144.

Root Stack. (s. f.). *Creación de aplicaciones escalables con React Native | Rootstack*. Recuperado 19 de febrero de 2025, de https://rootstack.com/es/blog/creacion-de-aplicaciones-escalables-con-react-native

Schwaber, K. (s. f.). *What is a Product Backlog? | Scrum.org*. Learn About the Scrum Artifact: Product Backlog. Recuperado 4 de diciembre de 2024, de https://www.scrum.org/resources/what-is-a-product-backlog

Scrum Alliance. (s. f.). *A Guide to Using the Fibonacci Sequence in Scrum | Resource Library*. Recuperado 4 de diciembre de 2024, de https://resources.scrumalliance.org/Article/guide-using-fibonacci-sequence-scrum

Shuheng.Ma. (2024, mayo 3). *Recursion vs Dynamic Programming—Fibonacci(Leetcode 509)*. Medium. https://towardsdatascience.com/dynamic-programming-i-python-8b20387870f5

Silvia U. (2022, marzo 4). *Introducción a Product Discovery | Cátedra Viewnext USAL*. https://viewnext.usal.es/blog/introducci%C3%B3n-product-discovery

Stanislav Naborshchikov. (2023, junio 14). *Cross-platform vs native app development: Final Comparison*. https://themobilereality.com/blog/Mobile Reality

StyleSheet · React Native. (s. f.). Recuperado 19 de febrero de 2025, de https://reactnative.dev/docs/stylesheet

Ugarte Pereira, J. M. (2019). La Semana Santa como hecho social y teatral. *Belezos: Revista de cultura popular y tradiciones de La Rioja*, *39*, 74-80.

UNIR, Grado en Informática. (2021, marzo 18). *Guía docente*. https://static.unir.net/guias_espana/guias_nuevas/gii25_tfg.htm

Vallés, T. (2018). *Catholic.net - [La Semana santa]*. catholic.net. https://es.catholic.net/op/articulos/18297/la-semana-santa.html#modal

webmaster. (2023, octubre 14). Requerimientos funcionales y no funcionales en Scrum. *Metodologias Ágiles*. https://metodologiasagiles.org/requerimientos-funcionales-y-no-funcionales-en-scrum/

Anexo A. Requisitos funcionales del sistema

RF01	Creación de paso de Semana Santa
Versión	1.0
Dependencias	Ninguna
Descripción	El sistema debe permitir al usuario crear un trono especificando: • Nombre • Descripción
Importancia	Alta

Tabla 24. RF01 Creación de pasos.

RF02	Añadir trabajaderas y huecos al paso
Versión	1.0
Dependencias	RF01
Descripción	El sistema debe permitir añadir trabajaderas a un paso especificando: • Altura • Número de huecos
Importancia	Alta

Tabla 25. RF02 Añadir trabajaderas y huecos al paso.

RF03	Editar / Borrar Paso
Versión	**1.0**
Dependencias	**RF01**
Descripción	**La aplicación debe permitir a un capataz editar los detalles de un paso creado con anterioridad, incluyendo poder borrar el paso. En caso de borrar el paso, se borrarán con él los ensayos y costaleros asociados al mismo.**
Importancia	**Alta**

Tabla 26. RF03 Editar - borrar paso

RF04	Añadir costaleros
Versión	**1.0**
Dependencias	**RF01**
Descripción	**El sistema debe permitir al capataz dar de alta a costaleros en un paso, siempre especificando:** • **Nombre** • **Apellidos** • **Teléfono** • **Altura en centímetros**
Importancia	**Alta**

Tabla 27. RF04 Añadir costaleros

RF05	Editar / Borrar Costalero
Versión	1.0
Dependencias	RF04
Descripción	La aplicación debe permitir a un capataz editar los datos de un costalero, así cómo eliminarlo de la cuadrilla.
Importancia	Alta

Tabla 28. RF05 Editar - borrar costalero

RF06	Añadir foto a ficha de costalero
Versión	1.0
Dependencias	RF04
Descripción	El capataz puede añadir una foto a la ficha de costalero para organizarlo mejor visualmente
Importancia	Baja

Tabla 29. RF06 Añadir foto a ficha de costalero

RF07	Crear ensayo
Versión	1.0
Dependencias	Ninguna
Descripción	El sistema debe permitir al capataz crear un ensayo, especificando fecha y hora de este
Importancia	Alta

Tabla 30. RF07 Crear ensayo

RF08	Ver histórico de ensayos
Versión	1.0
Dependencias	RF07
Descripción	La aplicación debe ofrecerle al capataz un histórico de los ensayos celebrados con anterioridad a la fecha de hoy
Importancia	Alta

Tabla 31. RF08 Ver histórico de ensayos

RF09	Gestionar ensayos pasados
Versión	1.0
Dependencias	RF08
Descripción	El capataz debe de ser capaz de ver los detalles de los ensayos pasados, así como ver informes de este.
Importancia	Media

Tabla 32. RF09 Gestionar ensayos pasados

RF10	Copiar ensayo
Versión	1.0
Dependencias	RF08
Descripción	La aplicación ofrecerá la opción de, en vez de crear un ensayo desde cero, crear uno reutilizando los detalles de otro creado anteriormente
Importancia	Media

Tabla 33. RF10 Copiar ensayo

RF11	Asignación automática de costaleros a las trabajaderas
Versión	**1.0**
Dependencias	**RF04, RF01, RF02, RF07**
Descripción	**La aplicación ofrecerá la opción de asignar los costaleros a las trabajaderas de forma óptima.** **Se considera forma óptima la opción en la que el número de tacos o suplementos a utilizar para compensar la altura con respecto a la trabajadera sea la mínima.**
Importancia	**Alta**

Tabla 34. RF11 Asignación automática de costaleros

RF12	Asignación de costaleros de forma manual
Versión	**1.0**
Dependencias	**RF04, RF01, RF02, RF07**
Descripción	**El capataz debe de poder asignar costaleros de forma manual.**
Importancia	**Alta**

Tabla 35. RF12 Asignar costaleros de forma manual

RF13	Generación de plantilla de costaleros
Versión	1.0
Dependencias	Ninguna
Descripción	La aplicación debe dar la posibilidad de generar un informe de cómo se han colocado los costaleros en el paso, para tener una visión clara y concisa de cómo están situados los costaleros en un determinado ensayo.
Importancia	Media

Tabla 36. RF13 Generación de plantilla de costaleros

RF14	Generar informe de asistencia
Versión	1.0
Dependencias	Ninguna
Descripción	Generación de un informe de asistencia durante un intervalo de ensayos. Este informe se agrupará por costalero y se totalizará por número de asistencias a ensayos
Importancia	Baja

Tabla 37. Generar informe de asistencia

RF15	Exportar informes a PDF
Versión	1.0
Dependencias	Ninguna
Descripción	Posibilidad de, los informes descritos en los RF13 y RF14, exportarlos en formato PDF
Importancia	Baja

Tabla 38. RF15 Exportar informes a PDF

RF16	Notificar no asistencia vía WhatsApp
Versión	1.0
Dependencias	Ninguna
Descripción	Integración con la API de WhatsApp para enviar un mensaje al costalero que tenga faltas de asistencia, recordándole la fecha del próximo ensayo
Importancia	Baja

Tabla 39. RF16 Notificar no asistencia vía WhatsApp

RF17	Registro de capataz en el sistema usando Google SSO
Versión	1.0
Dependencias	Ninguna
Descripción	Módulo de registro de un usuario en el sistema usando el usuario de Google y su API.
Importancia	Alta

Tabla 40. RF17 Registro capataz en el sistema usando Google SSO

RF18	Registro de usuario en el sistema usando correo y contraseña
Versión	1.0
Dependencias	Ninguna
Descripción	Módulo de registro de un usuario en el sistema de forma tradicional, facilitando usuario y contraseña.
Importancia	Baja

Tabla 41. RF18 Registro capataz en el sistema usando usuario y contraseña

RF19	Registro de usuario usando otros SSO
Versión	1.0
Dependencias	Ninguna
Descripción	Módulo de registro de un usuario en el sistema usando SSO cómo Facebook o Apple.
Importancia	Baja

Tabla 42. Registro de usuario usando otros SSO cómo Facebook o Apple

RF20	Enviar invitación a ayudantes
Versión	1.0
Dependencias	RF17, RF18, RF19
Descripción	El capataz debe ser capaz de invitar a un ayudante a ver los datos del trono
Importancia	Media

Tabla 43. RF20 Enviar invitación a capataces auxiliares

RF21	Gestionar capataces auxiliares
Versión	**1.0**
Dependencias	**RF20**
Descripción	**La aplicación debe poder gestionar los capataces auxiliares, pudiendo eliminar el capataz auxiliar o añadir nuevos.**
Importancia	**Media**

Tabla 44. RF21 Gestionar capataces auxiliares

Anexo B. Cuadrante Costaleros Semana Santa 2024

El siguiente anexo detalla el cuadrante de costaleros para la Semana Santa del año 2024. El nombre de los costaleros se ha modificado para cumplir con la normativa de protección de datos.

Delantera del paso					
T1 1.57	Mario Ortiz	Inma Rubio +2		David Gallardo +1	Clara Moreno +7
T2 1.52	Jesús Márquez	Paula Ortega +2		Lucas Márquez +2	María Arroyo +1
T3 1.54	Emilio Márquez	Adrián Rojas		Marcos Gómez	Andrea Domínguez +2
T4 1.48	Rafael Domínguez	Óscar Vargas	Mercedes Prieto	Bernardo Castro +1	Daniel Serrano
T5 1.47	Sebastián Reyes	Bruno González	Pablo Navarro	Agustín Fernández	Andrés Molina +2
T6 1.42	Miguel Sánchez	Luis Castillo +6	Manuel Jiménez +12	Jesús Romero +3	Raúl Herrera
T7 1.45	Gustavo Nieto	Jesús Herrera	Javier Castillo	Jorge Castillo	Moisés Sánchez

Tabla 45. Cuadrante procesión Semana Santa 2024

Coste total = T1 + T2 + T3 + T4 + T5 + T6 + T7 = 9 + 5 + 2 + 1 + 2 + 21 + 0 = 40

Índice de acrónimos

M

MVP. Minimum viable product.

MMP. Minimum marketable product.

N

NP. Del inglés *Nondeterministic polynomial time.* Tiempo polinómico no determinista

R

RF. Requisito funcional.

RNF. Requisito no funcional.

S

SDK. Software development kit.

T

TFG. Trabajo fin de grado

U

UI. Del inglés *User interface.* Interfaz de usuario