
**APLICACIÓN WEB PARA LA MEJORA DE
LA ALIMENTACIÓN Y LA PROMOCIÓN DE
LA ACTIVIDAD FÍSICA**
**WEB APPLICATION FOR THE
IMPROVEMENT OF DIET AND THE
PROMOTION OF PHYSICAL ACTIVITY**



**Trabajo de Fin de Grado
Curso 2022–2023**

Autoras

Almudena Naharro Muñoz
Inés Hernández Hurtado
Patricia Plata Barroso

Directora

Sonia Estévez Martín

Colaboradora

Mercedes Martínez Cortés

Grado en Ingeniería Informática

Facultad de Informática

Universidad Complutense de Madrid

APLICACIÓN WEB PARA LA MEJORA DE LA
ALIMENTACIÓN Y LA PROMOCIÓN DE LA
ACTIVIDAD FÍSICA
WEB APPLICATION FOR THE IMPROVEMENT OF
DIET AND THE PROMOTION OF PHYSICAL
ACTIVITY

Trabajo de Fin de Grado en Ingeniería Informática

Autoras
Almudena Naharro Muñoz
Inés Hernández Hurtado
Patricia Plata Barroso

Directora
Sonia Estévez Martín
Colaboradora
Mercedes Martinez Cortés

Convocatoria: *Junio 2023*

Grado en Ingeniería Informática
Facultad de Informática
Universidad Complutense de Madrid

7 de JUNIO de 2023

Dedicatoria

Queremos agradecer a nuestras familias por su constante apoyo durante todo el trayecto y, especialmente, durante esta última etapa en la que estamos cerca de finalizar la carrera. Su presencia ha sido muy valiosa para nosotras.

Agradecimientos

Nos gustaría expresar nuestro agradecimiento a nuestra tutora, Sonia Estévez Martín, y a Mercedes Martínez Cortés por brindarnos la oportunidad de llevar a cabo este Trabajo de Fin de Grado. Apreciamos mucho todo su apoyo y dedicación durante todo el proceso. En momentos necesarios, su determinación y decisión fueron fundamentales para ayudarnos a superar obstáculos y llevar a cabo el trabajo de manera exitosa. Su combinación de orientación y libertad nos permitió llevar a cabo este trabajo de manera exitosa y sin problemas.

Resumen

APLICACIÓN WEB PARA LA MEJORA DE LA ALIMENTACIÓN Y LA PROMOCIÓN DE LA ACTIVIDAD FÍSICA

La salud es un aspecto fundamental en nuestra vida, que nos permite disfrutar de bienestar y calidad. En los últimos años, se ha observado un aumento en los índices de obesidad, lo cual representa un riesgo para la salud de las personas.

Por el problema planteado, en este trabajo se ha desarrollado una aplicación web adaptable para dispositivos móviles para llevar un seguimiento adecuado de las personas que se encuentran en sobre peso o en riesgo de diabetes, inscritos en el programa del Centro de Salud Comunitaria Latina de Madrid.

La aplicación ofrece una funcionalidad completa para llevar un seguimiento preciso del peso y la medida de la cintura, permitiendo establecer objetivos personalizados. Además, existe la posibilidad de participar en actividades relacionadas con la salud y el ejercicio, así como mantener una comunicación constante con el coordinador del programa.

Se incluye un apartado de alimentación, donde el usuario puede registrar su ingesta de alimentos, calcular su índice calórico diario recomendado y crear sus propias recetas.

Mediante el uso de técnicas de aprendizaje automático (Machine Learning), la aplicación es capaz de recomendar recetas personalizadas que se ajustan a los gustos y necesidades del usuario. Para hacer la experiencia más interactiva y dinámica, se ha incorporado un juego educativo que enseña a los usuarios a cómo llevar a cabo una alimentación saludable.

Palabras clave

SpringBoot, Aplicación web, Alimentación, Seguimiento, Participante, Coordinador, Progreso, Objetivos, Estepper, Machine Learning

Abstract

WEB APPLICATION FOR THE IMPROVEMENT OF DIET AND THE PROMOTION OF PHYSICAL ACTIVITY

Health is a fundamental aspect of our lives, which allows us to enjoy well-being and quality. In recent years, an increase in obesity rates has been observed, which represents a risk to people's health.

Because of the problem posed, in this work we have developed an adaptive web application for mobile devices to keep track of people who are overweight or at risk of diabetes, enrolled in the program of Centro de Salud Comunitaria Latina de Madrid

The application offers full functionality to accurately track weight and waist measurement, allowing to set personalized goals. In addition, there is the possibility to participate in health and exercise related activities, as well as maintain constant communication with the program coordinator.

A food section is included, where the user can record their food intake, calculate their recommended daily calorie index and create their own recipes.

Through the use of Machine Learning techniques, the application is able to recommend personalized recipes that match the user's tastes and needs. To make the experience more interactive and dynamic, an educational game has been incorporated to teach users how to eat healthy.

Keywords

SpringBoot, Web application, Nutrition, monitoring, Participant, Coordinator, Progress, Objectives, Estepper, Machine Learning

Índice

1. Introducción	1
1.1. Antecedentes	1
1.2. Objetivos	2
1.3. Plan de trabajo	3
2. Estado del Arte	7
3. Tecnologías utilizadas	9
3.1. Front-end (lado del cliente)	9
3.1.1. HTML	9
3.1.2. JavaScript	9
3.1.3. CSS y Bootstrap	9
3.1.4. Thymeleaf	9
3.2. Back-end (lado del servidor)	10
3.2.1. Xampp	10
3.2.2. Java	10
3.2.3. Spring Boot	10
3.2.4. Jython	11
3.2.5. Gmail API	11
3.3. Bibliotecas	11
3.4. Aplicaciones	12
3.5. Instalación del servidor	13
4. Diseño gráfico de la aplicación	15
4.1. Nombre	15
4.2. Logo	16
4.3. Estilo	16
4.4. Colores	19
4.5. Bocetos y prototipos Balsamiq	19
4.6. Principios de diseño	23
5. Diseño funcional de la aplicación	27

5.1.	Arquitectura de la aplicación	27
5.2.	Base de datos	28
5.3.	Estructura del código	31
5.3.1.	Diagrama de componentes	31
5.3.2.	Modelo Vista Controlador	32
5.3.3.	Inyección de dependencias	33
5.3.4.	Seguridad	34
5.4.	Aplicación responsive	35
6.	Especificación de requisitos	39
6.1.	Actores	39
6.1.1.	Administrador	39
6.1.2.	Coordinador	39
6.1.3.	Participante	40
6.1.4.	Usuario	40
6.2.	Diagrama casos de uso UML	40
6.2.1.	Gestión de usuarios	40
6.2.2.	Actividades	41
6.2.3.	Materiales	41
6.2.4.	Progreso	42
6.2.5.	Chat	42
6.2.6.	Alimentación	43
6.2.7.	Objetivos	43
6.2.8.	Cuaderno y sesiones	44
6.2.9.	Gestión de participantes	44
6.2.10.	Gestión de grupos	45
6.3.	Casos de uso	45
6.3.1.	Gestión de usuarios	45
6.3.2.	Actividades	51
6.3.3.	Materiales	61
6.3.4.	Progreso	69
6.3.5.	Chat	75
6.3.6.	Alimentación	85
6.3.7.	Objetivos	96
6.3.8.	Cuaderno y sesiones	107
6.3.9.	Gestión de participantes	113
6.3.10.	Gestión de grupos	125
7.	Implementación de la aplicación	135
7.1.	Diagramas de actividad	135
7.1.1.	Gestión de usuarios	135
7.1.2.	Actividades	136
7.1.3.	Materiales	136
7.1.4.	Progreso	137

7.1.5. Chat	137
7.1.6. Alimentación	138
7.1.7. Objetivos	138
7.1.8. Cuaderno y sesiones	139
7.1.9. Gestión de participantes	139
7.1.10. Gestión de grupos	140
7.2. Back-End o lado del servidor	140
7.2.1. Gestión de usuarios	140
7.2.2. Actividades	145
7.2.3. Materiales	151
7.2.4. Progreso	154
7.2.5. Chat	159
7.2.6. Alimentación	164
7.2.7. Objetivos	170
7.2.8. Cuaderno y sesiones	177
7.2.9. Gestión de participantes	180
7.2.10. Gestión de grupos	184
7.3. Front-End o lado del cliente	188
7.3.1. Gestión de usuarios	188
7.3.2. Actividades	190
7.3.3. Materiales	193
7.3.4. Progreso	194
7.3.5. Chat	196
7.3.6. Alimentación	199
7.3.7. Objetivos	202
7.3.8. Cuaderno y sesiones	205
7.3.9. Gestión de participantes	209
7.3.10. Gestión de grupos	213
8. Resultados y discusión crítica	217
9. Conclusiones y Trabajo Futuro	219
Introduction	221
Conclusions and Future Work	227
Contribuciones Personales	229
Bibliografía	239
A. Vídeos de demostración	241
B. Guía de instalación	243

Índice de figuras

1.1. Diagrama de Gantt	5
4.1. Logo de Estepper	16
4.2. Diseño vista principal alimentación	17
4.3. Diseño pantalla principal vista móvil de un Participante	18
4.4. Diseño vasos de agua	18
4.5. Diseño bloc de notas	19
4.6. Paleta de colores	19
4.7. Ejemplos de bocetos	21
4.8. Ejemplos del prototipo Balsamiq	22
4.9. Vista del listado de participantes	23
4.10. Vista de la página de login	24
4.11. Vista de la página de registro	24
4.12. Vista móvil del chat de ayuda con el administrador	24
4.13. Ejemplos del principio de feedback en la aplicación	25
4.14. Acceso a los detalles de una actividad	25
4.15. Editar perfil del participante	26
5.1. Arquitectura de la aplicación	27
5.2. Esquema de base de datos	30
5.3. Diagrama de componentes	31
5.4. Ejemplo dependencia Thymeleaf	34
5.5. Vista responsive de inicio sesión, el inicio de participante y coordinador	35
5.6. Vista responsive de actividades, progreso y objetivos	36
5.7. Vista responsive de materiales, alimentación y sesiones	36
5.8. Vista responsive de gestión de participantes, grupos y chat	37
6.1. Diagrama de casos de uso de gestión de usuarios	40
6.2. Diagrama de casos de uso de actividades	41
6.3. Diagrama de casos de uso de materiales	41
6.4. Diagrama de casos de uso de progreso	42
6.5. Diagrama de casos de uso de chat	42

6.6. Diagrama de casos de uso de alimentación	43
6.7. Diagrama de casos de uso de objetivos	43
6.8. Diagrama de casos de uso de cuaderno y sesiones	44
6.9. Diagrama de casos de uso de gestión de participantes	44
6.10. Diagrama de casos de uso de grupos	45
6.11. Caso de uso U01	45
6.12. Caso de uso U02	46
6.13. Caso de uso U03	46
6.14. Caso de uso U04	47
6.15. Caso de uso U05	47
6.16. Caso de uso U06	48
6.17. Caso de uso U07	48
6.18. Caso de uso U08	49
6.19. Caso de uso U09	50
6.20. Caso de uso U10	50
6.21. Caso de uso U11	51
6.22. Caso de uso U12	51
6.23. Caso de uso AT01	52
6.24. Caso de uso AT03	52
6.25. Caso de uso AT14	53
6.26. Caso de uso AT15	53
6.27. Caso de uso AT04	54
6.28. Caso de uso AT05	55
6.29. Caso de uso AT16	56
6.30. Caso de uso AT06	56
6.31. Caso de uso AT17	57
6.32. Caso de uso AT07	57
6.33. Caso de uso AT02	58
6.34. Caso de uso AT08	58
6.35. Caso de uso AT09	59
6.36. Caso de uso AT10	59
6.37. Caso de uso AT18	60
6.38. Caso de uso AT11	60
6.39. Caso de uso AT12	61
6.40. Caso de uso AT13	61
6.41. Caso de uso M01	62
6.42. Caso de uso M03	62
6.43. Caso de uso M02	63
6.44. Caso de uso M05	63
6.45. Caso de uso M04	64
6.46. Caso de uso M08	64
6.47. Caso de uso M07	65
6.48. Caso de uso M06	65

6.49. Caso de uso M10	66
6.50. Caso de uso M09	67
6.51. Caso de uso M13	67
6.52. Caso de uso M12	68
6.53. Caso de uso M11	69
6.54. Caso de uso PG01	69
6.55. Caso de uso PG03	70
6.56. Caso de uso PG13	70
6.57. Caso de uso PG04	71
6.58. Caso de uso PG05	71
6.59. Caso de uso PG06	72
6.60. Caso de uso PG14	72
6.61. Caso de uso PG07	73
6.62. Caso de uso PG08	73
6.63. Caso de uso PG02	74
6.64. Caso de uso PG09	74
6.65. Caso de uso PG10	74
6.66. Caso de uso PG11	75
6.67. Caso de uso PG12	75
6.68. Caso de uso CH01	76
6.69. Caso de uso CH02	76
6.70. Caso de uso CH03	77
6.71. Caso de uso CH04	77
6.72. Caso de uso CH05	78
6.73. Caso de uso CH06	78
6.74. Caso de uso CH07	79
6.75. Caso de uso CH08	79
6.76. Caso de uso CH09	80
6.77. Caso de uso CH10	81
6.78. Caso de uso CH11	81
6.79. Caso de uso CH12	82
6.80. Caso de uso CH13	82
6.81. Caso de uso CH14	83
6.82. Caso de uso CH15	83
6.83. Caso de uso CH16	84
6.84. Caso de uso CH17	84
6.85. Caso de uso CH18	85
6.86. Caso de uso AL16	86
6.87. Caso de uso AL02	86
6.88. Caso de uso AL14	87
6.89. Caso de uso AL15	87
6.90. Caso de uso AL01	88
6.91. Caso de uso AL04	88

6.92. Caso de uso AL05	89
6.93. Caso de uso AL06	89
6.94. Caso de uso AL08	90
6.95. Caso de uso AL10	90
6.96. Caso de uso AL11	91
6.97. Caso de uso AL09	91
6.98. Caso de uso AL12	92
6.99. Caso de uso AL13	92
6.100Caso de uso AL07	93
6.101Caso de uso AL21	93
6.102Caso de uso AL03	94
6.103Caso de uso AL19	94
6.104Caso de uso AL20	95
6.105Caso de uso AL17	95
6.106Caso de uso AL18	96
6.107Caso de uso OB01	96
6.108Caso de uso OB02	97
6.109Caso de uso OB03	97
6.110Caso de uso OB04	98
6.111Caso de uso OB05	98
6.112Caso de uso OB06	99
6.113Caso de uso OB07	100
6.114Caso de uso OB08	101
6.115Caso de uso OB09	102
6.116Caso de uso OB10	102
6.117Caso de uso OB11	103
6.118Caso de uso OB12	103
6.119Caso de uso OB13	104
6.120Caso de uso OB14	104
6.121Caso de uso OB15	105
6.122Caso de uso OB16	105
6.123Caso de uso OB17	106
6.124Caso de uso OB18	106
6.125Caso de uso OB19	107
6.126Caso de uso S01	107
6.127Caso de uso S02	108
6.128Caso de uso S05	108
6.129Caso de uso S06	109
6.130Caso de uso S07	109
6.131Caso de uso S08	110
6.132Caso de uso S03	110
6.133Caso de uso S09	111
6.134Caso de uso S10	111

6.135Caso de uso S12	112
6.136Caso de uso S11	112
6.137Caso de uso S04	113
6.138Caso de uso S13	113
6.139Caso de uso PA01	114
6.140Caso de uso PA02	114
6.141Caso de uso PA03	115
6.142Caso de uso PA04	115
6.143Caso de uso PA05	116
6.144Caso de uso PA06	116
6.145Caso de uso PA07	117
6.146Caso de uso PA08	117
6.147Caso de uso PA09	118
6.148Caso de uso PA11	118
6.149Caso de uso PA10	119
6.150Caso de uso PA12	119
6.151Caso de uso PA13	120
6.152Caso de uso PA19	120
6.153Caso de uso PA14	121
6.154Caso de uso PA20	121
6.155Caso de uso PA15	122
6.156Caso de uso PA21	122
6.157Caso de uso PA16	123
6.158Caso de uso PA22	123
6.159Caso de uso PA17	124
6.160Caso de uso PA23	124
6.161Caso de uso PA18	125
6.162Caso de uso PA24	125
6.163Caso de uso GR01	126
6.164Caso de uso GR02	126
6.165Caso de uso GR03	127
6.166Caso de uso GR04	128
6.167Caso de uso GR05	128
6.168Caso de uso GR06	129
6.169Caso de uso GR07	130
6.170Caso de uso GR08	131
6.171Caso de uso GR09	132
6.172Caso de uso GR10	133
6.173Caso de uso GR11	133
6.174Caso de uso GR12	134
7.1. Diagrama de actividades de gestión de usuarios	135
7.2. Diagrama de actividades de gestión de actividades	136

7.3.	Diagrama de actividades de materiales	136
7.4.	Diagrama de actividades de progreso	137
7.5.	Diagrama de actividades de chat	137
7.6.	Diagrama de actividades de alimentación	138
7.7.	Diagrama de actividades de objetivos	138
7.8.	Diagrama de actividades de cuaderno y sesiones	139
7.9.	Diagrama de actividades de gestión de participantes	139
7.10.	Diagrama de actividades de grupos	140
7.11.	Vista de iniciar sesión siendo Usuario	188
7.12.	Vista de registrarse siendo Usuario	188
7.13.	Vista de recuperar datos siendo Usuario	189
7.14.	Vista de ver perfil siendo Usuario	189
7.15.	Vista de acceder al listado siendo Administrador	189
7.16.	Vista de crear coordinador siendo Administrador	190
7.17.	Vista de enviar mensaje de incidencia siendo Administrador	190
7.18.	Vista de acceder las actividades siendo Participante	190
7.19.	Vista de ver invitaciones siendo Participante	191
7.20.	Vista de ver actividad y realizar inscripción siendo Participante	191
7.21.	Vista de ver inscripciones siendo Participante	191
7.22.	Vista de gestionar las actividades siendo Coordinador	192
7.23.	Vista de crear actividad siendo Coordinador	192
7.24.	Vista de ver actividad siendo Coordinador	192
7.25.	Vista de editar actividad siendo Coordinador	193
7.26.	Vista de ver invitaciones siendo Coordinador	193
7.27.	Vista de los materiales grupales e individuales siendo Coordinador	193
7.28.	Vista de los materiales siendo Participante	194
7.29.	Vista de acceder al progreso propio siendo Participante	194
7.30.	Vista de ver gráfica peso y perímetro siendo Participante	194
7.31.	Vista de ver registros peso y perímetro siendo Participante	195
7.32.	Vista de registrar peso siendo Participante	195
7.33.	Vista de registrar perímetro siendo Participante	195
7.34.	Vista de acceder al progreso de un participante siendo Coordinador	196
7.35.	Vista de chat de ayuda siendo Coordinador	196
7.36.	Vista de chat privado con un participante siendo Coordinador	197
7.37.	Vista del chat privado con el coordinador siendo Participante	197
7.38.	Vista de mensaje rápido siendo Administrador	198
7.39.	Vista de conversaciones privadas con usuarios siendo Administrador	198
7.40.	Vista del inicio de la funcionalidad siendo Participante	199
7.41.	Vista del juego siendo Participante	199
7.42.	Vista de los alimentos de hoy siendo Participante	199
7.43.	Vista de crear un nuevo alimento siendo Participante	200
7.44.	Vista de las recetas siendo Participante	200
7.45.	Vista de una receta siendo Participante	200

7.46. Vista de las recomendaciones de recetas siendo Participante	201
7.47. Vista de crear una receta siendo Participante	201
7.48. Vista de los nutrientes necesarios siendo Participante	201
7.49. Vista del botón para ver las recetas parecidas siendo Participante . .	202
7.50. Vista de las recetas parecidas siendo Participante	202
7.51. Vista de objetivos recomendados (agua) siendo Participante	202
7.52. Vista de objetivos recomendados (ejercicio) siendo Participante . . .	203
7.53. Vista de objetivos recomendados (descanso) siendo Participante . . .	203
7.54. Vista de objetivos recomendados (estado de ánimo) siendo Participante	203
7.55. Vista de objetivos personalizados siendo Participante	204
7.56. Vista de crear objetivo personalizado siendo Participante	204
7.57. Vista de acceso al cuaderno siendo Participante	205
7.58. Vista de las diapositivas siendo Participante	205
7.59. Vista de las fichas siendo Participante	205
7.60. Vista de la ficha del taller siendo Participante	206
7.61. Vista de la ficha del peso siendo Participante	206
7.62. Vista de acceso a las fichas de elección siendo Participante	206
7.63. Vista de la ficha de la elección siendo Participante	207
7.64. Vista de acceso a las sesiones siendo Participante	207
7.65. Vista de una sesión siendo Participante	207
7.66. Vista de terminar una sesión siendo Participante	208
7.67. Vista de acceso a las sesiones siendo Coordinador	208
7.68. Vista de una sesión siendo Coordinador	208
7.69. Vista de datos de peso y cintura siendo Coordinador	209
7.70. Vista de test de Finsdrisc siendo Usuario	209
7.71. Vista de recomendaciones siendo Usuario	209
7.72. Vista de las notificaciones siendo Participante	210
7.73. Vista de la lista de participantes siendo Coordinador	210
7.74. Vista del expediente de un participante siendo Coordinador	210
7.75. Vista de la fase de valoración de un participante siendo Coordinador	211
7.76. Vista del test de exploración de un participante siendo Coordinador	211
7.77. Vista del test de findrisc de un participante siendo Coordinador . .	211
7.78. Vista del test de clasificación de un participante siendo Coordinador	212
7.79. Vista del test de antecedentes de un participante siendo Coordinador	212
7.80. Vista del test de adherencia a la dieta mediterránea de un participante siendo Coordinador	212
7.81. Vista del test de actividad física de un participante siendo Coordinador	213
7.82. Vista del listado de grupos dirigidos por un Coordinador	213
7.83. Vista de las notas y observaciones de un grupo siendo Coordinador .	213
7.84. Vista de un grupo siendo Coordinador	214
7.85. Vista de editar un grupo siendo Coordinador	214
7.86. Vista de mi grupo siendo Participante	215
9.1. Gantt Chart	224

Capítulo **1**

Introducción

“La tecnología, aplicada correctamente, puede ser un agente transformador en el cuidado de la salud”
— Eric Topol

En esta sección se presenta el proyecto de aplicación web para la mejora de la alimentación y la promoción de la actividad física. Se detallan los antecedentes, los objetivos, así como el plan de trabajo y las asignaturas que han sido útiles en su realización.

1.1. Antecedentes

Este trabajo surge de la necesidad de informatizar parte del programa de Alimentación, Actividad física y Salud (ALAS) del Organismo Autónomo de Madrid Salud. El programa tiene como objetivos facilitar la realización de ejercicio físico de forma regular y promover una alimentación sana. Está diseñado especialmente para la población que tiene sobrepeso u obesidad y las personas con mayor riesgo de desarrollar diabetes y enfermedades cardiovasculares.

Para lograr estos objetivos, el programa ALAS ofrece talleres dirigidos a la población general con el fin de informar y sensibilizar sobre cómo los hábitos saludables en cuanto a alimentación y actividad física pueden mejorar la salud y la calidad de vida. Además, el programa se dirige a la población de la ciudad de Madrid en diferentes ámbitos, como el familiar y comunitario, escolar, empresarial y sanitario.

El programa también incluye una intervención intensiva con las personas que tienen diferentes grados de riesgo derivados de una alimentación inadecuada y del sedentarismo. Esta intervención se enfoca en ofrecer a estas personas un seguimiento individualizado y un plan de acción específico que les ayude a adoptar hábitos saludables en su día a día y mejorar su estado de salud. En resumen, el programa ALAS tiene como objetivo principal promover un estilo de vida saludable y prevenir enfermedades relacionadas con la alimentación y la falta de actividad física.

Mercedes Martínez Cortés, una de las organizadoras del programa ALAS, nos ha proporcionado información adicional sobre las mejoras que planea implementar en el programa para lograr sus objetivos. Según Mercedes, una de las principales

mejoras que propone es la implementación de una aplicación web para monitorizar mejor a los participantes y facilitar su seguimiento individualizado.

Además, Mercedes sugiere la creación de herramientas y recursos para los participantes, como ideas de comidas saludables, recetas y alimentos recomendados, para ayudarles a controlar su alimentación y mejorar sus hábitos. También sugiere la realización de actividades y eventos que permitan a los participantes interactuar entre sí y fomentar un ambiente de apoyo y motivación. Mercedes también plantea que se permita a los participantes proponerse sus objetivos para conseguirlos poco a poco, lo que puede ser de gran ayuda para fomentar la motivación y el compromiso con el programa.

Otra de las mejoras propuestas por Mercedes para el programa ALAS incluyen la implementación de un chat para que los participantes puedan comunicarse y recibir orientación en tiempo real. Además, se creará un dashboard en el que los participantes puedan registrar y hacer seguimiento de su progreso en términos de actividad física y hábitos alimentarios. Esto permitiría a los participantes tener una visión clara de su evolución y hacer ajustes en su plan de acción si es necesario.

Por otra parte, se ha propuesto la creación de un campus virtual para enviar y recibir materiales, como ideas de comidas saludables, recetas y ejercicios recomendados. Esto brindará a los participantes un acceso fácil a la información y recursos necesarios para llevar a cabo su plan de acción y lograr sus objetivos.

En resumen, Mercedes Martínez Cortés, organizadora del programa ALAS, ha compartido información adicional sobre las mejoras que planea implementar en el programa. Entre estas mejoras se encuentra la creación de una aplicación web, que se pueda usar también en el móvil, para monitorizar mejor a los participantes y facilitar su seguimiento individualizado, así como la creación de herramientas y recursos como ideas de comidas saludables y recetas recomendadas. También se propone la implementación de un chat para que los participantes puedan comunicarse y recibir orientación en tiempo real, un dashboard para registrar y hacer seguimiento de su progreso, y un campus virtual para enviar y recibir materiales. Además, se permitirá a los participantes proponerse sus objetivos para conseguirlos poco a poco y fomentar su motivación y compromiso con el programa.

1.2. Objetivos

Hemos dividido los requisitos planteados por Mercedes Martínez Cortés en distintas funcionalidades con el fin de mejorar la efectividad del programa ALAS. A continuación, se describen cada una de estas funcionalidades:

Gestión de usuarios: Implementar una aplicación web, adaptable a pantallas de dispositivos móviles, que permita a los usuarios registrarse, crear su perfil y acceder a las herramientas y recursos del programa.

Gestión de participantes: Facilitar el seguimiento individualizado de los participantes mediante la monitorización de su progreso, comenzando por realizar una fase de valoración.

Gestión de grupos: Facilitar el seguimiento de los grupos de participantes para fomentar su interacción mediante la realización de un chat.

Cuaderno y sesiones: Proporcionar a los participantes herramientas y recursos para mejorar sus hábitos alimentarios, informatizando fichas y sesiones del propio programa. Además de hacer un seguimiento de la asistencia a las distintas sesiones y sus avances.

Materiales: Crear un campus virtual para enviar y recibir materiales relacionados con el programa, como ideas de comidas saludables, recetas y ejercicios recomendados.

Chat: Implementar un chat para que los participantes puedan comunicarse y recibir orientación en tiempo real de parte de los coordinadores y otros participantes.

Actividades: Organizar actividades y eventos que fomenten un ambiente de apoyo y motivación entre los participantes, tales como clases de yoga, caminatas grupales, entre otras.

Objetivos: Permitir a los participantes proponerse sus propios objetivos para conseguirlos poco a poco, fomentando así su motivación y compromiso con el programa.

Alimentación: Proporcionar herramientas y recursos para mejorar la alimentación de los participantes, tales como ideas de recetas y alimentos recomendados. Además de hacer un seguimiento de los alimentos que ha consumido durante el día para controlar fácilmente los nutrientes obtenidos.

Progreso: Crear un dashboard para que los participantes puedan registrar y hacer seguimiento de su progreso en términos de actividad física y hábitos alimentarios, permitiendo así una visión clara de su evolución y hacer ajustes en su plan de acción si es necesario.

Cada una de estas funcionalidades es esencial para brindar a los participantes del programa ALAS herramientas y recursos que les permitan mejorar sus hábitos alimentarios y aumentar su actividad física, además de crear un ambiente de apoyo y motivación para fomentar su compromiso y lograr el éxito.

1.3. Plan de trabajo

Para la realización de este proyecto, se ha decidido utilizar la metodología Scrum, la cual se basa en un enfoque ágil para la gestión y desarrollo de proyectos de software.

Scrum se compone de ciclos de trabajo o sprints, en los que se planifica, diseña, desarrolla, prueba y se presenta el software en incrementos funcionales. En cada sprint, se seleccionan las tareas que se deben realizar y se asignan a los miembros del equipo. Durante las reuniones diarias, se evalúa el progreso y se realizan ajustes si es necesario. Al final del sprint, se realiza una revisión del trabajo realizado y se planifica el siguiente sprint.

El equipo de trabajo se organiza en roles bien definidos, como el Product Owner, el Scrum Master y los desarrolladores. El Product Owner es el encargado de definir los requisitos del proyecto y priorizar las tareas a realizar, en este caso, ha sido Mercedes Martínez. El Scrum Master es el facilitador del proceso, asegurando que se sigan los principios de Scrum y eliminando obstáculos para el equipo, en este caso ha sido la tutora del trabajo de fin de grado, Sonia Estévez. Los desarrolladores son

los encargados de realizar el trabajo y garantizar que se cumplan los objetivos del sprint, que hemos sido Almudena Naharro, Inés Hernández y Patricia Plata.

Para el seguimiento del proceso del equipo, hemos realizado un documento compartido en el que hemos ido asignando las tareas a realizar con un plazo determinado. Además de reuniones con habitualidad para asegurar el progreso, revisión y retrospectiva al final de cada sprint.

Se espera que el uso de la metodología Scrum permita al equipo ser más eficiente y flexible en el desarrollo del proyecto, al tiempo que se asegura la satisfacción del cliente y la calidad del producto final.

Además, para llevar a cabo este trabajo, nos hemos basado en diferentes asignaturas del plan de estudios, como Ingeniería del Software (IS), Aplicaciones Web (AW), Diseño de Sistemas de Interactivos (DSI), Minería de Datos y Paradigma Big Data (MIN), Tecnologías de Programación (TP) y Bases de Datos (BD). Estas asignaturas nos han proporcionado las habilidades y conocimientos necesarios para desarrollar un proyecto de calidad, siguiendo buenas prácticas y utilizando las tecnologías más adecuadas.

Hemos diseñado un diagrama de Gantt (Figura 1.1), para mostrar gráficamente cómo nos hemos organizado y cómo hemos ido construyendo esta aplicación web desde cero. Dándole mucha importancia a la planificación y el diseño.

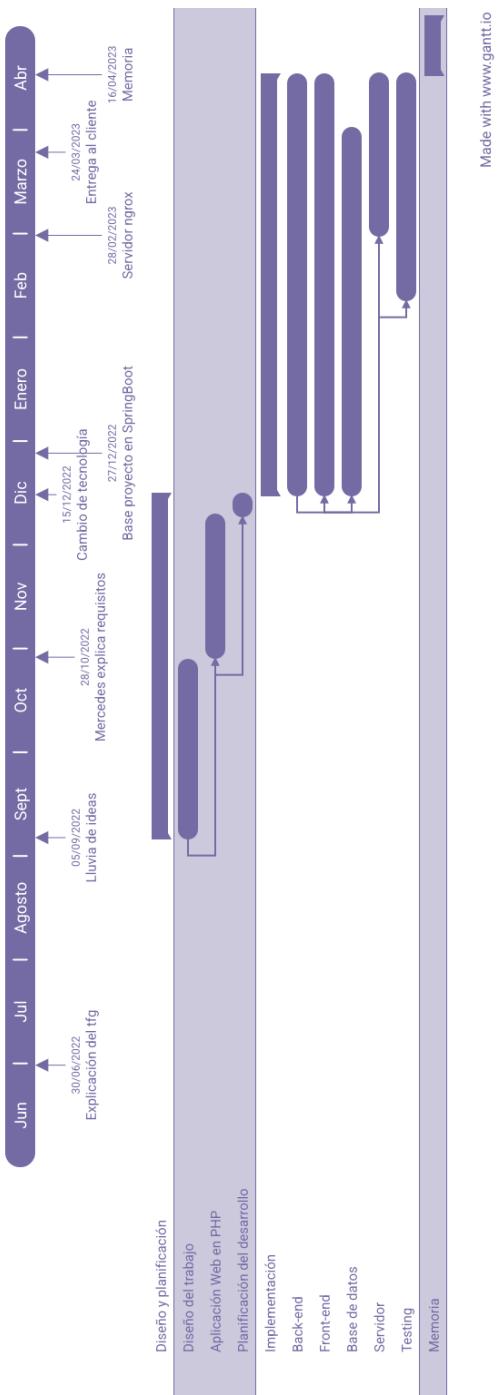


Figura 1.1: Diagrama de Gantt

Como se puede apreciar en el diagrama de Gantt, durante la etapa de diseño y planificación del proyecto, se presentó la necesidad de cambiar de tecnología. Inicialmente, se estaba trabajando en una página web utilizando PHP. No obstante, debido a la complejidad de los requerimientos a implementar, se tomó la decisión de cambiar a Java, ya que cuenta con diversas herramientas y frameworks, como SpringBoot, que facilitaron el desarrollo del sitio web.

Finalmente, nos centramos en documentar todo el código haciendo esta memoria. Para ello la hemos estructurado siguiendo los siguientes epígrafes:

1. Introducción: en esta sección se incluye un resumen de la finalidad de este trabajo en sí y los objetivos que se pretenden cumplir. Se explica la iniciativa de esta aplicación web y porqué surge, así como lo que se pretende conseguir con la realización de este Trabajo de Fin de Grado. Además, hemos incluido un diagrama de Gantt donde se describen las tareas en un marco temporal.
2. Estado del Arte: en este apartado se ha hecho una investigación de mercado para identificar las empresas, y aplicaciones web, que cubren funcionalidades similares a las de este proyecto.
3. Tecnologías utilizadas: en este epígrafe hemos detallado todas las tecnologías que hemos aprendido y utilizado para implementar el Trabajo de Fin de Grado.
4. Diseño gráfico de la aplicación: en este epígrafe se detallan los asuntos que implican el diseño frontend de la aplicación, es decir, el logo, los colores usados, el estilo, y los distintos bocetos que se han realizado para llevar a cabo la aplicación.
5. Especificación de requisitos: en esta sección se especificarán con detalle los requisitos necesarios para implementar las funcionalidades objetivo. Para ello, utilizaremos varios diagramas de casos de uso.
6. Diseño funcional de la aplicación: en este apartado se describe con detalle la lógica de la aplicación web y su arquitectura. Más específicamente, se habla de la base de datos, de la arquitectura, de las entidades y del código responsive.
7. Implementación de la aplicación: en esta sección se explica el diseño tanto front-end como back-end de la aplicación. Así como aspectos referentes al framework utilizado y el modelo elegido, ayudándonos de varias gráficas e imágenes para mostrar la estructura.
8. Resultados y discusión crítica: en este apartado se concretan los resultados obtenidos una vez acabado el código. Además, se discute si se ha cumplido con los objetivos establecidos.
9. Conclusiones y trabajo futuro: en este epígrafe se hace una reflexión sobre los resultados obtenidos y cómo se puede mejorar o ampliar de cara a un futuro.
10. Contribuciones personales: en esta sección se detallan las aportaciones de cada una de las desarrolladoras.
11. Bibliografía: en este apartado se muestran las distintas fuentes y referencias que nos han servido para completar este Trabajo de Fin de Grado.

Capítulo 2

Estado del Arte

En el proceso de desarrollo de esta aplicación web, se ha llevado a cabo una investigación exhaustiva del mercado para identificar las empresas y aplicaciones web que ofrecen funcionalidades similares a las de nuestro proyecto. En el ámbito de la salud y la alimentación, hemos identificado una amplia variedad de soluciones tecnológicas diseñadas para ayudar a las personas a mejorar sus hábitos alimenticios y alcanzar sus objetivos de bienestar personal.

En particular, se han encontrado varias aplicaciones web que comparten algunos de los objetivos de nuestra aplicación web, como ayudar a los usuarios a establecer objetivos, realizar un seguimiento de su progreso y comunicarse con expertos en el área de la salud. Entre las aplicaciones más destacadas se encuentran "MyFitnessPal", "Fitbit", "Noom", "Lifesum" y "Fooducate".

Cada una de estas aplicaciones ofrece una combinación única de características y funcionalidades para sus usuarios, desde la monitorización de la actividad física y la ingesta de alimentos, hasta la conexión con entrenadores personales y dietistas. Además, estas aplicaciones también cuentan con interfaces de usuario intuitivas y atractivas, que ofrecen una experiencia de usuario satisfactoria y un alto grado de compromiso por parte del usuario.

En conclusión, nuestra investigación ha permitido identificar varias soluciones tecnológicas relevantes y exitosas en el mercado. Sin embargo, nuestra aplicación se diferencia por la combinación de características y funcionalidades específicas que ofrece a sus usuarios, especialmente el seguimiento personalizado a través de un coordinador, lo que la convierte en una herramienta única y valiosa para aquellos que buscan mejorar su bienestar personal y alcanzar sus objetivos de salud.

Capítulo 3

Tecnologías utilizadas

3.1. Front-end (lado del cliente)

3.1.1. HTML

HTML (Hypertext Markup Language) es un lenguaje que sirve para crear el contenido de la aplicación web. Está formado por un conjunto de etiquetas que tienen el propósito de definir texto, imágenes, listados, tablas, formularios y otros elementos.

3.1.2. JavaScript

JavaScript es un lenguaje de programación orientado a objetos, que permite implementar funciones complejas de actualización dinámica para poder hacer una página web interactiva, mejorando la experiencia del usuario. En Estepper se ha utilizado para actualizar las gráficas del progreso, entre otros muchos ejemplos.

3.1.3. CSS y Bootstrap

CSS (Cascading Style Sheets) es un lenguaje que se usa para realizar el diseño y el estilo de documentos HTML, XML y XHTML. Permite establecer cómo se muestra el contenido en una aplicación web con elementos visuales como los colores, el tipo de letra, los tamaños, entre otros.

A partir de este lenguaje se pueden definir estilos específicos para diferentes dispositivos y pantallas, consiguiendo así una aplicación responsive.

Por otro lado, Bootstrap es un framework de desarrollo web gratuito de código abierto para desarrollar aplicaciones que se adapten a cualquier dispositivo. Cuenta con un conjunto de herramientas y componentes predefinidas basadas en HTML, CSS y JavaScript. En la aplicación Estepper se ha utilizado en casos muy específicos, ya que la gran mayoría ha sido desarrollado por las autoras.

3.1.4. Thymeleaf

Thymeleaf es un motor de plantillas para aplicaciones web desarrolladas con el lenguaje de programación Java. El objetivo principal es permitir la creación de

plantillas HTML con la utilización de atributos propios de Thymeleaf, facilitando la integración de datos dinámicos.

3.2. Back-end (lado del servidor)

3.2.1. Xampp

Xampp es un paquete de software libre y de código abierto que se utiliza para crear y administrar servidores web locales en un entorno de desarrollo. El nombre es un acrónimo compuesto por las iniciales de los programas que forma: el servidor web HTTP Apache, los sistemas relacionales de bases de datos MySQL y MariaDB, y los lenguajes de programación PHP y Perl.

Para gestionar la base de datos de MariaDB basada en SQL se utiliza PhpMyAdmin, una interfaz web donde se realiza la creación, eliminación y modificación de las tablas, campos y registros.

3.2.2. Java

Java es un lenguaje de programación de alto nivel, orientado a objetos y estático. Se utiliza para el desarrollo de aplicaciones web, aplicaciones móviles, juegos, entre otros. Incluye una gran cantidad de bibliotecas estándar y herramientas de desarrollo que facilitan el desarrollo y el mantenimiento de aplicaciones Java.

3.2.3. Spring Boot

Spring Boot es una extensión del framework de Spring basado en Java que permite crear aplicaciones web y servicios RestFul de manera sencilla. Elimina gran parte de la configuración manual, proporciona configuraciones predeterminadas para muchas bibliotecas y frameworks populares, tiene un servidor de aplicaciones integrado (Tomcat) y se integra con la arquitectura de microservicios.

1. Hibernate

Hibernate es una herramienta basada en el patrón de diseño ORM (Object-Relational Mapping) para Java, que permite el acceso a bases de datos relacionales de manera sencilla. Proporciona una capa de abstracción entre la aplicación y la base de datos, lo que facilita la creación de aplicaciones independientes de la plataforma y evita la necesidad de escribir código de acceso a la base de datos manualmente. También incluye optimización de consultas, soporte para las transacciones e implementa la especificación de JPA (Java Persistence API).

2. JPA

JPA proporciona una API para realizar consultas CRUD (Create, Read, Update, Delete) en las entidades y simplifica el proceso de persistencia de datos mediante la creación automática de objetos Java a partir de objetos Java en lugar de tablas de bases de datos.

3. Maven

Maven es una herramienta de gestión de proyectos de software para Java, que se utiliza para la gestión de dependencias como herramienta de compilación y depuración. Se utiliza un archivo propio POM (Project Object Model) llamado pom.xml en formato XML, que contiene la información necesaria del proyecto y las dependencias (librerías externas). Tiene todo lo necesario para generar el fichero ejecutable de la aplicación.

Además, Maven es la herramienta que utiliza Spring Boot por defecto.

3.2.4. Jython

Jython es un lenguaje de programación de alto nivel y orientado a objetos que combina las características de Python y Java. Es una implementación de Python diseñado para ejecutarse en la plataforma Java Virtual Machine (JVM) y que puede utilizar todas las bibliotecas y módulos de Java. Se ha utilizado para poder usar Python en el proyecto.

3.2.5. Gmail API

Gmail API es una interfaz de programación de aplicaciones (API) desarrollada por Google que permite a los desarrolladores interactuar con las cuentas de Gmail de sus usuarios. Se utiliza para integrar la funcionalidad de Gmail en las aplicaciones, permitiendo la capacidad de leer y enviar correos electrónicos desde una aplicación de terceros. En este caso se ha utilizado para enviar un correo en caso de que el usuario haya olvidado su código de acceso o también para enviar un correo por necesidad de realizar una analítica.

3.3. Bibliotecas

1. FontAwesome

FontAwesome es una galería de iconos vectoriales y estilos CSS. Los iconos se pueden personalizar fácilmente con hojas de estilo CSS, permitiendo al desarrollador cambiar el tamaño, el color y otros aspectos.

2. Chart.js

Chart.js es una biblioteca gratuita de JavaScript de código abierto para la visualización de datos en gráficos interactivos. La biblioteca cuenta con una variedad de tipos de gráficos, como gráficos de barras, de líneas, circular, de pastel, entre otros. Chart.js se basa en una tecnología HTML5 Canvas con muchas opciones de configuración, siendo de las bibliotecas más populares para crear gráficos.

3. jQuery

jQuery es una biblioteca de JavaScript que permite simplificar y acelerar la creación de páginas web interactivas y dinámicas. Ofrece una gran cantidad

de funciones y métodos que posibilitan el manejo de eventos, la manipulación del estilo y la comunicación con el servidor de forma más sencilla y rápida.

4. sweetAlert2

SweetAlert2 es una biblioteca de código abierto que se utiliza para crear ventanas emergentes personalizables en páginas web. Se basa en las tecnologías de JavaScript y jQuery con opciones de configuración muy sencillas.

5. jaydebeapi

jaydebeapi es una biblioteca de Python que permite a los desarrolladores conectarse y ejecutar consultas en bases de datos comunes utilizando el JDBC (Java Database Connectivity) desde Python.

Para utilizar esta biblioteca es necesario instalar y realizar la configuración del controlador JDBC de la base de datos correspondiente.

3.4. Aplicaciones

1. Visual Studio Code

Visual Studio Code es un editor de código fuente desarrollado por Microsoft que está disponible para Windows, Linux y MacOS. Es un software gratuito que se utiliza para desarrollar y depurar aplicaciones web y de escritorio. Incluye una gran variedad de herramientas para la edición del código. Además, es posible utilizar una amplia gama de extensiones creadas por la comunidad para extender las funcionalidades que ofrece Visual Studio Code. En este caso ha sido necesario descargar la extensión "Spring Boot Dashboard" para poder ejecutar la aplicación, en otras muchas.

2. GitHub y Git

GitHub es una plataforma en línea que fue creada para que los desarrolladores puedan alojar, compartir y colaborar en proyectos de software, como aplicaciones web, utilizando el sistema de control de versiones Git. En este caso se ha utilizado un repositorio de GitHub, realizando la conexión desde Visual Studio Code, para almacenar todo el código fuente de la aplicación web.

3. Latex

TeX es un lenguaje de programación y sistema de composición de textos que se utiliza principalmente para producir documentos escritos con alta calidad tipográfica, como libros o artículos académicos. Los usuarios de TeX suelen utilizar LaTeX, que es un conjunto de macros para TeX que facilita la creación de documentos con una estructura compleja. LaTeX proporciona una amplia variedad de estilos y plantillas predefinidas para poder producir documentos bien estructurados de forma sencilla y de alta calidad. Estas tecnologías se han utilizado para desarrollar la memoria de la aplicación.

4. UMLet

UMLet es una herramienta de código abierto basado en Java que sirve para dibujar diagramas UML (Lenguaje de Modelado Unificado). Se basa en una interfaz gráfica de usuario intuitiva que facilita la creación de diagramas de casos de uso, diagrama de clases y otros diagramas.

5. Google Drawings

Google Drawings es una herramienta de dibujo en línea que forma parte del conjunto gratuito de aplicaciones de Google Drive. Permite crear gráficos, diseños y otros dibujos de forma sencilla y en tiempo real con otros usuarios. Los dibujos se pueden exportar en formato PDF, PNG y SVG. Se ha utilizado para dibujar los diagramas de actividad.

3.5. Instalación del servidor

Para poder realizar el testeo de la aplicación web por parte del cliente, se ha instalado un servidor a partir de un ordenador ubicado en la Facultad de Informática de la Universidad Complutense de Madrid.

Como la aplicación utiliza la tecnología de Spring Boot, es necesario que el código este ejecutándose continuamente desde Visual Studio Code, por eso se utiliza el ordenador.

Para llevar a cabo la instalación hay que descargar Maven, Java JDK, Xampp, Git, Visual Studio Code, Global Protect y ngrok.

1. Global Protect

Es una VPN (servicio de web privada), que permite a los usuarios conectarse de forma segura a la red de una empresa, en este caso la UCM, desde ubicaciones remotas fuera de la red corporativa. Es una forma de conectarse a una red evitando ataques cibernéticos y violaciones de datos. Como la red del ordenador tiene asignada una IP (Internet Protocol) de la UCM, es necesario conectarse a través de la VPN correspondiente.

2. ngrok

Es una herramienta que permite exponer un servidor local en internet de manera temporal, haciendo posible el acceso desde cualquier lugar. El servidor se expone en una url pública HTTPS, por lo que utiliza el protocolo criptográfico SSL para cifrar los datos que se transmiten desde el servidor y el navegador web.

Los pasos a seguir para llevar a cabo la instalación son los siguientes:

1. Conectarse al ordenador ubicado en la facultad a través de Global Protect y el escritorio remoto
2. Instalar las herramientas Maven, Java JDK, Xampp, Git, Visual Studio Code, Global Protect y ngrok

3. Crear la base de datos en PhpMyAdmin
4. Realizar la clonación del repositorio GitHub en Visual Studio Code
5. Ejecutar la aplicación con Spring Boot Dashboard desde Visual Studio Code
6. Abrir una terminal y navegar hasta donde se ha instalado ngrok
7. Ejecutar el comando `./ngrok http [puerto]`, donde el puerto es el número de puerto local que se desea exponer a través de ngrok
8. Tras la ejecución, se crea una URL pública HTTPS lista para conectarse

Capítulo 4

Diseño gráfico de la aplicación

4.1. Nombre

A la hora de decidir el nombre que tendría la aplicación implementada, se realizó la técnica conocida como lluvia de ideas o brainstorming.

Para desarrollar esta metodología, los miembros del grupo se reunieron en un ambiente relajado y establecieron una serie de pautas. De esta manera, se fijó un período de 10 minutos en el que cada componente del equipo debía apuntar la mayor cantidad de sugerencias posibles sobre un papel. Posteriormente, se hizo una puesta en común en un documento de word, se combinaron ideas y se inició el proceso de selección. Tras evaluar distintas alternativas, se observó cómo la mayoría de los nombres propuestos giraban en torno a una idea central: el bienestar del cuerpo y la mente de los individuos y su capacidad de superarse. También se contemplaron aspectos como la brevedad del nombre, el idioma, la facilidad de pronunciación y recuerdo, el mensaje que debía sugerir, etc. Algunas de las opciones que destacaron fueron ByB (BeYourBest), StayFit o ZAS. Sin embargo, finalmente se optó por Estepper.

El origen de la palabra Estepper se encuentra relacionado con diversos significados. En primer lugar, se asocia con una máquina escaladora o stepper, que es un aparato de musculación disponible en la mayoría de gimnasios que trata de reproducir el movimiento que se lleva a cabo al subir una escalera. Además, una particularidad de esta máquina es que tiene un tamaño reducido y compacto que permite que sea fácil de transportar y de guardar, por lo que son muchas las personas que disponen de ella en sus casas. Estos dos conceptos resultan fundamentales. Por un lado, la acción de subir una escalera implica esforzarse por alcanzar un punto más alto, lo que se asemeja al fenómeno de cumplir o alcanzar un objetivo. Por otro lado, el hecho de ser un objeto que se puede llevar a cualquier lugar, refuerza el razonamiento de que no es necesario salir de casa para cuidarse, sino que por el contrario, la mejora de la alimentación y la promoción de la actividad física se logran con una actitud positiva. Además, puesto que la escaladora es una máquina para hacer ejercicio, esto se encuentra estrechamente relacionado con el pensamiento anterior.

En segundo lugar, al extraer de la palabra Estepper el fragmento stepper, se puede apreciar que la traducción directa al inglés es “paso a paso”, lo cual encaja perfectamente con el mensaje que la aplicación intenta transmitir: paso a paso, con esfuerzo

y actitud, el objetivo de un estilo de vida saludable es posible.

Asimismo, la variedad de significados que se asocian a Estepper se ajustan entre ellos dando una visión global a partir de la cual se pueden realizar distintas interpretaciones. Por ejemplo, para unos puede implicar que poco a poco van a alcanzar el peso deseado o para otros que el deporte es un elemento esencial para mantenerse saludable. Otro punto remarcable es que se comprobó a través de internet que el nombre no estuviera registrado. Por todo ello, se ha considerado que Estepper es el nombre más adecuado para la aplicación, pues cumple con los requisitos y promueve un mensaje acertado y coherente con lo que se ha desarrollado.

4.2. Logo

En lo que respecta a la elección del logo, el proceso de decisión fue similar. Se propusieron varios diseños por cada miembro del grupo y se sometieron a votación. Aquí se observó que al haber decidido primero el nombre, todos tenían en común la representación de una escalera. Además, todos los diseños fueron creados utilizando Adobe Photoshop.

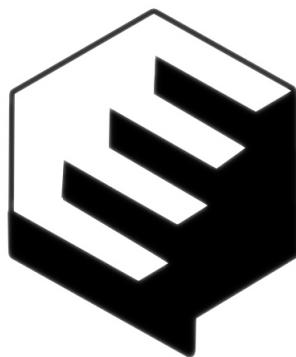


Figura 4.1: Logo de Estepper

Finalmente, se escogió el logo mostrado en la figura 4.1. Es un diseño simple en blanco y negro que al igual que el nombre, evoca la sensación de subir una escalera. Al mismo tiempo, permite combinar sus escalones para formar una E, primera letra del nombre de la aplicación. Por otro lado, estos elementos simbolizan el funcionamiento y mensaje de Estepper, la mejora de la alimentación y promoción de la actividad física que se puede alcanzar subiendo poco a poco los peldaños de la escalera.

4.3. Estilo

De entre los numerosos estilos de diseño utilizados actualmente en el desarrollo de interfaces digitales, en este proyecto se ha optado por un estilo plano con ciertos toques de esqueuomorfismo.

La elección de un estilo plano surge porque desde un principio se tenía claro que la aplicación web debía transmitir armonía entre los elementos a la vez que sencillez, pues iba a ser empleada por personas de distinto rango de edad y, por tanto, resultaba fundamental que el uso de la aplicación fuera cómodo, claro e intuitivo. Para ello,

en todo momento se ha perseguido un estilo que fomentara entre otros aspectos, un alto grado de usabilidad, accesibilidad, eficiencia, coherencia o confianza a la vez que una grata experiencia de usuario.

Desde esta perspectiva, el estilo plano se refleja además de en la sencillez ya mencionada, en el empleo de formas limpias y funcionales. En concreto, en la aplicación abunda el uso de formas geométricas, especialmente cuadrados y rectángulos, y se caracteriza por la ausencia de profundidad en los iconos. La tipografía es clara y simple con mensajes cortos y, como se indicará con más detalle, se utilizan colores planos y desaturados, sin gradientes ni texturas.

Entre las ventajas que ha reportado en el desarrollo del trabajo el empleo de este estilo plano se pueden señalar las siguientes:

1. Al emplear elementos sencillos, resulta fácil adaptar el diseño de aplicación web a vista móvil. Además, dichos elementos se cargan de forma más rápida que si se tratara de gráficos complejos con mayor necesidad de recursos.
2. La apariencia del estilo plano da una sensación moderna y elegante que la mayoría de los usuarios pueden encontrar atrayente.
3. Se trata de un estilo simple que vuelve la aplicación muy intuitiva, de manera que un usuario sabe rápidamente dónde hacer o no hacer click para encontrar lo que busca.

A continuación, se muestran algunos ejemplos de vistas de la aplicación que incluyen los conceptos mencionados.

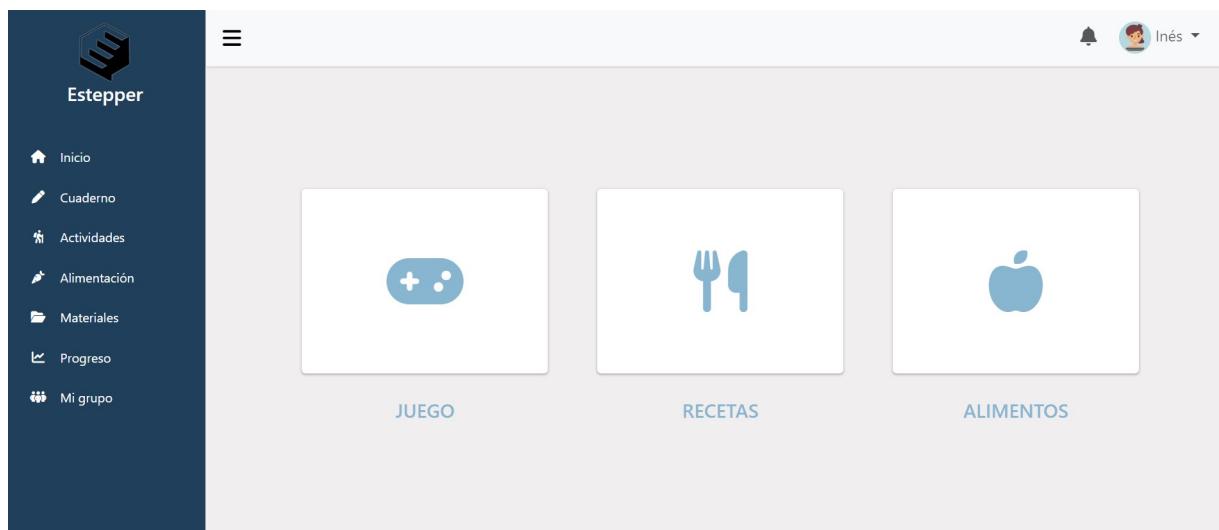


Figura 4.2: Diseño vista principal alimentación

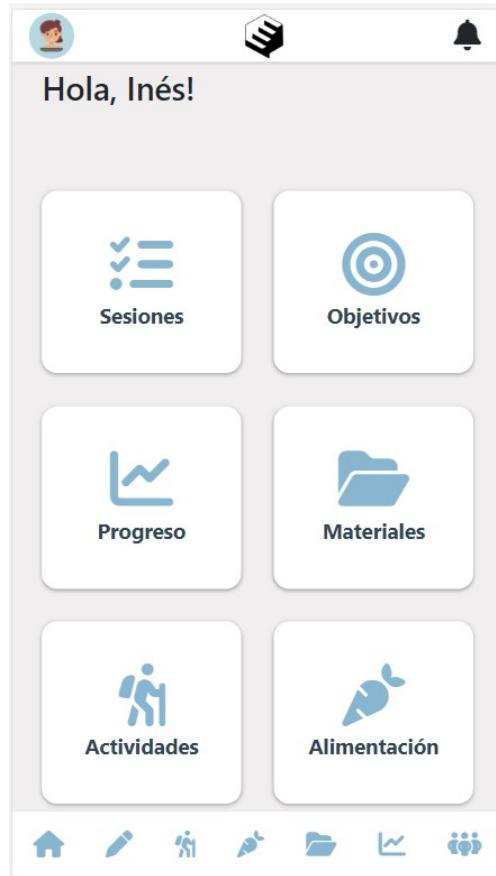


Figura 4.3: Diseño pantalla principal vista móvil de un Participante

Por otro lado, es cierto que aunque el estilo predominante en la aplicación es el diseño plano, algunas secciones presentan cierto esquemomorfismo digital, es decir, elementos que tratan de reproducir los objetos del mundo real que por lo general, son innecesarios desde el punto de vista de la interacción.

Dos ejemplos de esto se dan en la funcionalidad de objetivos. Así, en la zona correspondiente al objetivo relacionado con el consumo diario de agua, a la hora de añadir o eliminar el registro de un vaso, además de los botones, se ve cómo los vasos de agua se llenan, quedando en color negro los vasos que aún no se ha bebido el participante y en azul los consumidos. Por otra parte, en la zona inferior, correspondiente a los objetivos personalizados aparece un apartado que se asemeja a un bloc de notas, con el mismo tono de color y forma, donde se muestran los objetivos para el día seleccionado.



Figura 4.4: Diseño vasos de agua

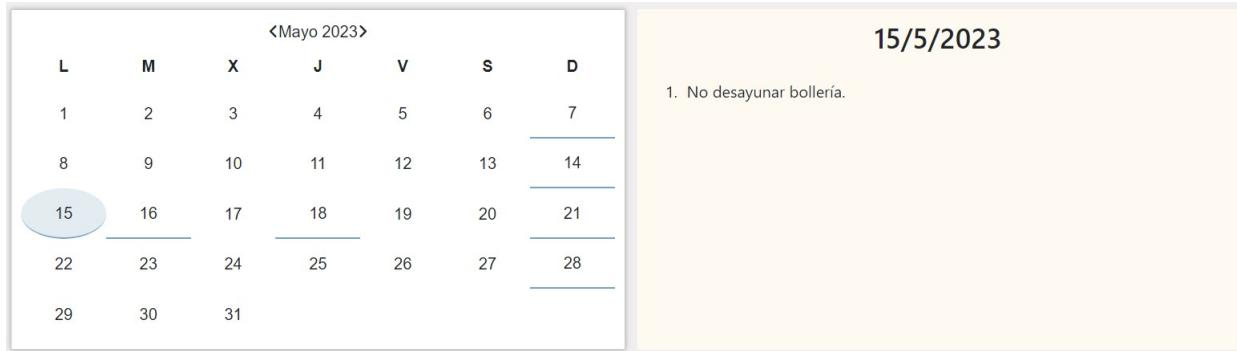


Figura 4.5: Diseño bloc de notas

4.4. Colores

Los colores utilizados en la aplicación, al igual que el estilo, han sido elegidos atendiendo a distintos criterios. Así, la paleta seleccionada es la que se muestra en la Figura 4.6. Se trata de un conjunto de colores planos que se encuentran generalmente asociados con la calma, confianza, descanso, seguridad y tranquilidad. Además de esos colores, también se utilizan otros como tonos de amarillo o de verde, aunque en menor medida y con fines específicos. El amarillo, utilizado sobre todo en la funcionalidad de actividades, se relaciona con la energía y felicidad, por lo que se considera un color estimulante y positivo. En cuanto al verde, color de la esperanza, transmite sensaciones de equilibrio o naturaleza. Por ello, se emplea en algunas zonas como por ejemplo en el objetivo de deportes o en los chats.

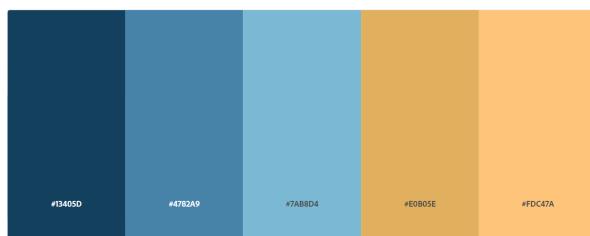


Figura 4.6: Paleta de colores

4.5. Bocetos y prototipos Balsamiq

Antes de comenzar a desarrollar el código de la aplicación, al igual que con otros elementos como el logo, nombre o colores, se realizaron varios diseños de manera individual que más tarde se sometieron a votación.

De esta manera, los bocetos iniciales se implementaron con la aplicación GoodNotes del AppleStore. En la figura 4.7 se pueden ver algunos de estos bocetos. Después de esta fase inicial, se seleccionaron aquellos que mejor podían encajar y se pasó a una segunda fase en la que, a partir de los bocetos, se diseñó de manera incremental un prototipo más aproximado a la realidad. Estos mockups se implementaron con la

herramienta Balsamiq que permite la interacción entre pantallas. En la figura 5.8 se muestran varias de las vistas creadas con Balsamiq. En caso de querer acceder al prototipo de Balsamiq completo, puede hacerse a través de este enlace de drive: https://drive.google.com/file/d/1bLWY4RIfz9Vw4Gap0amEmwjPm0e2_qVd/view?usp=sharing

ACTIVIDADES PROPUESTAS

Fecha	Nombre actividad.	Participantes: 3	<input type="button" value="Borrar"/>
Fecha	Nombre actividad	Participantes: 10	<input type="button" value="Borrar"/>
Fecha	Nombre actividad	Participantes: 15	<input type="button" value="Borrar"/>

+ AÑADIR ACTIVIDAD

GESTIONAR INVITACIONES → pensarlo después, no se si arrojarlo aquí o no

Pie

ACTIVIDADES DISPONIBLES EN NOVIEMBRE

<input type="button" value="Reservar invitación"/>				
--	--	--	--	--

MIS ACTIVIDADES

MIS INVITACIONES

Pie

Stepper

Código:
Contraseña:

¿Has abierto tu código? Puedes [ingresar](#) para iniciar en el campo. Si no sabes si unirte o no a nuestro proyecto, te recomendamos hacer este TEST DE FINESTRICH

ACTIVIDAD : NOMBRE ACTIVIDAD FECHA : _ / _ /

Detalles de la actividad.

Breve descripción

Localización : Calle ___, Madrid

Invitaciones : 2

1 - María Jiménez	<input type="button" value="Borrar"/>
2 - María Jiménez	<input type="button" value="Borrar"/>

+ NUEVA INVITACIÓN

Despliegue: Invitación propia Despliegue individual

EDITAR ACTIVIDAD

Pie

MIS ACTIVIDADES

ACTIVIDADES PRÓXIMAS

Fecha	Nombre actividad.	Participantes: 3
Fecha	Nombre actividad.	Participantes: 10

ACTIVIDADES YA REALIZADAS

Fecha	Nombre actividad.	Participantes: 3
Fecha	Nombre actividad.	Participantes: 10

Pie

Recomendaciones generales para mantenerse saludable

Agua	Deporte	Descanso	Alimentación
•	•	•	•
Leer más	Leer más	Leer más	Leer más

TEST DE FINESTRICH

Práctica obtenida:

¡Estupendos! Te encuentras perfectamente sano, no es necesario que participes en el proyecto, pero aquí tienes nuestras recomendaciones generales para mantener todo nuestro cuerpo.

Felicidades **Nombrelusuario**, ya tienes cuenta en Stepper.
Tu código de usuario es: 079785G
¡No te olvides! Lo necesitarás para poder acceder a la app.

ESTADO DE LA CUENTA : DESACTIVADA

Teléfono de contacto: +34 --- --- ---

Contacta por correo Contacta por teléfono



Figura 4.7: Ejemplos de bocetos

Top Left: Login Page

A Web Page
https://www.estepper.com

Código: _____
Contraseña: _____
Iniciar sesión

¿Has olvidado tu código? Pincha [aquí](#) para recibirlo en el correo.
Si no sabes si unirte o no a nuestro proyecto, te recomendamos hacer este [TEST DE FINDRISC](#)

Bottom Left: Dashboard - Juego para crear tus propias dietas

A Web Page
https://www.estepper.com

Inicio Cuaderno Actividades Alimentación Materiales Progreso Mi grupo

Juego para crear tus propias dietas
Calorías: 2000
Busca un alimento y arrástralo a la casilla:
Lácteo: Lácteo:
pollo manzana lentejas aceite
Lácteo Proteína Verdura Harina Fruta Grasa
Desayuno
Media mañana
Comida
Merienda
Cena
Comprobar

Top Right: Session Details - Sesión 1

A Web Page
https://www.estepper.com

Hola usuario ▾

Inicio Sesión 1

Asistencia: Sí
Motivo: _____
Observaciones: _____

Diapositivas: Diapositiva 1.pdf
Cuestionario: Cuestionario 1

¿Has terminado?

Bottom Right: Diet Tracking

A Web Page
https://www.estepper.com

Hola usuario ▾

Inicio ¿Qué has comido hoy?

Proteínas	Fibra	Sal	Hidratos de carbono	Grasas saturadas
Alimento: Manzana	Raciones: 2	Fecha: 11:35	<input type="checkbox"/>	

¿Has comido algo más? Añádelo aquí:

¿No ves tu alimento aquí? [Créalo aquí!](#)

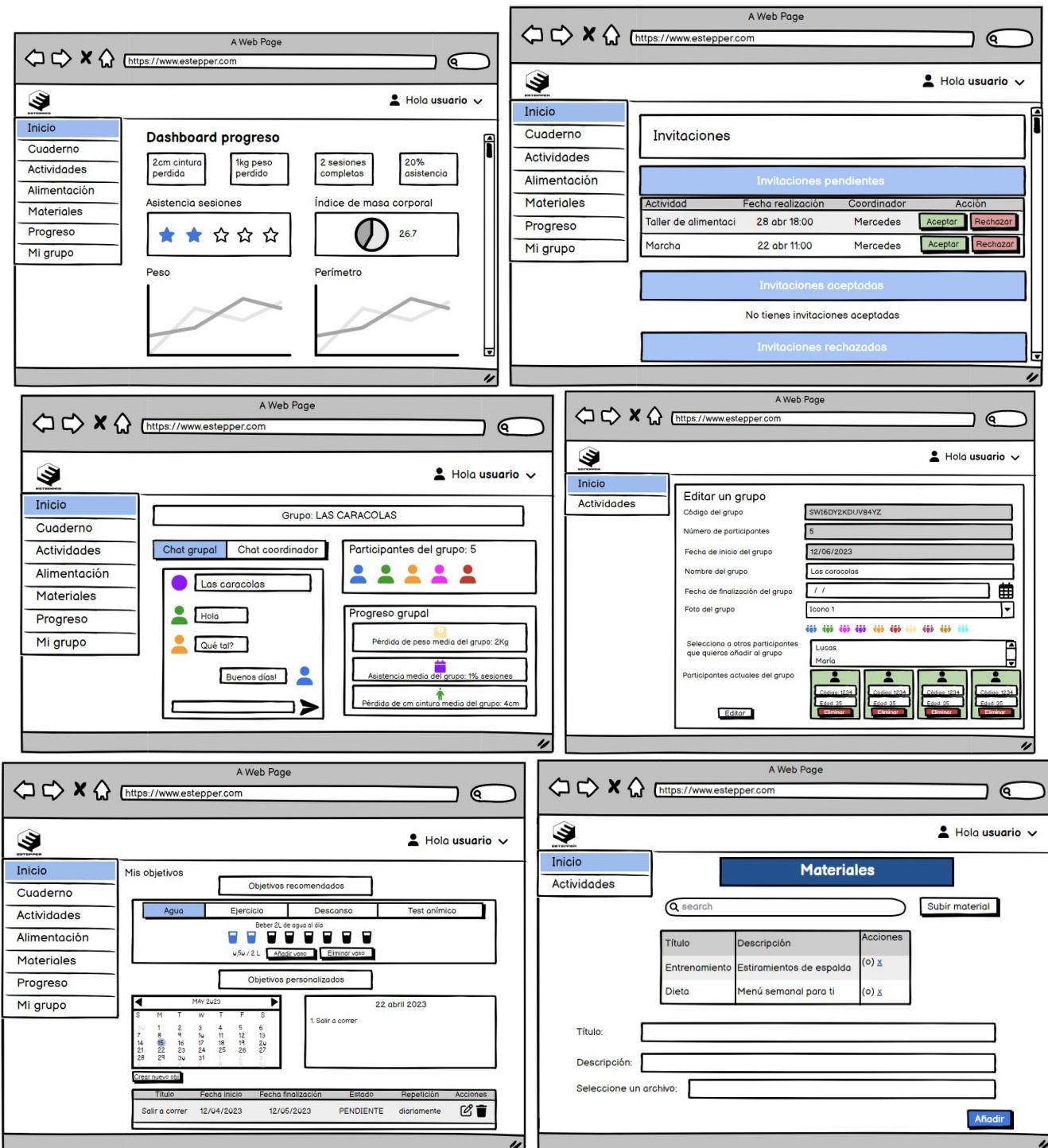


Figura 4.8: Ejemplos del prototipo Balsamiq

Durante el desarrollo del proyecto, este prototipo ha demostrado ser de gran utilidad. Como ventaja principal, fue útil para tener una idea general muy completa sobre lo que se iba a hacer, los requisitos, vistas, estructura, contenido, elementos nuevos que se debían considerar, etc. Además, en todo momento ha sido clave a la hora de organizar el trabajo y partir de una idea predefinida y consensuada.

4.6. Principios de diseño

Esta sección se centra en describir los principios de diseño que se han llevado a cabo en el transcurso de implementar la aplicación.

1. Principio de proximidad

El principio de proximidad explica que el cerebro relaciona aquellos elementos que se encuentran próximos entre sí. En el proyecto se hace uso de este principio en diversas pantallas. Por ejemplo, en tablas como las de participantes o grupos se incluye una columna de acciones compuesta por diversos botones que, por su localización, se puede intuir que actúan sobre el mismo usuario o grupo.

Listado de participantes					
	Alias	Código	Email	Estado	Acciones
	Patricia	172	pplata@ucm.es	ALTA	
	Almudena	52654	almunaha@ucm.es	ALTA	
	Placi	65412	placi@gmail.com	BAJA	
	Inés	44960	ines@gmail.com	ALTA	

Figura 4.9: Vista del listado de participantes

2. Principio de cierre

En el mundo del diseño, el principio de cierre hace referencia a la costumbre que tenemos los individuos de buscar un elemento simple cuando observamos un conjunto complejo de elementos.

3. Principio de consistencia interna y externa

Cuando los usuarios entran en contacto con una interfaz nueva, lo habitual es que aprendan más fácilmente a utilizarla si los conceptos son consistentes con aquello que ya conocen. De este modo, dentro de este principio aparecen dos tipologías, la consistencia interna y la externa. La consistencia interna indica que los controles que tengan una funcionalidad similar deberían tener también una apariencia similar. En cuanto a la externa, implica que la consistencia se da no solo dentro del entorno de la aplicación sino dentro de un sistema específico o entorno más general.

Los ejemplos más claros de estos principios en la implementación de Estepper, son las páginas de login y registro. Los controles internos de ambas vistas tienen las mismas funcionalidades y apariencia, lo que garantiza consistencia interna. Además, las vistas son similares a las de otras aplicaciones web, por lo que cuando un usuario entra por primera vez, no tendrá ningún problema ya que conoce cómo funcionan estos formularios con los que ya se encuentra muy familiarizado. Esto último supone el cumplimiento de consistencia externa.

Este stepper es una interfaz de usuario para iniciar sesión. Se divide en tres secciones principales:

- Entrada:** Contiene campos para "Código" (con icono de persona) y "Contraseña" (con icono de candado). Una botón azul "Iniciar sesión" se encuentra debajo de los campos.
- Mensajes:** Un cuadro que dice: "¿Has olvidado tu código o contraseña? Pincha aquí para recibirlo en el correo." y "Si no sabes si unirte o no a nuestro proyecto, te recomendamos hacer el TEST DE FINDRISC."
- Botones:** Un cuadro que incluye un enlace "Pincha aquí para iniciar sesión".

Figura 4.10: Vista de la página de login

Este stepper es una interfaz de usuario para registrarse. Se divide en tres secciones principales:

- Entrada:** Contiene campos para "Alias/Nickname" (con icono de persona), "Correo electrónico" (con icono de envelope) y "Contraseña" (con icono de candado). Hay un campo para "Confirmar tu contraseña" (con icono de candado).
- Aceptación:** Un cuadro que dice: "Acepto los términos y condiciones y autorizo el uso de mis datos personales." con un checkbox.
- Botones:** Un cuadro que incluye un botón azul "Registrarse" y un enlace "Pincha aquí para iniciar sesión".

Figura 4.11: Vista de la página de registro

4. Principio de visibilidad y feedback

Este principio se encuentra relacionado con los recursos de la interfaz que transmiten las acciones posibles o recomendables así como el estado actual de la aplicación o una retroalimentación sobre la ejecución de una tarea en un momento determinado. Como ejemplo ilustrativo de esto, podemos encontrar la vista de la figura 4.12, en la que el usuario sabe rápidamente que ya se encuentra logueado y que ya puede escribir en la zona correspondiente del chat.



Figura 4.12: Vista móvil del chat de ayuda con el administrador

Dentro de este punto, se considera relevante saber dirigir la atención a la hora de realizar el feedback. Esto puede hacerse mediante tipografía, opacidad de elementos, tonos de los colores, relieve, restricciones culturales, etc. Así, en la figura 4.13 se observa una diferencia entre el botón de mensajes de dos usuarios, uno dado de alta y con el que ya se puede interactuar, y uno con estado de baja con la acción de intercambiar mensajes desactivada. De la misma manera, en otra de las imágenes vemos como el color rojo simboliza la existencia de un error, en este caso, escribir una dirección de correo incompleta. Un tercer ejemplo del feedback recibido se da en acciones como eliminar una actividad, material, objetivo, grupo u otros elementos de la aplicación en las que salta un modal de alerta para confirmar dicha acción.



Figura 4.13: Ejemplos del principio de feedback en la aplicación

5. Principio de Gestión del estado visible

Para asegurar que el usuario recuerde en qué parte de un proceso se encuentra o en qué zona de la aplicación está se ha seguido un diseño riguroso prestando especial atención a la disposición de los elementos y los tonos de color utilizados. Así, si un coordinador quiere acceder a la información de una actividad creada puede acceder a los detalles de la misma con facilidad.

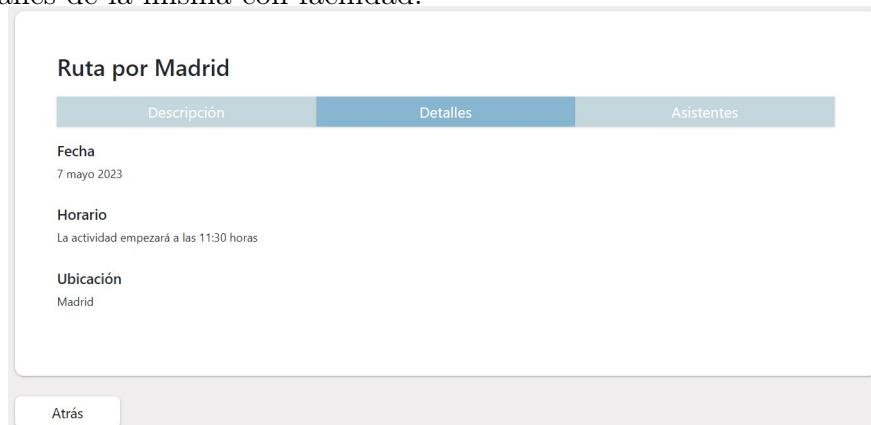


Figura 4.14: Acceso a los detalles de una actividad

6. Principio de libertad y control del usuario

Para garantizar la libertad y control del usuario y proporcionarle una experiencia dinámica en la que pueda interaccionar con comodidad, se ha tratado de no interrumpir acciones del mismo. En lugar de ello, se le sugieren ciertas operaciones que le hacen sentir el control. De este modo, en el caso particular de un participante del proyecto que quiere editar su perfil, se le otorga el control sobre el proceso, por ejemplo, permitiéndole abandonarlo en cualquier momento pinchando en el botón de cancelar.

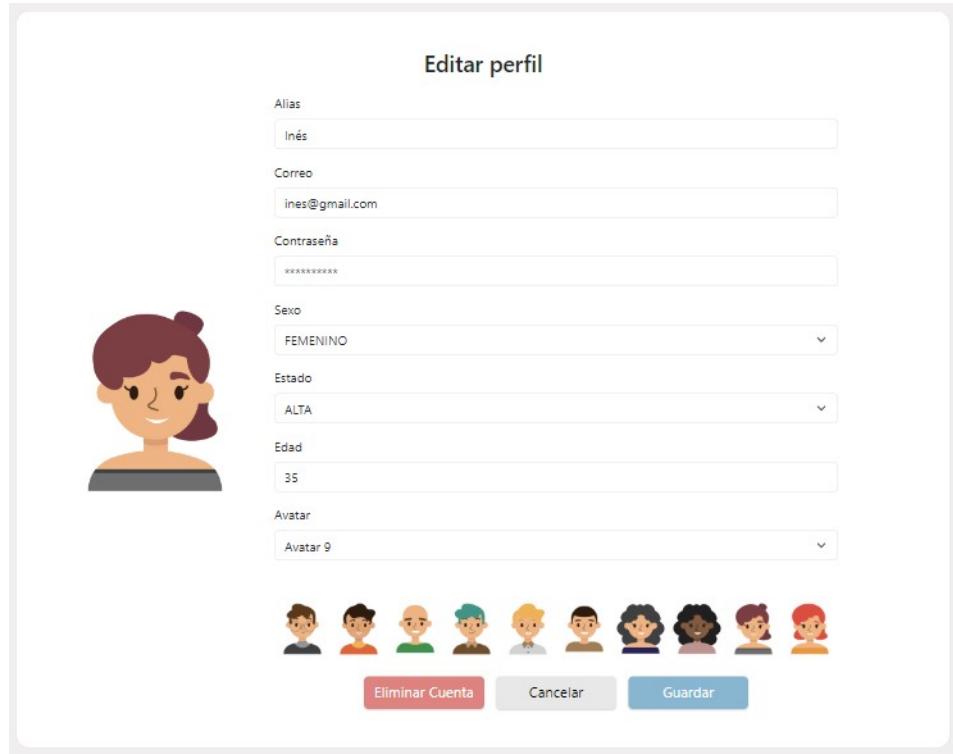


Figura 4.15: Editar perfil del participante

Capítulo 5

Diseño funcional de la aplicación

5.1. Arquitectura de la aplicación

En esta sección se describe mediante un diagrama la arquitectura software que se ha desarrollado en la aplicación.

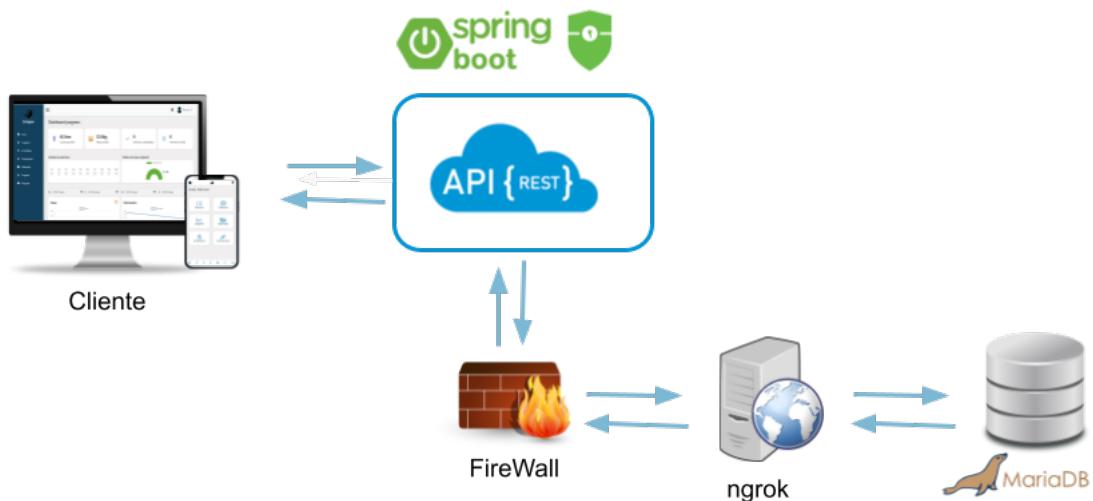


Figura 5.1: Arquitectura de la aplicación

La arquitectura de esta aplicación, como se indica en la Figura 5.1, se basa en una comunicación cliente-servidor a través de una API REST implementada en Spring Boot.

El cliente, que puede ser una aplicación web o móvil (frontend), se conecta a la API para enviar y recibir datos. Además, Spring Security, antes de que el usuario inicie

en la aplicación, verifica su identidad y autentica sus credenciales. El tráfico entre el cliente y el servidor atraviesa un firewall que controla el acceso y la seguridad de la red.

El servidor ngrok actúa como un túnel seguro que permite exponer la aplicación local al público a través de Internet, haciendo que sea accesible desde cualquier lugar del mundo. Por último, la base de datos mariaDB, interactúa con el servidor ngrok para almacenar los datos necesarios para el funcionamiento de la aplicación.

5.2. Base de datos

Todas las entidades tienen un atributo id que les identifica, además hemos usando la anotación @GeneratedValue que se utiliza para generar automáticamente el valor de la clave primaria. En todos los casos hemos usado la estrategia GenerationType.IDENTITY, que indica que se usará la funcionalidad de autonumeración proporcionada por la base de datos subyacente. Es decir, que cada vez que se inserta un nuevo registro en la tabla correspondiente a esta entidad, la base de datos genera automáticamente un valor único para la columna de la clave primaria.

Además, hemos usado varios tipos de relaciones en nuestra base de datos. La de tipo many-to-any, que se utiliza cuando la entidad puede estar relacionada con múltiples tipos de entidades diferentes. La de tipo many-to-one, que se utiliza cuando varias entidades de un tipo están relacionadas con una única entidad de otro tipo. La de one-to-one, que se utiliza cuando una entidad está relacionada con una única entidad de otro tipo y viceversa. La de one-to-many, que se utiliza cuando una entidad está relacionada con múltiples entidades de otro tipo. Y, por último, la de many-to-many, que se utiliza cuando varias entidades de un tipo están relacionadas con múltiples entidades de otro tipo y viceversa.

En el caso de la entidad actividades, mantiene una relación many-to-any con participantes. Para representar esta relación many-to-any en una base de datos relacional, es necesario crear una tabla intermedia que permita establecer las relaciones entre las entidades. En este caso, la tabla intermedia se llama ".^asistencia_actividades". De esta forma, se transforma la relación many-to-any y ahora la relación se encuentra en la tabla intermedia y es de many-to-one con la tabla de participantes y con la tabla de actividades. La entidad de alimentosconsumidos mantiene una relación de many-to-one con participantes y de many-to-one con alimentos. La entidad de fasevaloracion mantiene una relación de tipo many-to-one con la tabla de participantes. Además, hemos implementado herencia siguiendo la estrategia de tipo "joined", es decir, cada entidad de la jerarquía tiene su propia tabla, que contiene los atributos específicos de esa entidad, y una clave foránea que hace referencia a la tabla de la entidad raíz. La tabla de la entidad raíz, a su vez, tiene una clave primaria compartida con las tablas de las entidades hijas. Cada tabla de entidad tiene una relación one-to-one con la tabla de la entidad raíz. Así que, en este caso, las tablas de exploracion, antecedentes, alimentacionval, actfisica, findrisc y clasificacion mantienen una relación one-to-one con la tabla de fase de valoración. La entidad fichas mantiene una relación de tipo many-to-one con la tabla de participantes. Además, se usa la estrategia de herencia joined por lo que las entidades fichaeleccion, fichaob-

jetivo y fichataller mantienen una relación de one-to-one con fichas. Por otro lado, la entidad grupo tiene una relación de many-to-one con la tabla de coordinadores, y de one-to-many con la tabla de participantes. La entidad de invitación mantiene una relación many-to-one con actividad, participante y coordinador. La entidad de materiales mantiene una relación many-to-one con grupo y participante. La entidad mensaje tiene una relación many-to-one con grupo y usuario. Mensajeadmin una relación many-to-one con usuario y emisor. Mensajeprivado mantiene una relación many-to-one con coordinador, participante y usuario. La entidad de notificación mantiene una relación many-to-one con participantes. Al igual que la entidad objetivo, objetivo_descanso, objetivo_agua, objetivo_estado_animo y objetivo_ejercicio. La entidad observaciones mantiene una relación many-to-one con grupo y con coordinador. Lo mismo pasa con la entidad participantes. La entidad progreso tiene una relación de tipo many-to-one con participantes. La entidad recetas mantiene una relación many-to-many con la tabla alimentacion, pasa lo mismo que con la entidad de actividades, se necesita crear una tabla intermedia. En este caso se crea la tabla receta_alimentacion que mantiene una relación many-to-one con recetas y con alimentación. La entidad sesión mantiene una relación many-to-one con participantes. Por último, en la entidad usuario hemos implementado herencia siguiendo la estrategia de tipo "joined", por lo que las entidades coordinador, administrador y participante mantienen una relación one-to-one con usuario.

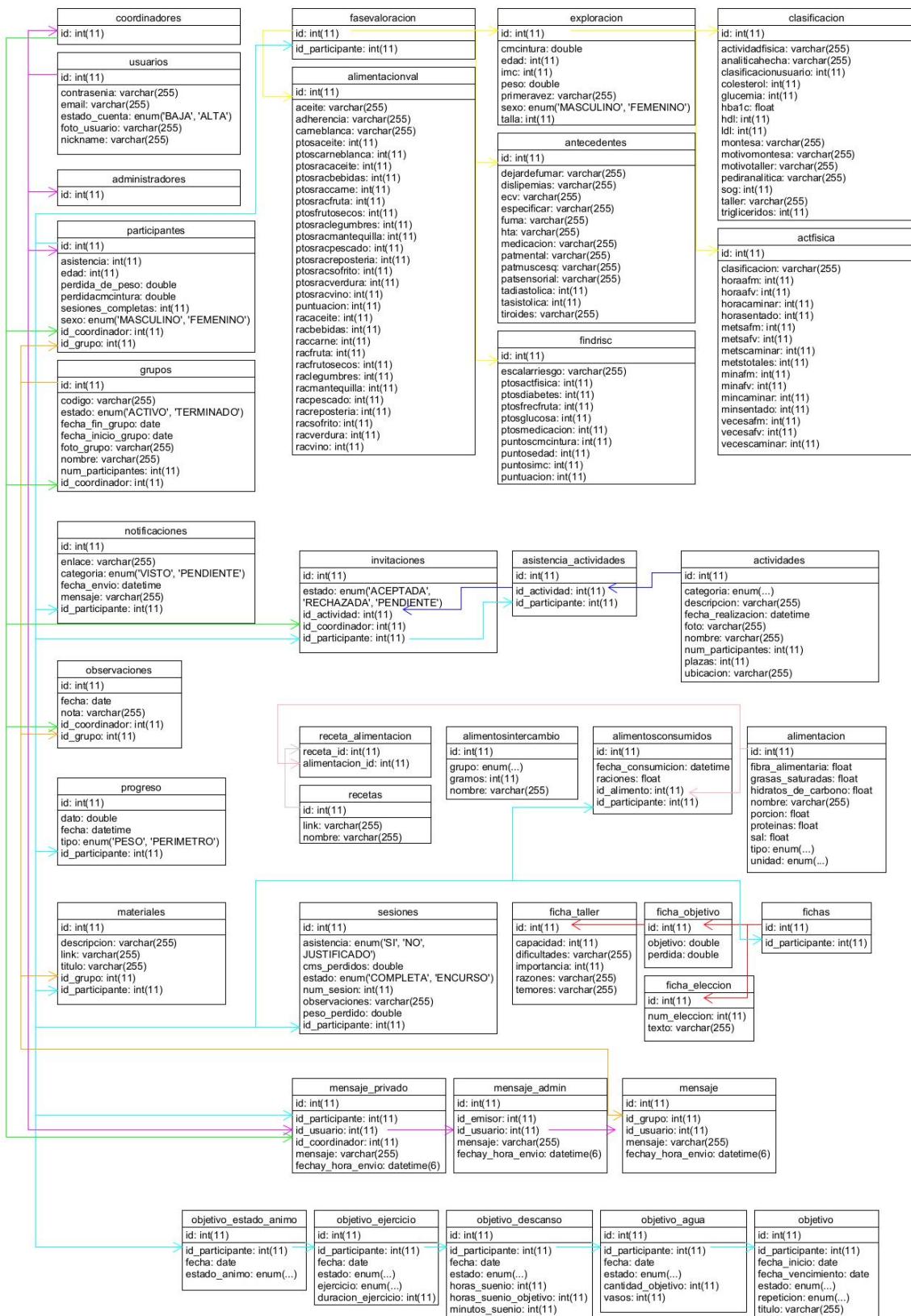


Figura 5.2: Esquema de base de datos

5.3. Estructura del código

5.3.1. Diagrama de componentes

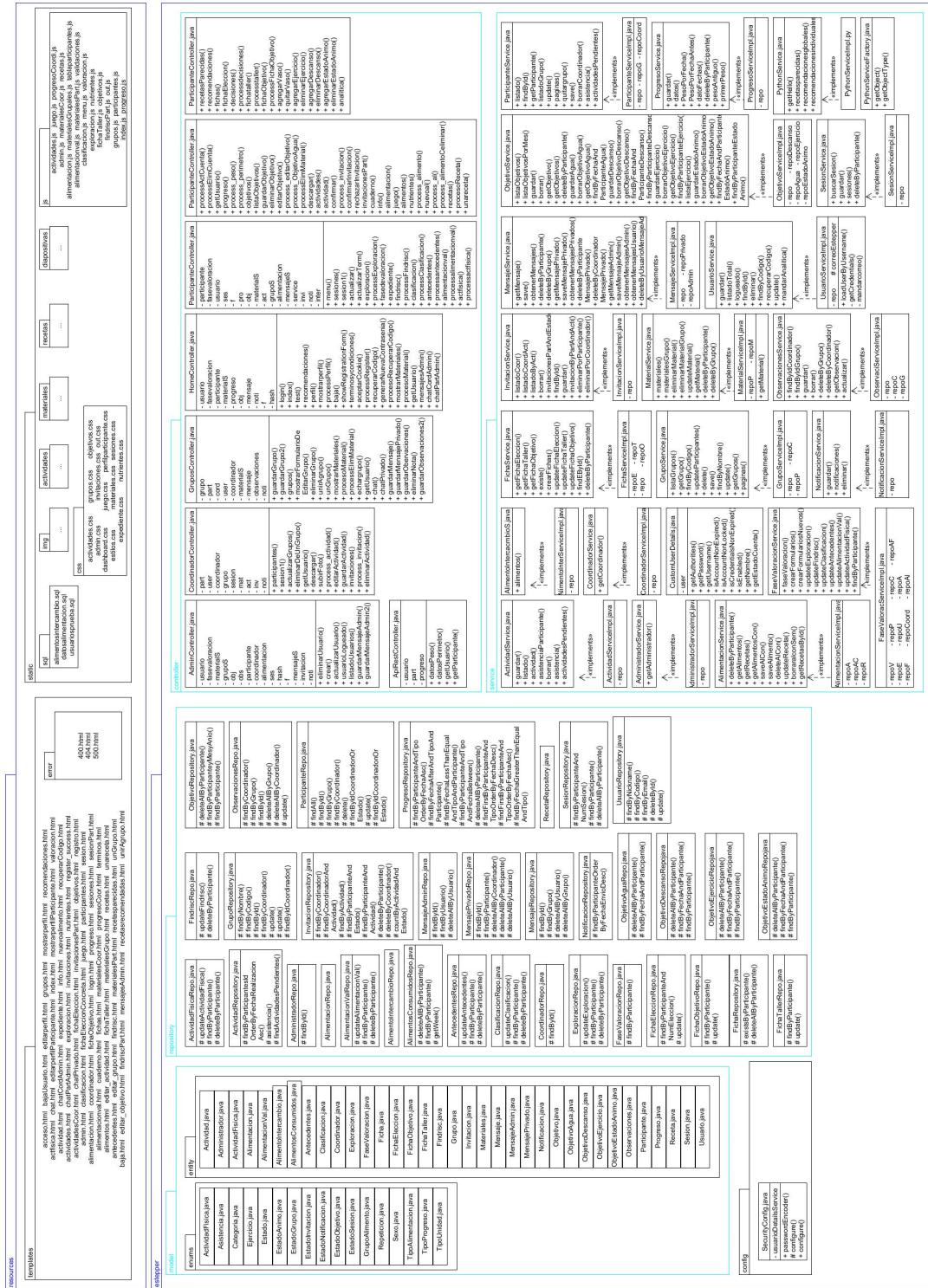


Figura 5.3: Diagrama de componentes

5.3.2. Modelo Vista Controlador

El Modelo Vista Controlador (MVC) es un patrón de arquitectura de software que se utiliza para separar la lógica, presentación y control de la aplicación mediante los tres elementos que componen su nombre. Un factor esencial para comprender este modelo es el DispatcherServlet. Este componente, es el que recibe las peticiones HTTP y devuelve las respectivas respuestas. De esta manera, se encarga de llamar al controlador correspondiente y seleccionar la vista adecuada a la que además, le envía los datos necesarios para que se muestre correctamente.

En la realización de este proyecto, se ha seguido este patrón de arquitectura, de manera que en el código se pueden distinguir los tres tipos de componentes del MVC.

Esto puede verse en el Diagrama de componentes 5.3 diseñado. En él, se visualiza de manera muy clara la distinción entre modelos, repositorios, controladores y servicios.

En relación a los modelos comprende tanto los enumerados como las entidades necesarias de la aplicación. Para identificar las entidades se utiliza la notación @Entity. Este modelo es el responsable de representar los datos los datos de la aplicación y las relaciones entre ellos. Por ejemplo, entre las entidades fundamentales de este proyecto destacan las que hacen referencia a los actores que intervienen en el mismo, es decir, Participante.java, Coordinador.java y Administrador.java. Estas entidades, que además contienen enumerados como por ejemplo Asistencia.java para el seguimiento de los participantes, se relacionan con el resto de componentes.

En lo que respecta a los controladores, se caracterizan por tener la notación @Controller. Así, todos los controladores implementados tienen esta notación salvo el controller para el API rest que es un caso especial cuya notación es @RestController. Gracias a esta terminología, el contenedor de Spring puede detectarlos rápidamente. Cabe destacar, que estos controladores contienen una gran variedad de métodos que van a permitir desarrollar el funcionamiento de la aplicación. De este modo, para implementar los métodos de manera adecuada, se tienen que tener en cuenta dos puntos principales. Por un lado, es necesario injectar recursos de los servicios utilizados al controlador y por otro, es conveniente conocer otras notaciones que se pueden incluir justo encima de cada función. Las más utilizadas son @GetMapping, @PostMapping y @RequestMapping. Por otro lado, resulta importante destacar que a partir de los controladores se accede a las distintas vistas implementadas en html.

Este trabajo se ha realizado implementando un sistema de micro servicios, que en el código se identifican por tener la notación @Service. Son los componentes que proporcionan la lógica de negocio de la aplicación. Así, son los encargados de recibir y procesar las solicitudes de los distintos controladores operando en el modelo y devolviendo los resultados obtenidos al controlador. En el proyecto, se ha separado entre la declaración e implementación de las funciones que se llevan a cabo en estos servicios, teniendo estos últimos la notación de @Override para un buen funcionamiento. Además, para poder realizar las consultas necesarias a la base de datos, es imprescindible que los servicios tengan injectados los recursos de los repositorios.

Esto se hace a través de la notación @Autowired. De esta manera, son los repositorios los que se comunican con la base de datos. Por ende, es en ellos donde se implementan los métodos necesarios para llevar a cabo las operaciones CRUD (create, read, update, delete). Por ello, en ellos es frecuente el uso de notaciones como @Modifying, @Transactional o @Query, pues es la manera de acceder a la base de datos.

Para exemplificar esto en un caso concreto se va a explicar el procedimiento que seguiría la aplicación para mostrar al administrador la vista principal con los usuarios de la misma. Para llevar a cabo este proceso, la entidad implicada es Usuario, con enumerados como el estado de la cuenta. De este modo, en el HomeController.java (notación: @Controller), se inyectan los recursos del UsuarioService mediante @Autowired. Gracias a esto, en la función index se van a poder realizar diversas operaciones. Cabe destacar, que esta función tiene en la parte superior la notación @GetMapping("/"), que se utiliza para asignarle a la ruta raíz o URL base, la solicitud HTTP GET. A continuación, dentro de la función, después de comprobar que el usuario que accede es de tipo administrador, se accede al listado total de usuarios mediante el servicio inyectado. De esta manera, al hacer usuario.listadoTotal(), se accede a la definición de dicha función en el UsuarioService.java, que a su vez dirige a la implementación de la misma en el UsuarioServiceImpl.java. Llegados a este punto, en el script del servicio, se inyectan los recursos del repositorio UsuarioRepository mediante @Autorwired. Gracias a ello, en la función listadoTotal, se puede acceder a la base de datos utilizando la función findAll() que devuelve la lista de usuarios que finalmente será enviada da nuevo al controlador. Finalmente, el controlador recibe la lista y la manda a la vista admin.html a través del empleo del model.addAttribute. Por consiguiente, desde el html puede accederse a la lista, recorrerla y mostrarla de la manera deseada, en este caso, en formato tabla.

5.3.3. Inyección de dependencias

El framework de Spring Boot utilizado para el desarrollo de este proyecto tiene una serie de particularidades. De esta manera, la forma de relacionar los distintos componentes de la aplicación se lleva a cabo mediante la inyección de dependencias o Dependency Injection (DI). Para ello, Spring Boot cuenta con un contenedor de inversión de control o Inversion of Control (IoC) que es el encargado de la administración de estas dependencias, que se gestionan a través de diversas anotaciones, así como de la creación, configuración y administración de los objetos usados en la aplicación.

En el archivo pom.xml, se definen las dependencias y configuración de todo el proyecto. De manera simplificada, es un archivo que contiene información sobre las distintas dependencias y plugins que se necesitan para que todo funcione correctamente.

Por ejemplo, con la dependencia de la figura 5.4 se permite utilizar el complemento de Thymeleaf.

```
<dependency>
    <groupId>org.thymeleaf.extras</groupId>
    <artifactId>thymeleaf-extras-springsecurity6</artifactId>
</dependency>
```

Figura 5.4: Ejemplo dependencia Thymeleaf

Dos conceptos relevantes para comprender adecuadamente la inyección de dependencias son Bean y ApplicationContext.

Un bean es un objeto configurado a través de xml que se gestiona por el contenedor IoC, es decir, el contenedor crea y gestiona el ciclo de vida del objeto devuelto a través del método con una notación de @Bean. Con esto se consigue que dichos objetos puedan ser utilizados como dependencias por otros componentes de la aplicación. Un caso de esto se da en el archivo EstepperApplication.java que utiliza la notación @Bean sobre el método PythonFactory de manera que los diversos componentes puedan utilizar el servicio de factoría de código en python.

Finalmente, en lo que respecta al ApplicationContext, se trata de un contenedor de Spring más avanzado, por lo que es capaz de realizar una gran variedad de funcionalidades a partir de numerosas notaciones. En concreto, en el archivo EstepperApplication anteriormente mencionado se hace uso de la notación @SpringBootApplication que a su vez combina otras 3 anotaciones: @Configuration, @EnableAutoConfiguration y @ComponentScan. De esta manera se pretende marcar la clase como la principal de la aplicación, se habilita la configuración automática de Spring Boot y se permite escanear componentes Spring a la vez que definir otros beans personalizados.

5.3.4. Seguridad

Proporcionar una aplicación segura es un requisito imprescindible en la implementación de cualquier proyecto. De esta manera, en el script SecurityConfig.java se ha integrado el módulo de seguridad proporcionado por Spring Boot. Así, gracias a la notación @EnableWebSecurity se habilita la configuración de seguridad web de la aplicación.

Otro aspecto relevante a tener en cuenta es el contenido de la clase SecurityConfig, pues en ella se definen numerosas reglas de seguridad. Por ejemplo, en el método configure (HttpSecurity http) se controla la autenticación y autorización de los usuarios de la aplicación. Además, a pesar de que la mayoría de las reglas indican URLs y recursos que pueden ser accedidos públicamente al utilizar método permitAll, se obliga a que el usuario esté autenticado para poder acceder a cualquiera de ellos (método authenticated).

Por otro lado, se ha especificado el método passwordEncoder cuya finalidad es encriptar y comparar las contraseñas de los distintos usuarios de una forma segura. Además, al emplearse un algoritmo unidireccional se impide la desencriptación. De hecho, es por esto mismo que al recuperar el código se obliga a generar una nueva contraseña.

5.4. Aplicación responsive

Uno de los requisitos de Mercedes, era desarrollar una aplicación que se pudiera utilizar desde un dispositivo móvil, para que los usuarios la pudieran utilizar desde cualquier lugar. Por esta razón, para llevar a cabo este objetivo se han adaptado las pantallas a cualquier dispositivo, con el fin de que la aplicación web sea accesible y funcional en dispositivos móviles mediante un diseño responsivo. Este diseño mejora la accesibilidad de la aplicación, traduciéndose en una mayor satisfacción del usuario. Para realizar esta adaptación se hace a través de CSS Media Queries, que permite aplicar estilos específicos según las características de la pantalla, como el ancho y la altura. Además, es muy importante probar la aplicación en distintos dispositivos para comprobar que la adaptación se ha desarrollado correctamente.

En el siguiente listado, aparecen algunas de las vistas de la aplicación adaptada a la pantalla de un dispositivo móvil.

1. Gestión de usuarios

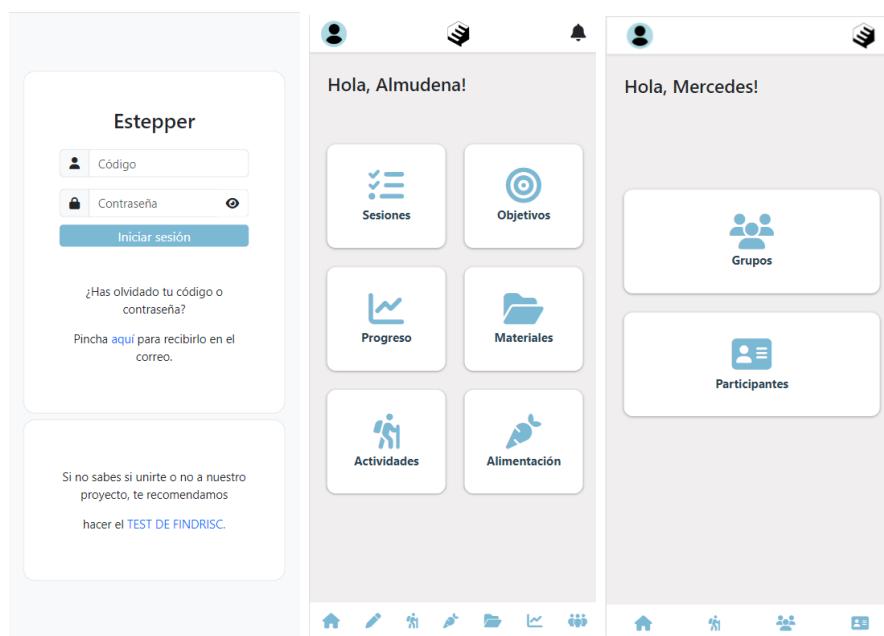


Figura 5.5: Vista responsive de inicio sesión, el inicio de participante y coordinador

2. Actividades, Progreso y Objetivos

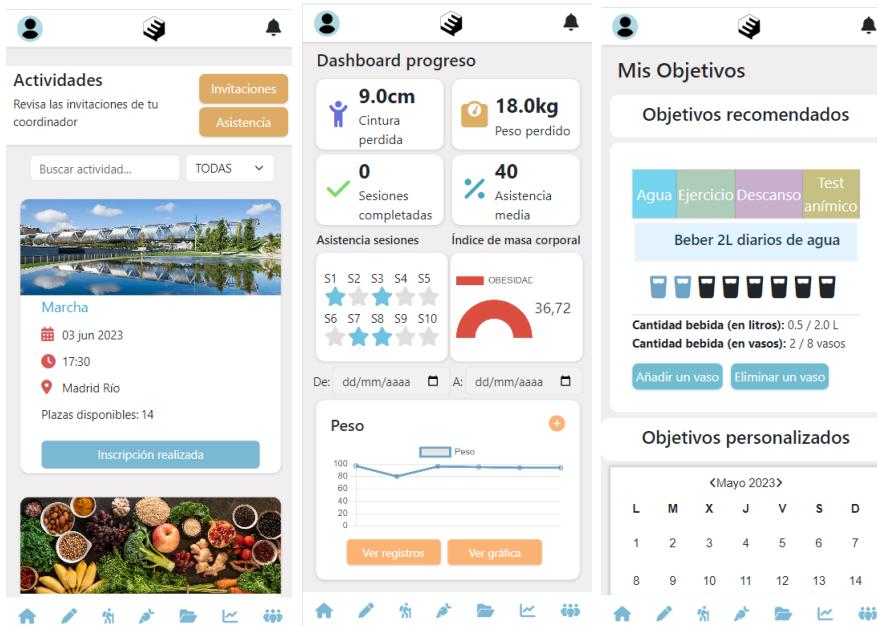


Figura 5.6: Vista responsive de actividades, progreso y objetivos

3. Materiales, Alimentación y Sesiones

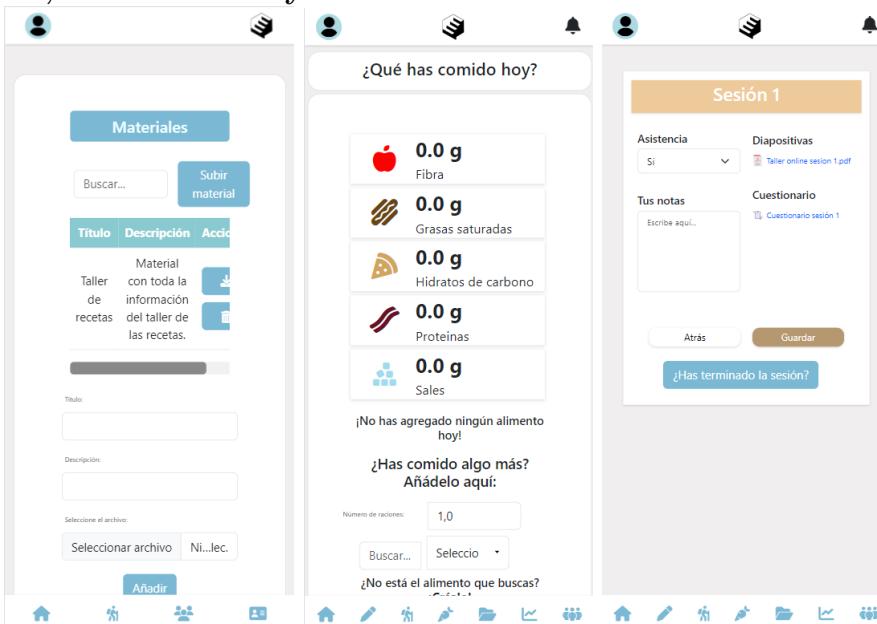


Figura 5.7: Vista responsive de materiales, alimentación y sesiones

4. Gestión de participantes, Gestión de grupos y Chat

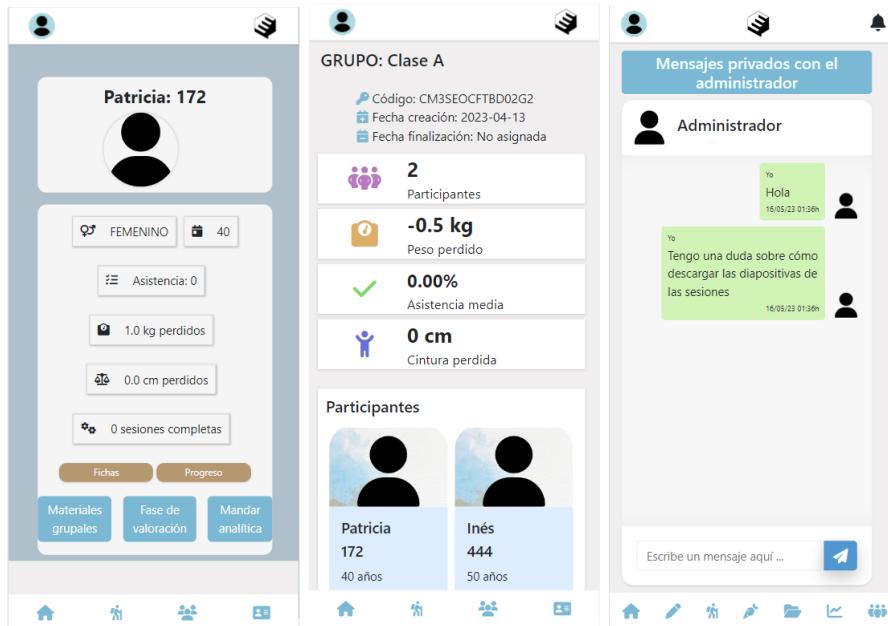


Figura 5.8: Vista responsive de gestión de participantes, grupos y chat

Capítulo 6

Especificación de requisitos

6.1. Actores

Al implementar las funcionalidades de esta aplicación web hemos identificado los siguientes actores:

6.1.1. Administrador

Habrá una cuenta por defecto ya creada de tipo Administrador. Introduciendo los datos de acceso de su cuenta, es decir, el código y la contraseña, se accede al panel de administración. Desde ahí, el administrador puede realizar las siguientes acciones:

- Ver los datos de todos los usuarios registrados en la aplicación.
- Filtrar los datos de los usuarios registrados.
- Editar el correo electrónico, el nickname o el estado de la cuenta de cualquier usuario registrado.
- Eliminar una cuenta ya existente.
- Crear una nueva cuenta de un coordinador.
- Mensajear a cualquier usuario registrado.

6.1.2. Coordinador

Existirá un usuario de tipo Coordinador por defecto, pero se podrán crear nuevos desde el usuario de tipo Administrador. Introduciendo los datos de acceso de estas cuentas, se accedería al panel de coordinación. Desde este punto, el coordinador podrá acceder a varias funcionalidades que se describirán más en detalle en los casos de uso.

6.1.3. Participante

Habrá varias cuentas creadas por defecto de tipo Participante. Estas pueden estar de Baja, o de Alta, en función de si el coordinador decide que el participante entre en el programa o no. Desde el inicio de sesión, introduciendo los datos de acceso correspondiente se accede al panel de los participantes. Desde ahí, el participante podrá acceder a varias funcionalidades que se describirán más en detalle en los casos de uso.

6.1.4. Usuario

Los actores de participante, coordinador y participante serán herencias de usuario, que es la cuenta raíz que comparten todos. Por tanto, cuando hablemos de un caso de uso que puedan usar varios actores diferentes usaremos el genérico de usuario. Asimismo, este actor servirá cuando aún no se haya registrado la cuenta en la aplicación.

6.2. Diagrama casos de uso UML

6.2.1. Gestión de usuarios

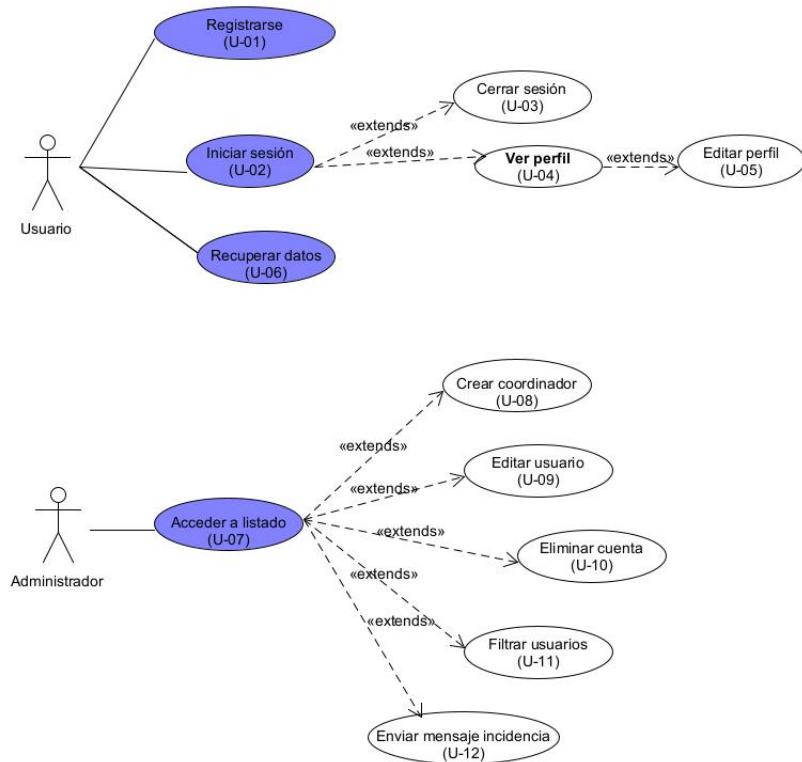


Figura 6.1: Diagrama de casos de uso de gestión de usuarios

6.2.2. Actividades

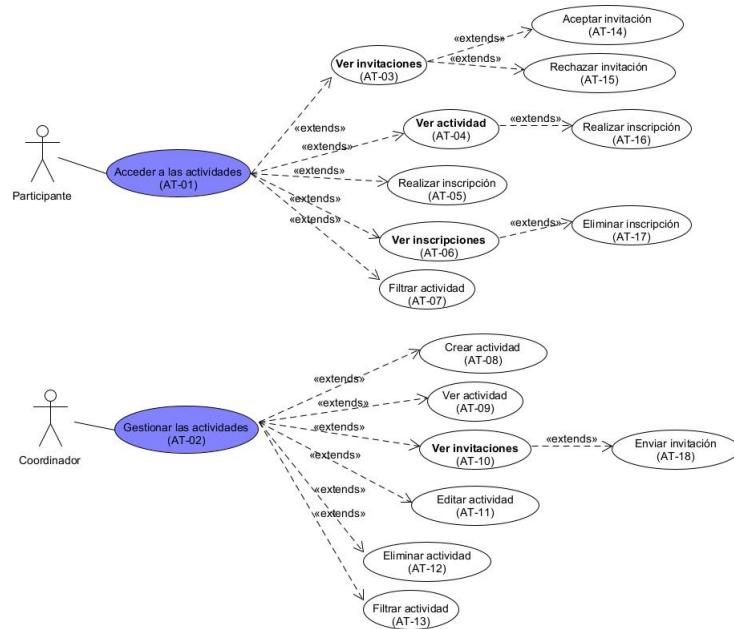


Figura 6.2: Diagrama de casos de uso de actividades

6.2.3. Materiales

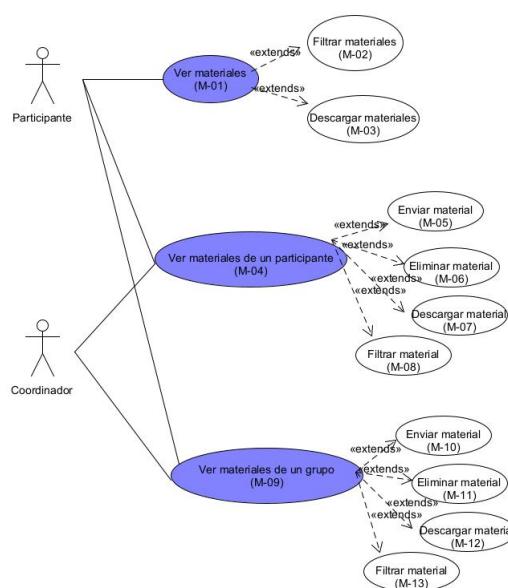


Figura 6.3: Diagrama de casos de uso de materiales

6.2.4. Progreso

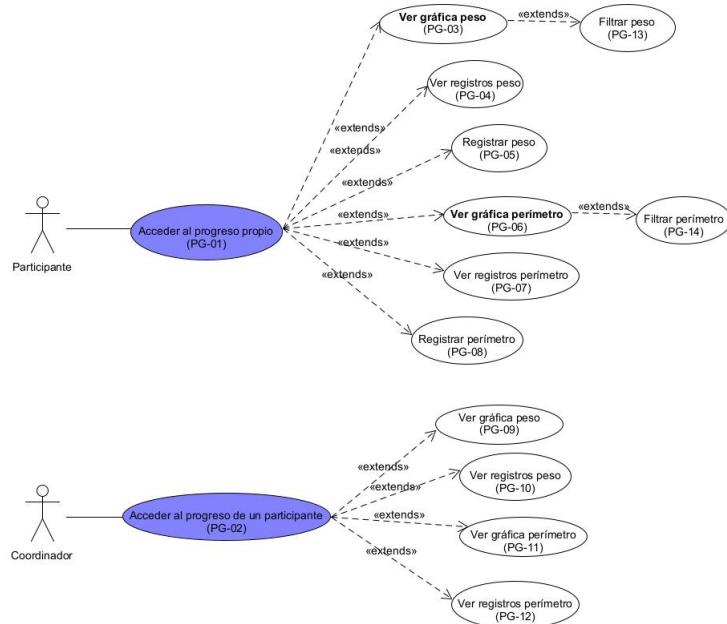


Figura 6.4: Diagrama de casos de uso de progreso

6.2.5. Chat

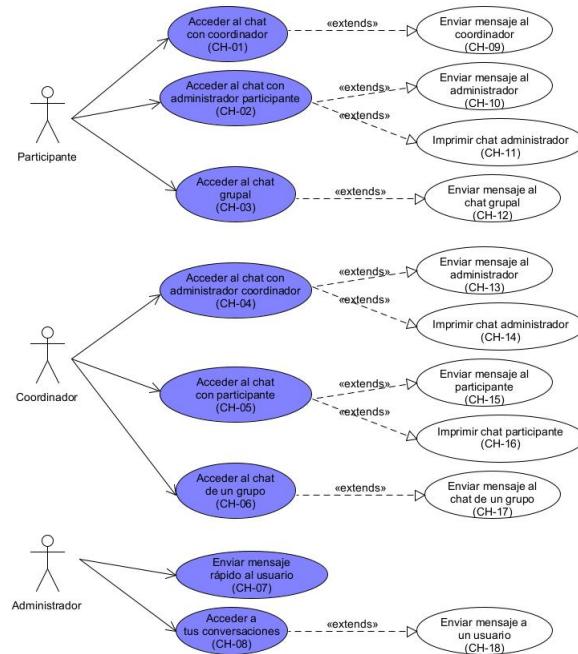


Figura 6.5: Diagrama de casos de uso de chat

6.2.6. Alimentación

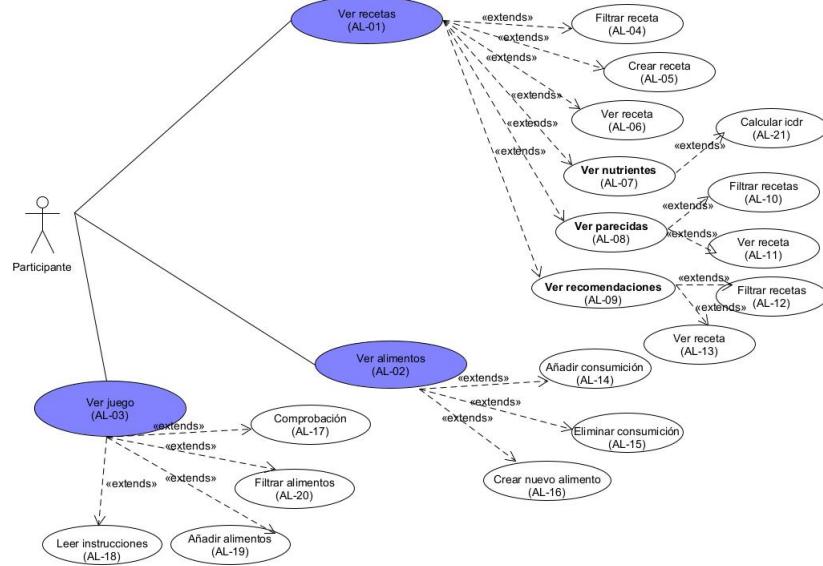


Figura 6.6: Diagrama de casos de uso de alimentación

6.2.7. Objetivos

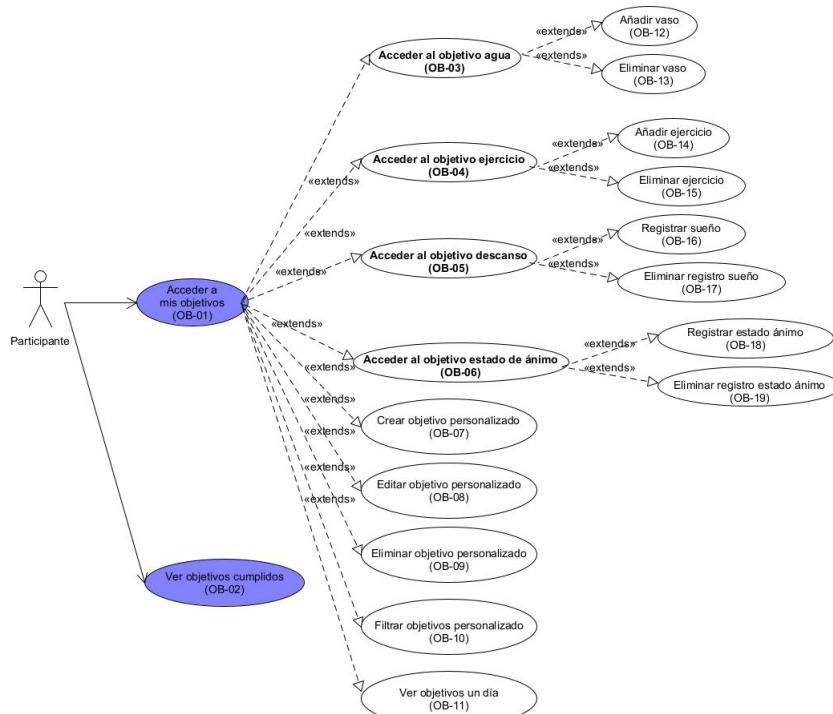


Figura 6.7: Diagrama de casos de uso de objetivos

6.2.8. Cuaderno y sesiones

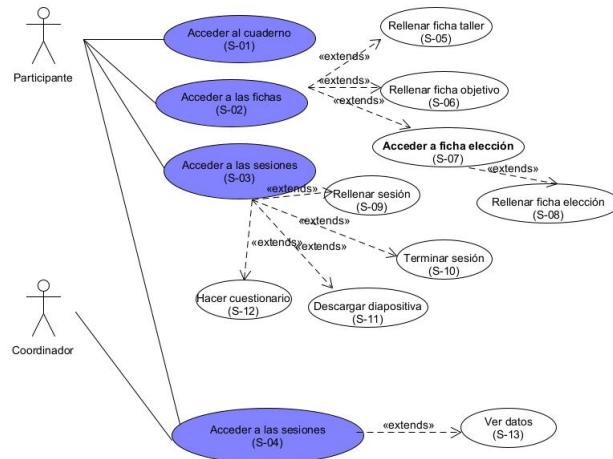


Figura 6.8: Diagrama de casos de uso de cuaderno y sesiones

6.2.9. Gestión de participantes

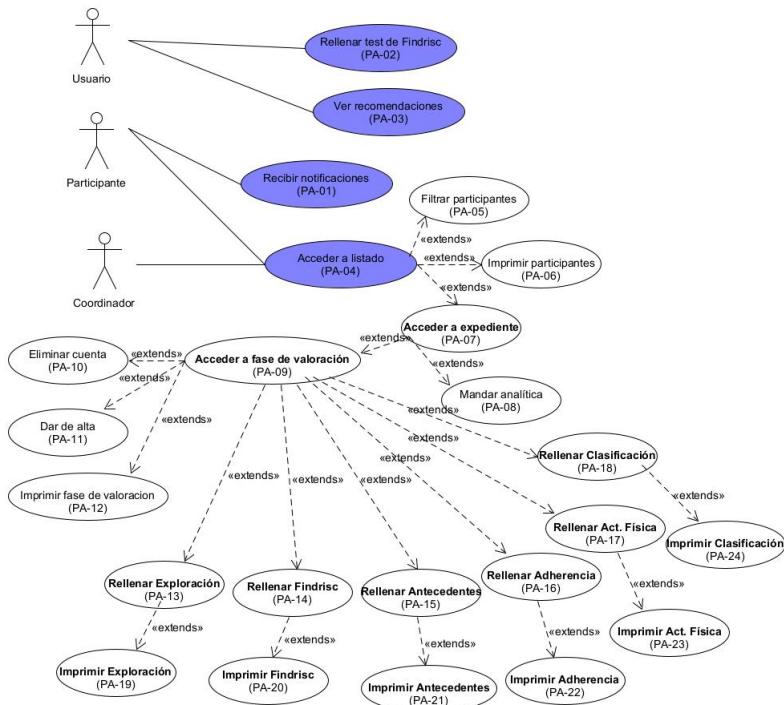


Figura 6.9: Diagrama de casos de uso de gestión de participantes

6.2.10. Gestión de grupos

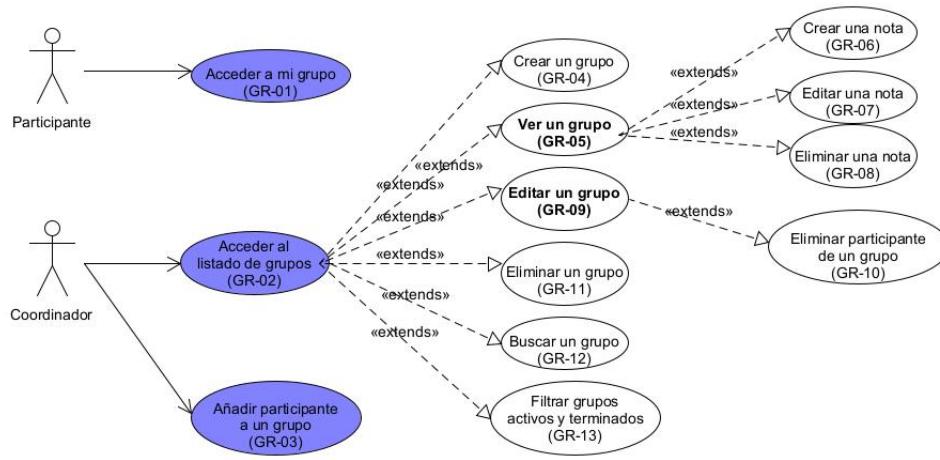


Figura 6.10: Diagrama de casos de uso de grupos

6.3. Casos de uso

6.3.1. Gestión de usuarios

1. Registrarse siendo Usuario

Caso de uso	Registrarse
Identificador	CU_U_01
Objetivo en contexto	Registrarse como nuevo usuario de Estepper.
Actores	Usuario
Entidades que usa	Usuario
Precondiciones	1. Que el usuario se encuentre en la vista del test de findrisc y por la puntuación se le recomiente registrarse.
Postcondiciones	Fallo: Se indican los errores. Éxito: Se crea el usuario correctamente y se redirige a la vista de usuario de baja.
Flujo Principal	1. El usuario pulsa en Pinchando aquí para registrarse. 2. El sistema muestra el formulario de registro. 3. El usuario introduce los datos solicitados. 4. El usuario pulsa en Registrarse. 5. El sistema comprueba que los datos introducidos son correctos. 6. El sistema guarda el usuario en la base de datos y redirige a la vista dado de baja.
Flujo Secundario	6. a) El sistema detecta un error en los datos introducidos. 6. b) El sistema pide que se rellenen bien los datos.

Figura 6.11: Caso de uso U01

2. Iniciar sesión siendo Usuario

Caso de uso	Iniciar sesión
Identificador	CU_U_02
Objetivo en contexto	Iniciar sesión en la aplicación.
Actores	Usuario
Entidades que usa	Usuario
Precondiciones	1. Que el usuario esté registrado y se encuentre en la vista de inicio de sesión.
Postcondiciones	Fallo: Se indican los errores. Éxito: Se inicia sesión y el usuario entra en la vista correspondiente de la aplicación.
Flujo Principal	1. El usuario rellena el formulario de inicio de sesión. 2. El usuario pulsa Iniciar sesión. 3. El sistema comprueba que los datos introducidos son correctos. 4. El sistema comprueba que el usuario esté de alta. 5. El sistema redirige al usuario a la vista principal.
Flujo Secundario	4. a) El sistema detecta un error en los datos introducidos. 4. b) El sistema pide que se rellenen bien los datos. 5. a) El sistema detecta que el usuario está de baja. 5. b) El sistema redirige al usuario a la vista de baja.

Figura 6.12: Caso de uso U02

3. Cerrar sesión siendo Usuario

Caso de uso	Cerrar sesión
Identificador	CU_U_03
Objetivo en contexto	Cerrar sesión en la aplicación.
Actores	Usuario
Entidades que usa	Usuario
Precondiciones	1. Que el usuario haya iniciado sesión.
Postcondiciones	Fallo: Se devuelve a la página principal. Éxito: Se cierra sesión y el usuario sale de la aplicación que se redirige al inicio de sesión.
Flujo Principal	1. El usuario pulsa en Cerrar sesión. 2. El sistema cierra la sesión del usuario. 3. El sistema se redirige a la vista de inicio de sesión.

Figura 6.13: Caso de uso U03

4. Ver perfil siendo Usuario

Caso de uso	Ver perfil
Identificador	CU_U_04
Objetivo en contexto	Ver tu perfil de usuario en la aplicación.
Actores	Usuario
Entidades que usa	Usuario
Precondiciones	1. Que el usuario haya iniciado sesión.
Postcondiciones	Fallo: Se devuelve a la página principal. Éxito: El usuario se encuentra en la vista de mostrar perfil donde tiene su información.
Flujo Principal	1. El usuario pulsa en Mi perfil. 3. El sistema se redirige a la vista de mi perfil del usuario.

Figura 6.14: Caso de uso U04

5. Editar perfil siendo Usuario

Caso de uso	Editar perfil
Identificador	CU_U_05
Objetivo en contexto	Editar tu perfil de usuario en la aplicación.
Actores	Usuario
Entidades que usa	Usuario
Precondiciones	1. Que el usuario se encuentre en la vista de mi perfil.
Postcondiciones	Fallo: Se devuelve a la página principal. Éxito: Se modifica la información del perfil del usuario en la base de datos.
Flujo Principal	1. El usuario pulsa en Editar perfil. 2. El sistema muestra el formulario. 3. El usuario modifica los campos deseados. 4. El usuario pulsa Guardar. 5. El sistema comprueba que los datos introducidos son correctos. 6. El sistema redirige al usuario a la vista del perfil.
Flujo Secundario	5. a) El sistema detecta un error en los datos introducidos. 5. b) El sistema pide que se rellenen bien los datos.

Figura 6.15: Caso de uso U05

6. Recuperar datos siendo Usuario

Caso de uso	Recuperar datos
Identificador	CU_U_06
Objetivo en contexto	Recuperar la contraseña o código del usuario.
Actores	Usuario
Entidades que usa	Usuario
Precondiciones	1. Que el usuario se encuentre en la vista de iniciar sesión.
Postcondiciones	Fallo: Se muestran los errores. Éxito: Se envía un correo con la información recuperada al usuario.
Flujo Principal	1. El usuario pulsa en el link para recuperar los datos. 2. El sistema muestra el formulario. 3. El usuario introduce su dirección de correo. 4. El usuario pulsa Enviar. 5. El sistema comprueba que los datos introducidos son correctos. 6. El sistema redirige al usuario a la vista de iniciar sesión. 7. El sistema envía un correo al usuario con los datos recuperados.
Flujo Secundario	5. a) El sistema detecta un error en los datos introducidos. 5. b) El sistema pide que se rellenen bien los datos.

Figura 6.16: Caso de uso U06

7. Acceder al listado de usuarios siendo Administrador

Caso de uso	Acceder al listado
Identificador	CU_U_07
Objetivo en contexto	Acceder al listado de usuarios.
Actores	Administrador, Coordinador, Participante
Entidades que usa	Administrador, Coordinador, Participante
Precondiciones	1. Que el administrador haya accedido a la aplicación.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Aparece el listado de usuarios. y el acceso a las funcionalidades que extiende.
Flujo Principal	1. El coordinador está en la página principal Usuarios. 2. El sistema muestra los usuarios.

Figura 6.17: Caso de uso U07

8. Crear coordinador siendo Administrador

Caso de uso	Crear un nuevo coordinador
Identificador	CU_U_08
Objetivo en contexto	Añadir un nuevo coordinador al listado de coordinadores.
Actores	Administrador, Coordinador
Entidades que usa	Administrador, Coordinador
Precondiciones	1. Que el administrador se encuentre en la vista de Usuarios.
Postcondiciones	Fallo: Se muestran los errores en el formulario. Éxito: Se crea el coordinador correctamente y se redirige a la vista de Usuarios.
Flujo Principal	<ol style="list-style-type: none"> 1. El administrador pulsa en "Nuevo coordinador". 2. El sistema muestra el formulario. 3. El administrador rellena el formulario para crear el coordinador. 4. Pulsa en "Guardar". 5. El sistema comprueba que los datos introducidos son correctos. 6. El sistema añade el coordinador a la base de datos.
Flujo Secundario	<ol style="list-style-type: none"> 6. a) El sistema detecta un error en los datos introducidos. 6. b) El sistema pide que se rellenen bien los datos.

Figura 6.18: Caso de uso U08

9. Editar usuario siendo Administrador

Caso de uso	Editar un usuario
Identificador	CU_U_09
Objetivo en contexto	Editar un usuario del listado de usuarios.
Actores	Administrador, Coordinador, Participante
Entidades que usa	Administrador, Coordinador, Participante
Precondiciones	1. Que el administrador se encuentre en la vista de Usuarios.
Postcondiciones	Fallo: Se indican los errores. Éxito: Se modifica el usuario correctamente y se redirige a la vista de Usuarios.
Flujo Principal	1. El administrador pulsa en el icono de Editar. 2. El sistema indica los campos editables. 3. El administrador modifica los datos. 4. Pulsa en el icono de Guardar. 5. El sistema comprueba que los datos introducidos son correctos. 6. El sistema actualiza el usuario en la base de datos.
Flujo Secundario	6. a) El sistema detecta un error en los datos introducidos. 6. b) El sistema pide que se rellenen bien los datos.

Figura 6.19: Caso de uso U09

10. Eliminar cuenta siendo Administrador

Caso de uso	Eliminar una cuenta
Identificador	CU_U_10
Objetivo en contexto	Eliminar una cuenta de usuario.
Actores	Administrador, Coordinador, Participante
Entidades que usa	Administrador, Coordinador, Participante
Precondiciones	1. Que el administrador se encuentre en la vista de Usuarios.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Se elimina el usuario correctamente.
Flujo Principal	1. El administrador pulsa en el icono de Eliminar del usuario seleccionado. 2. Se elimina el usuario. 3. El sistema muestra el listado actualizado.

Figura 6.20: Caso de uso U10

11. Filtrar usuarios siendo Administrador

Caso de uso	Filtrar usuarios
Identificador	CU_U_11
Objetivo en contexto	Buscar el usuario indicado.
Actores	Administrador, Coordinador, Participante
Entidades que usa	Administrador, Coordinador, Participante
Precondiciones	1. Que el administrador se encuentre en la vista de Usuarios.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Se muestran los usuarios con el filtro indicado.
Flujo Principal	1. El administrador se encuentra en la página de Usuarios. 2. Escribe un texto en el buscador. 3. El sistema muestra una lista de usuarios obtenidos con el filtro.

Figura 6.21: Caso de uso U11

12. Enviar mensaje incidencia siendo Administrador

Caso de uso	Enviar mensaje incidencia
Identificador	CU_U_12
Objetivo en contexto	Enviar un mensaje de incidencia a un usuario.
Actores	Administrador, Coordinador, Participante
Entidades que usa	Administrador, Coordinador, Participante
Precondiciones	1. Que el administrador se encuentre en la vista de Usuarios.
Postcondiciones	Fallo: Se vuelve a la vista principal. Éxito: Se envía el mensaje correctamente y se redirige a la vista de Mensajes.
Flujo Principal	1. El administrador pulsa en el ícono de Mensaje del usuario seleccionado. 2. El sistema muestra el formulario del mensaje. 3. El coordinador escribe el mensaje a enviar. 4. Pulsa en Enviar. 5. Se envía el mensaje. 6. El sistema muestra la conversación con el usuario.

Figura 6.22: Caso de uso U12

6.3.2. Actividades

1. Acceder a las actividades siendo Participante

Caso de uso	Acceder a las actividades
Identificador	CU_AT_01
Objetivo en contexto	Acceder a la lista de actividades
Actores	Participante
Entidades que usa	Actividad, Participante
Precondiciones	1. Que el participante esté en estado de Alta. 2. Que el participante haya accedido a la aplicación.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Aparecen las actividades con la información y el acceso a las funcionalidades que extiende.
Flujo Principal	1. El participante está en la página principal 2. Pulsa en “Actividades” 3. El sistema muestra las actividades

Figura 6.23: Caso de uso AT01

2. Ver invitaciones siendo Participante

Caso de uso	Ver invitaciones
Identificador	CU_AT_03
Objetivo en contexto	Ver las invitaciones a las actividades
Actores	Participante
Entidades que usa	Actividad, Participante, Invitación
Precondiciones	1. Que el participante esté en estado de Alta. 2. Que el participante se encuentre en la vista de Actividades.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Aparecen las invitaciones diferenciadas por el estado de éstas, pudiendo ser pendientes, aceptadas o rechazadas. Además del acceso a aceptar o rechazar una invitación pendiente.
Flujo Principal	1. El participante pulsa en “Invitaciones” 2. El sistema muestra las invitaciones

Figura 6.24: Caso de uso AT03

3. Aceptar una invitación siendo Participante

Caso de uso	Aceptar una invitación
Identificador	CU_AT_14
Objetivo en contexto	Aceptar una invitación pendiente
Actores	Participante
Entidades que usa	Actividad, Participante, Invitación
Precondiciones	1. Que el participante esté en estado de Alta. 2. Que el participante se encuentre en la vista de Invitaciones.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Se confirma la asistencia a la actividad.
Flujo Principal	1. El participante pulsa en “Aceptar” de una invitación 2. Se acepta la invitación 3. El sistema comprueba que queden plazas 4. El sistema muestra las invitaciones actualizadas
Flujo Secundario	2.1. El sistema reconoce que ya no quedan plazas al realizar la inscripción El sistema manda una notificación a los que hayan sido invitados

Figura 6.25: Caso de uso AT14

4. Rechazar una invitación siendo Participante

Caso de uso	Rechazar una invitación
Identificador	CU_AT_15
Objetivo en contexto	Rechazar una invitación pendiente
Actores	Participante
Entidades que usa	Actividad, Participante, Invitación
Precondiciones	1. Que el participante esté en estado de Alta. 2. Que el participante se encuentre en la vista de Invitaciones.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Se rechaza la invitación a la actividad.
Flujo Principal	1. El participante pulsa en “Rechazar” de una invitación 2. Se rechaza la invitación 2. El sistema muestra las invitaciones actualizadas

Figura 6.26: Caso de uso AT15

5. Ver una actividad siendo Participante

Caso de uso	Ver una actividad
Identificador	CU_AT_04
Objetivo en contexto	Ver una actividad concreta
Actores	Participante
Entidades que usa	Actividad, Participante
Precondiciones	1. Que el participante esté en estado de Alta. 2. Que el participante se encuentre en la vista de Actividades.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Aparece la información de la actividad.
Flujo Principal	1. El sistema comprueba si tiene asistencia confirmada 2. El participante pulsa en “Inscribirme” de una actividad concreta 3. El participante pulsa en "Más información" 4. El sistema muestra la información detallada de la actividad correspondiente
Flujo Secundario	2. a) El sistema reconoce que tiene asistencia confirmada. 2. b) El participante pulsa en “Inscripción realizada” 2. c) Va al paso 3

Figura 6.27: Caso de uso AT04

6. Realizar inscripción a una actividad siendo Participante desde la vista de Actividades

Caso de uso	Realizar inscripción
Identificador	CU_AT_05
Objetivo en contexto	Realizar inscripción a una actividad para confirmar la asistencia
Actores	Participante
Entidades que usa	Actividad, Participante
Precondiciones	1. Que el participante esté en estado de Alta. 2. Que el participante se encuentre en la vista de Actividades.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Se confirma la asistencia a la actividad.
Flujo Principal	1. El participante pulsa en "Inscribirme" de una actividad 2. El participante pulsa en "Más información" 3. El participante pulsa en "Inscripción" 4. El participante pulsa en "Confirmar asistencia" 5. Se realiza la inscripción 6. El sistema comprueba que quedan plazas 7. El sistema muestra el listado de actividades
Flujo Secundario	6. a) El sistema reconoce que ya no quedan plazas al realizar la inscripción El sistema manda una notificación a los que hayan sido invitados

Figura 6.28: Caso de uso AT05

7. Realizar inscripción a una actividad siendo Participante desde una actividad concreta

Caso de uso	Realizar inscripción
Identificador	CU_AT_16
Objetivo en contexto	Realizar inscripción a una actividad para confirmar la asistencia
Actores	Participante
Entidades que usa	Actividad, Participante
Precondiciones	1. Que el participante esté en estado de Alta. 2. Que el participante se encuentre en la vista de Actividad.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Se confirma la asistencia a la actividad.
Flujo Principal	1. El participante pulsa en "Inscribirme" de una actividad 2. El participante pulsa en "Confirmar" 3. Se realiza la inscripción 4. El sistema comprueba que quedan plazas 5. El sistema actualiza la asistencia confirmada en el calendario y muestra el listado de actividades
Flujo Secundario	4. a) El sistema reconoce que ya no quedan plazas al realizar la inscripción El sistema manda una notificación a los que hayan sido invitados

Figura 6.29: Caso de uso AT16

8. Ver inscripciones siendo Participante

Caso de uso	Ver inscripciones
Identificador	CU_AT_06
Objetivo en contexto	Ver inscripciones confirmadas a las actividades
Actores	Participante
Entidades que usa	Actividad, Participante
Precondiciones	1. Que el participante esté en estado de Alta. 2. Que el participante se encuentre en la vista de Actividades.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Aparece el calendario con las inscripciones.
Flujo Principal	1. El participante se encuentra en la vista de Actividades 2. El participante pulsa en "Asistencia" 3. El sistema muestra el calendario de inscripciones

Figura 6.30: Caso de uso AT06

9. Eliminar inscripción a una actividad siendo Participante

Caso de uso	Eliminar inscripción
Identificador	CU_AT_17
Objetivo en contexto	Eliminar inscripción a una actividad
Actores	Participante
Entidades que usa	Actividad, Participante
Precondiciones	1. Que el participante esté en estado de Alta. 2. Que el participante se encuentre en el modal de Asistencia.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Se elimina la inscripción correctamente.
Flujo Principal	1. El participante pulsa en la "cruz" de la inscripción seleccionada 2. Se elimina la inscripción 3. El sistema muestra el calendario actualizado

Figura 6.31: Caso de uso AT17

10. Filtrar actividades siendo Participante

Caso de uso	Filtrar actividad
Identificador	CU_AT_07
Objetivo en contexto	Buscar la actividad indicada
Actores	Participante
Entidades que usa	Actividad, Participante
Precondiciones	1. Que el participante esté en estado de Alta. 2. Que el participante se encuentre en la vista de Actividades.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Se muestran las actividades con el filtro indicado.
Flujo Principal	1. El participante se encuentra en la página de Actividades 2. Escribe un texto de búsqueda 3. Selecciona un filtro 4. El sistema muestra una lista de actividades que se obtiene por el filtro

Figura 6.32: Caso de uso AT07

11. Gestionar las actividades siendo Coordinador

Caso de uso	Gestionar las actividades
Identificador	CU_AT_02
Objetivo en contexto	Acceder a la lista de actividades para gestionarlas
Actores	Coordinador
Entidades que usa	Actividad, Coordinador
Precondiciones	1. Que el coordinador haya accedido a la aplicación.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Aparecen las actividades con la información y el acceso a las funcionalidades que extiende.
Flujo Principal	1. El coordinador está en la página principal 2. Pulsa en "Actividades" 3. El sistema muestra las actividades

Figura 6.33: Caso de uso AT02

12. Crear actividad siendo Coordinador

Caso de uso	Crear una nueva actividad
Identificador	CU_AT_08
Objetivo en contexto	Añadir una nueva actividad al listado de actividades
Actores	Coordinador
Entidades que usa	Actividad, Coordinador
Precondiciones	1. Que el coordinador se encuentre en la vista de Actividades.
Postcondiciones	Fallo: Se muestran los errores en el formulario. Éxito: Se crea la actividad correctamente y se redirige a la vista de Actividades.
Flujo Principal	1. El coordinador pulsa en "Nueva Actividad". 2. El sistema muestra el formulario. 3. El coordinador rellena el formulario para crear la actividad. 4. Pulsa en "Guardar". 5. El sistema comprueba que los datos introducidos son correctos. 6. El sistema añade la actividad a la base de datos.
Flujo Secundario	6. a) El sistema detecta un error en los datos introducidos. 6. b) El sistema pide que se rellenen bien los datos

Figura 6.34: Caso de uso AT08

13. Ver una actividad siendo Coordinador

Caso de uso	Ver una actividad
Identificador	CU_AT_09
Objetivo en contexto	Ver una actividad concreta
Actores	Coordinador
Entidades que usa	Actividad, Coordinador
Precondiciones	1. Que el coordinador se encuentre en la vista de Actividades.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Aparece la información de la actividad.
Flujo Principal	1. El coordinador pulsa en “Información” de una actividad concreta 2. El sistema muestra la información detallada de la actividad correspondiente

Figura 6.35: Caso de uso AT09

14. Ver invitaciones de una actividad siendo Coordinador

Caso de uso	Ver invitaciones
Identificador	CU_AT_10
Objetivo en contexto	Ver invitaciones de una actividad concreta
Actores	Coordinador
Entidades que usa	Actividad, Coordinador, Invitación
Precondiciones	1. Que el coordinador se encuentre en la vista de Actividades.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Aparecen las invitaciones de la actividad.
Flujo Principal	1. El coordinador pulsa en “Invitaciones” de una actividad concreta 2. El sistema muestra las invitaciones de la actividad correspondiente y el acceso a la funcionalidad que extiende.

Figura 6.36: Caso de uso AT10

15. Enviar invitación de una actividad siendo Coordinador

Caso de uso	Enviar invitación
Identificador	CU_AT_18
Objetivo en contexto	Enviar invitación a una actividad
Actores	Coordinador
Entidades que usa	Actividad, Coordinador, Invitación
Precondiciones	1. Que el coordinador se encuentre en la vista de las Invitaciones.
Postcondiciones	Fallo: Se redirige a la vista de Invitaciones. Éxito: Se envía la invitación correctamente y se redirige a la vista de Invitaciones de la actividad
Flujo Principal	1. El coordinador selecciona el destinatario. 2. El sistema comprueba si es individual. 3. Pulsa en Enviar. 4. Se envía la invitación. 5. Se envía la notificación de la invitación.
Flujo Secundario	3. a) El sistema detecta que es una invitación individual, comprueba si existe y si no existe se informa al usuario.

Figura 6.37: Caso de uso AT18

16. Editar actividad siendo Coordinador

Caso de uso	Editar una actividad
Identificador	CU_AT_11
Objetivo en contexto	Editar una actividad del listado de actividades
Actores	Coordinador
Entidades que usa	Actividad, Coordinador
Precondiciones	1. Que el coordinador se encuentre en la vista de Actividades.
Postcondiciones	Fallo: Se muestran los errores en el formulario. Éxito: Se modifica la actividad correctamente y se redirige a la vista de Actividades.
Flujo Principal	1. El coordinador pulsa en el ícono de Editar. 2. El sistema muestra el formulario. 3. El coordinador modifica el formulario para editar la actividad. 4. Pulsa en "Guardar". 5. El sistema comprueba que los datos introducidos son correctos. 6. El sistema actualiza la actividad en la base de datos.
Flujo Secundario	6. a) El sistema detecta un error en los datos introducidos. 6. b) El sistema pide que se rellenen bien los datos

Figura 6.38: Caso de uso AT11

17. Eliminar actividad siendo Coordinador

Caso de uso	Eliminar actividad
Identificador	CU_AT_12
Objetivo en contexto	Eliminar la actividad seleccionada
Actores	Coordinador
Entidades que usa	Actividad, Coordinador
Precondiciones	1. Que el coordinador se encuentre en la vista de Actividades.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Se elimina la actividad correctamente.
Flujo Principal	1. El coordinador pulsa en el ícono de Eliminar de una actividad concreta 2. Se elimina la actividad

Figura 6.39: Caso de uso AT12

18. Filtrar actividades siendo Coordinador

Caso de uso	Filtrar actividad
Identificador	CU_AT_13
Objetivo en contexto	Buscar la actividad indicada
Actores	Coordinador
Entidades que usa	Actividad, Coordinador
Precondiciones	1. Que el coordinador se encuentre en la vista de Actividades.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Se muestran las actividades con el filtro indicado.
Flujo Principal	1. El coordinador se encuentra en la página de Actividades 2. Escribe un texto de búsqueda 3. Selecciona un filtro 4. El sistema muestra una lista de actividades que se obtiene por el filtro

Figura 6.40: Caso de uso AT13

6.3.3. Materiales

1. Ver materiales siendo Participante

Caso de uso	Ver materiales
Identificador	CU_M_01
Objetivo en contexto	Ver la lista de materiales
Actores	Participante
Entidades que usa	Materiales, Participante
Precondiciones	1. Que el participante esté en estado de Alta. 2. Que el participante haya accedido a la aplicación.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Aparece una lista con los materiales y acceso a las funcionalidades que extiende.
Flujo Principal	1. El participante está en la página principal 2. Pulsa en "Materiales" 3. El sistema muestra la lista de materiales

Figura 6.41: Caso de uso M01

2. Descargar un material siendo un Participante

Caso de uso	Descargar material
Identificador	CU_M_03
Objetivo en contexto	Descargar el material seleccionado de la lista
Actores	Participante
Entidades que usa	Materiales, Participante
Precondiciones	1. Que el participante esté en estado de Alta. 2. Que el participante haya accedido a la aplicación.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Se descarga el material correctamente.
Flujo Principal	1. El participante está en la página principal 2. Pulsa en "Materiales" 3. El sistema muestra la lista de materiales 4. Pulsa en el símbolo de Descargar de algún material. 5. Se descarga el material.

Figura 6.42: Caso de uso M03

3. Filtrar materiales siendo Participante

Caso de uso	Filtrar material
Identificador	CU_M_02
Objetivo en contexto	Buscar el material indicado
Actores	Participante
Entidades que usa	Materiales, Participante
Precondiciones	1. Que el participante esté en estado de Alta. 2. Que el participante haya accedido a la aplicación.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Se muestran los materiales con el filtro indicado.
Flujo Principal	1. El participante está en la página principal 2. Pulsa en "Materiales" 3. El sistema muestra la lista de materiales 4. Escribe un texto de búsqueda 5. Se muestra una lista con los resultados devueltos por el filtro.

Figura 6.43: Caso de uso M02

4. Enviar un material a un participante en concreto siendo Coordinador

Caso de uso	Enviar un material
Identificador	CU_M_05
Objetivo en contexto	Enviar un material a un participante en concreto
Actores	Participante y Coordinador
Entidades que usa	Materiales, Participante, Coordinador
Precondiciones	1. Que el coordinador haya accedido a la aplicación.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Se muestran los materiales enviados.
Flujo Principal	1. El coordinador está en la página principal 2. Pulsa en "Participantes" 3. El coordinador pulsa en la carpeta de materiales de un participante 4. El sistema muestra la lista de materiales 5. El coordinador rellena el formulario para enviar el material. 6. El sistema comprueba que los datos introducidos son correctos. 7. El sistema añade el material a la lista. 8. Se muestra una lista con los materiales
Flujo Secundario	7. a) El sistema detecta un error en los datos introducidos. 7. b) El sistema pide que cumplimenten bien esos datos.

Figura 6.44: Caso de uso M05

5. Ver los materiales enviados a un participante en concreto siendo Coordinador

Caso de uso	Ver materiales
Identificador	CU_M_04
Objetivo en contexto	Ver materiales de un participante en concreto
Actores	Participante y Coordinador
Entidades que usa	Materiales, Participante, Coordinador
Precondiciones	1. Que el coordinador haya accedido a la aplicación.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Aparece una lista con los materiales y acceso a las funcionalidades que extiende.
Flujo Principal	1. El coordinador está en la página principal 2. Pulsa en "Participantes" 3. El coordinador pulsa en la carpeta de materiales de un participante 4. El sistema muestra la lista de materiales

Figura 6.45: Caso de uso M04

6. Filtrar los materiales enviados a un participante en concreto siendo Coordinador

Caso de uso	Filtrar material
Identificador	CU_M_08
Objetivo en contexto	Buscar el material indicado
Actores	Participante, Materiales, Coordinador
Entidades que usa	Participante, Materiales, Coordinador
Precondiciones	1. Que el coordinador haya accedido a la aplicación.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Se muestran los materiales con el filtro indicado.
Flujo Principal	1. El coordinador está en la página principal 2. Pulsa en "Participantes" 3. El coordinador pulsa en la carpeta de materiales de un participante 4. El sistema muestra la lista de materiales 5. Escribe un texto de búsqueda 6. Se muestra una lista con los resultados devueltos por el filtro.

Figura 6.46: Caso de uso M08

7. Descargar un material de los que se han enviado a un participante siendo Coordinador

Caso de uso	Descargar material
Identificador	CU_M_07
Objetivo en contexto	Descargar el material seleccionado de la lista
Actores	Participante, Coordinador
Entidades que usa	Materiales, Participante, Coordinador
Precondiciones	1. Que el coordinador haya accedido a la aplicación.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Se descarga el material correctamente.
Flujo Principal	<ol style="list-style-type: none"> 1. El coordinador está en la página principal 2. Pulsa en "Participantes" 3. El coordinador pulsa en la carpeta de materiales de un participante 4. El sistema muestra la lista de materiales 5. Pulsa en el símbolo de Descargar de algún material. 6. Se descarga el material.

Figura 6.47: Caso de uso M07

8. Eliminar un material de los que se han enviado a un participante siendo Coordinador

Caso de uso	Eliminar material
Identificador	CU_M_06
Objetivo en contexto	Eliminar el material seleccionado de la lista
Actores	Participante, Coordinador
Entidades que usa	Materiales, Participante, Coordinador
Precondiciones	1. Que el coordinador haya accedido a la aplicación.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Se elimina el material correctamente.
Flujo Principal	<ol style="list-style-type: none"> 1. El coordinador está en la página principal 2. Pulsa en "Participantes" 3. El coordinador pulsa en la carpeta de materiales de un participante 4. El sistema muestra la lista de materiales 5. Pulsa en el símbolo de Eliminar de algún material. 6. Se elimina el material.

Figura 6.48: Caso de uso M06

9. Enviar un material a un grupo en concreto siendo Coordinador

Caso de uso	Enviar un material
Identificador	CU_M_10
Objetivo en contexto	Enviar un material a un participante en concreto
Actores	Participante y Coordinador
Entidades que usa	Materiales, Participante, Coordinador
Precondiciones	1. Que el coordinador haya accedido a la aplicación.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Se muestran los materiales enviados.
Flujo Principal	<ol style="list-style-type: none"> 1. El coordinador está en la página principal 2. Pulsa en "Participantes" 3. El coordinador pulsa en la carpeta de expediente de un participante 4. El sistema muestra el expediente 5. Pulsa en "Materiales Grupales" 6. Accede a la lista de materiales del grupo 7. El coordinador rellena el formulario para enviar el material. 8. El sistema comprueba que los datos introducidos son correctos. 9. El sistema añade el material a la lista. 10. Se muestra una lista con los materiales
Flujo Secundario	<ol style="list-style-type: none"> 2. a) Pulsa en "Grupos" 3. a) Pulsa en el botón de Materiales. 4. a) Salto a paso 6. del flujo principal. 8. a) El sistema detecta un error en los datos introducidos. 8. b) El sistema pide que cumplimenten bien esos datos.

Figura 6.49: Caso de uso M10

10. Ver los materiales enviados a un grupo en concreto siendo Coordinador

Caso de uso	Ver materiales
Identificador	CU_M_09
Objetivo en contexto	Ver materiales de un grupo en concreto
Actores	Participante y Coordinador
Entidades que usa	Materiales, Participante, Coordinador
Precondiciones	1. Que el coordinador haya accedido a la aplicación.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Aparece una lista con los materiales y acceso a las funcionalidades que extiende.
Flujo Principal	<ol style="list-style-type: none"> 1. El coordinador está en la página principal 2. Pulsa en "Participantes" 3. El coordinador pulsa en la carpeta de expediente de un participante 4. El sistema muestra el expediente 5. Pulsa en "Materiales Grupales" 6. Accede a la lista de materiales del grupo
Flujo Secundario	<ol style="list-style-type: none"> 2. a) Pulsa en "Grupos" 3. a) Pulsa en el botón de Materiales. 4. a) Salto a paso 6. del flujo principal.

Figura 6.50: Caso de uso M09

11. Filtrar los materiales enviados a un grupo en concreto siendo Coordinador

Caso de uso	Filtrar material
Identificador	CU_M_13
Objetivo en contexto	Buscar el material indicado
Actores	Participante, Materiales, Coordinador
Entidades que usa	Participante, Materiales, Coordinador
Precondiciones	1. Que el coordinador haya accedido a la aplicación.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Se muestran los materiales con el filtro indicado.
Flujo Principal	<ol style="list-style-type: none"> 1. El coordinador está en la página principal 2. Pulsa en "Participantes" 3. El coordinador pulsa en la carpeta de expediente de un participante 4. El sistema muestra el expediente 5. Pulsa en "Materiales Grupales" 6. Accede a la lista de materiales del grupo 7. Escribe un texto de búsqueda 8. Se muestra una lista con los resultados devueltos por el filtro.
Flujo Secundario	<ol style="list-style-type: none"> 2. a) Pulsa en "Grupos" 3. a) Pulsa en el botón de Materiales. 4. a) Salto a paso 6. del flujo principal.

Figura 6.51: Caso de uso M13

12. Descargar un material de los que se han enviado a un grupo siendo Coordinador

Caso de uso	Descargar material
Identificador	CU_M_12
Objetivo en contexto	Descargar el material seleccionado de la lista
Actores	Participante, Coordinador
Entidades que usa	Materiales, Participante, Coordinador
Precondiciones	1. Que el coordinador haya accedido a la aplicación.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Se descarga el material correctamente.
Flujo Principal	<ol style="list-style-type: none"> 1. El coordinador está en la página principal 2. Pulsa en "Participantes" 3. El coordinador pulsa en la carpeta de expediente de un participante 4. El sistema muestra el expediente 5. Pulsa en "Materiales Grupales" 6. Accede a la lista de materiales del grupo 5. Pulsa en el símbolo de Descargar de algún material. 6. Se descarga el material.
Flujo Secundario	<ol style="list-style-type: none"> 2. a) Pulsa en "Grupos" 3. a) Pulsa en el botón de Materiales. 4. a) Salto a paso 6. del flujo principal.

Figura 6.52: Caso de uso M12

13. Eliminar un material de los que se han enviado a un grupo siendo Coordinador

Caso de uso	Eliminar material
Identificador	CU_M_11
Objetivo en contexto	Eliminar el material seleccionado de la lista
Actores	Participante, Coordinador
Entidades que usa	Materiales, Participante, Coordinador
Precondiciones	1. Que el coordinador haya accedido a la aplicación.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Se elimina el material correctamente.
Flujo Principal	<ol style="list-style-type: none"> 1. El coordinador está en la página principal 2. Pulsa en "Participantes" 3. El coordinador pulsa en la carpeta de expediente de un participante 4. El sistema muestra el expediente 5. Pulsa en "Materiales Grupales" 6. Accede a la lista de materiales del grupo 7. Pulsa en el símbolo de Eliminar de algún material. 8. Se elimina el material.
Flujo Secundario	<ol style="list-style-type: none"> 2. a) Pulsa en "Grupos" 3. a) Pulsa en el botón de Materiales. 4. a) Salto a paso 6. del flujo principal.

Figura 6.53: Caso de uso M11

6.3.4. Progreso

1. Acceder a progreso propio siendo Participante

Caso de uso	Acceder a progreso
Identificador	CU_PG_01
Objetivo en contexto	Acceder a progreso propio
Actores	Participante
Entidades que usa	Progreso, Participante
Precondiciones	<ol style="list-style-type: none"> 1. Que el participante esté en estado de Alta. 2. Que el participante haya accedido a la aplicación.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Aparecen los datos del progreso.
Flujo Principal	<ol style="list-style-type: none"> 1. El participante está en la página principal 2. Pulsa en "Progreso" 3. El sistema muestra el dashboard de progreso.

Figura 6.54: Caso de uso PG01

2. Ver gráfica de peso siendo Participante

Caso de uso	Ver gráfica de peso
Identificador	CU_PG_03
Objetivo en contexto	Ver el progreso del peso en una gráfica.
Actores	Participante
Entidades que usa	Progreso, Participante
Precondiciones	1. Que el participante esté en estado de Alta. 2. Que el participante se encuentre en la vista de Progreso.
Postcondiciones	Fallo: Se devuelve la vista de Progreso. Éxito: Aparece la gráfica con los datos del peso.
Flujo Principal	1. Pulsa en Ver gráfica del Peso. 2. El sistema muestra la gráfica del progreso.

Figura 6.55: Caso de uso PG03

3. Filtrar peso siendo Participante

Caso de uso	Filtrar el peso
Identificador	CU_PG_13
Objetivo en contexto	Filtrar el peso por fecha
Actores	Participante
Entidades que usa	Progreso, Participante
Precondiciones	1. Que el participante esté en estado de Alta. 2. Que el participante se encuentre en la gráfica del peso de la vista de Progreso.
Postcondiciones	Fallo: Se devuelve la vista de Progreso. Éxito: Se muestran los datos de peso del intervalo de fechas indicado.
Flujo Principal	1. El participante se encuentra en la gráfica del peso. 2. Selecciona el intervalo de fechas. 3. El sistema muestra los datos del peso obtenidos del intervalo.

Figura 6.56: Caso de uso PG13

4. Ver registros de peso siendo Participante

Caso de uso	Ver registros de peso
Identificador	CU_PG_04
Objetivo en contexto	Ver el progreso del peso con datos
Actores	Participante
Entidades que usa	Progreso, Participante
Precondiciones	1. Que el participante esté en estado de Alta. 2. Que el participante se encuentre en la vista de Progreso.
Postcondiciones	Fallo: Se devuelve la vista de Progreso. Éxito: Aparece la tabla con los datos del peso.
Flujo Principal	1. Pulsa en Ver Registros del Peso. 2. El sistema muestra los registros del progreso.

Figura 6.57: Caso de uso PG04

5. Registrar peso siendo Participante

Caso de uso	Registrar el peso
Identificador	CU_PG_05
Objetivo en contexto	Registrar el peso en una fecha concreta
Actores	Participante
Entidades que usa	Progreso, Participante
Precondiciones	1. Que el participante esté en estado de Alta. 2. Que el participante se encuentre en la sección del peso de la vista de Progreso.
Postcondiciones	Fallo: Se devuelve la vista de Progreso. Éxito: Se añade el dato del peso correctamente y se redirige a la vista de Progreso.
Flujo Principal	1. El participante pulsa en el icono de Añadir (+). 2. El sistema muestra el formulario. 3. El participante rellena la fecha y el peso. 4. Pulsa en Registrar. 5. El sistema comprueba que los datos introducidos son correctos. 6. El sistema añade el dato de peso a la base de datos.
Flujo Secundario	6. a) El sistema detecta un error en los datos introducidos. 6. b) El sistema pide que se rellenen bien los datos.

Figura 6.58: Caso de uso PG05

6. Ver gráfica del perímetro siendo Participante

Caso de uso	Ver gráfica del perímetro
Identificador	CU_PG_06
Objetivo en contexto	Ver el progreso del perímetro en una gráfica.
Actores	Participante
Entidades que usa	Progreso, Participante
Precondiciones	1. Que el participante esté en estado de Alta. 2. Que el participante se encuentre en la vista de Progreso.
Postcondiciones	Fallo: Se devuelve la vista de Progreso. Éxito: Aparece la gráfica con los datos del perímetro.
Flujo Principal	1. Pulsa en Ver gráfica del Perímetro. 2. El sistema muestra la gráfica del perímetro.

Figura 6.59: Caso de uso PG06

7. Filtrar perímetro siendo Participante

Caso de uso	Filtrar el perímetro
Identificador	CU_PG_14
Objetivo en contexto	Filtrar el perímetro por fecha
Actores	Participante
Entidades que usa	Progreso, Participante
Precondiciones	1. Que el participante esté en estado de Alta. 2. Que el participante se encuentre en la gráfica del perímetro de la vista de Progreso.
Postcondiciones	Fallo: Se devuelve la vista de Progreso. Éxito: Se muestran los datos de perímetro del intervalo de fechas indicado.
Flujo Principal	1. El participante se encuentra en la gráfica del perímetro. 2. Selecciona el intervalo de fechas. 3. El sistema muestra los datos del perímetro obtenidos del intervalo.

Figura 6.60: Caso de uso PG14

8. Ver registros de perímetro siendo Participante

Caso de uso	Ver registros de perímetro
Identificador	CU_PG_07
Objetivo en contexto	Ver el progreso del perímetro con datos
Actores	Participante
Entidades que usa	Progreso, Participante
Precondiciones	1. Que el participante esté en estado de Alta. 2. Que el participante se encuentre en la vista de Progreso.
Postcondiciones	Fallo: Se devuelve la vista de Progreso. Éxito: Aparece la tabla con los datos del perímetro.
Flujo Principal	1. Pulsa en Ver Registros del Perímetro. 2. El sistema muestra los registros del perímetro.

Figura 6.61: Caso de uso PG07

9. Registrar perímetro siendo Participante

Caso de uso	Registrar el perímetro
Identificador	CU_PG_08
Objetivo en contexto	Registrar el perímetro en una fecha concreta
Actores	Participante
Entidades que usa	Progreso, Participante
Precondiciones	1. Que el participante esté en estado de Alta. 2. Que el participante se encuentre en la sección del perímetro de la vista de Progreso.
Postcondiciones	Fallo: Se devuelve la vista de Progreso. Éxito: Se añade el dato del perímetro correctamente y se redirige a la vista de Progreso.
Flujo Principal	1. El participante pulsa en el icono de Añadir (+). 2. El sistema muestra el formulario. 3. El participante rellena la fecha y el perímetro. 4. Pulsa en Registrar. 5. El sistema comprueba que los datos introducidos son correctos. 6. El sistema añade el dato de perímetro a la base de datos.
Flujo Secundario	6. a) El sistema detecta un error en los datos introducidos. 6. b) El sistema pide que se rellenen bien los datos.

Figura 6.62: Caso de uso PG08

10. Acceder al progreso de un participante siendo Coordinador

Caso de uso	Acceder al progreso
Identificador	CU_PG_02
Objetivo en contexto	Acceder al progreso de un participante
Actores	Coordinador, Participante
Entidades que usa	Progreso, Participante, Coordinador
Precondiciones	1. El coordinador debe estar en la vista de expediente de un participante.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Se devuelve la vista del Progreso individual.
Flujo Principal	1. El coordinador pulsa en Progreso. 3. El sistema muestra el dashboard de progreso.

Figura 6.63: Caso de uso PG02

11. Ver gráfica de peso siendo Coordinador

Caso de uso	Ver gráfica de peso
Identificador	CU_PG_09
Objetivo en contexto	Ver el progreso del peso de un participante en una gráfica.
Actores	Coordinador, Participante
Entidades que usa	Progreso, Coordinador, Participante
Precondiciones	1. Que el coordinador se encuentre en la vista de Progreso individual.
Postcondiciones	Fallo: Se devuelve la vista de Progreso individual. Éxito: Aparece la gráfica con los datos del peso.
Flujo Principal	1. Pulsa en Ver gráfica del Peso. 2. El sistema muestra la gráfica del progreso.

Figura 6.64: Caso de uso PG09

12. Ver registros de peso siendo Coordinador

Caso de uso	Ver registros de peso
Identificador	CU_PG_10
Objetivo en contexto	Ver el progreso del peso de un participante con datos.
Actores	Coordinador, Participante
Entidades que usa	Progreso, Coordinador, Participante
Precondiciones	1. Que el coordinador se encuentre en la vista de Progreso individual.
Postcondiciones	Fallo: Se devuelve la vista de Progreso individual. Éxito: Aparece la tabla con los datos del peso.
Flujo Principal	1. Pulsa en Ver Registros del Peso. 2. El sistema muestra los registros del progreso.

Figura 6.65: Caso de uso PG10

13. Ver gráfica de perímetro siendo Coordinador

Caso de uso	Ver gráfica de perímetro
Identificador	CU_PG_11
Objetivo en contexto	Ver el progreso del perímetro de un participante en una gráfica.
Actores	Coordinador, Participante
Entidades que usa	Progreso, Coordinador, Participante
Precondiciones	1. Que el coordinador se encuentre en la vista de Progreso individual.
Postcondiciones	Fallo: Se devuelve la vista de Progreso. Éxito: Aparece la gráfica con los datos del perímetro.
Flujo Principal	1. Pulsa en Ver gráfica del Perímetro. 2. El sistema muestra la gráfica del progreso.

Figura 6.66: Caso de uso PG11

14. Ver registros de perímetro siendo Coordinador

Caso de uso	Ver registros de perímetro
Identificador	CU_PG_12
Objetivo en contexto	Ver el progreso del perímetro de un participante con datos.
Actores	Coordinador, Participante
Entidades que usa	Progreso, Coordinador, Participante
Precondiciones	1. Que el coordinador se encuentre en la vista de Progreso individual.
Postcondiciones	Fallo: Se devuelve la vista de Progreso individual. Éxito: Aparece la tabla con los datos del perímetro.
Flujo Principal	1. Pulsa en Ver Registros del Perímetro. 2. El sistema muestra los registros del progreso.

Figura 6.67: Caso de uso PG12

6.3.5. Chat

1. Acceder al chat privado con el coordinador siendo participante

Caso de uso	Acceder al chat con coordinador
Identificador	CU_CH_01
Objetivo en contexto	Acceder al chat privado con el coordinador del participante.
Actores	Participante, Coordinador
Entidades que usa	MensajePrivado, Participante
Precondiciones	1. Que el participante esté en estado de Alta. 2. Que el participante se encuentre en la vista de chat.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Aparece la vista con el chat privado con el coordinador actualizada
Flujo Principal	1. El participante pulsa en Chat coordinador. 2. El sistema muestra el chat privado con el coordinador

Figura 6.68: Caso de uso CH01

2. Acceder al chat privado con el administrador siendo participante

Caso de uso	Acceder al chat con administrador
Identificador	CU_CH_02
Objetivo en contexto	Acceder al chat de ayuda con el administrador de la aplicación.
Actores	Participante, Administrador
Entidades que usa	Participante, Administrador, MensajeAdmin
Precondiciones	1. Que el participante esté en estado de Alta. 2. Que el participante haya accedido a la aplicación.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Aparece la vista con el chat privado con el administrador actualizada
Flujo Principal	1. El participante pulsa en Ayuda. 2. El sistema muestra la conversación con el administrador

Figura 6.69: Caso de uso CH02

3. Acceder al chat grupal siendo participante

Caso de uso	Acceder al chat grupal
Identificador	CU_CH_03
Objetivo en contexto	Acceder al chat del grupo del participante
Actores	Participante
Entidades que usa	Participante, Mensaje
Precondiciones	1. Que el participante esté en estado de Alta. 2. Que el participante se encuentre en la vista de chat
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Aparece la vista con el chat grupal actualizada
Flujo Principal	1. El participante pulsa en Chat grupal. 2. El sistema muestra el chat grupal actualizado

Figura 6.70: Caso de uso CH03

4. Acceder al chat privado con el administrador siendo coordinador

Caso de uso	Acceder al chat con administrador
Identificador	CU_CH_04
Objetivo en contexto	Acceder al chat de ayuda con el administrador de la aplicación.
Actores	Coordinador, Administrador
Entidades que usa	Coordinador, Administrador, MensajeAdmin
Precondiciones	1. Que el coordinador esté en estado de Alta. 2. Que el coordinador haya accedido a la aplicación.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Aparece la vista con el chat privado con el administrador actualizada
Flujo Principal	1. El coordinador pulsa en Ayuda. 2. El sistema muestra la conversación con el administrador

Figura 6.71: Caso de uso CH04

5. Acceder al chat privado con un participante siendo coordinador

Caso de uso	Acceder al chat con un participante concreto
Identificador	CU_CH_05
Objetivo en contexto	Acceder al chat con un participante concreto de los gestionados por el coordinador.
Actores	Coordinador, Participante
Entidades que usa	Coordinador, Participante, Mensaje, Grupo
Precondiciones	1. Que el coordinador esté en estado de Alta. 2. Que el coordinador se encuentre en la vista de un grupo.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Aparece la vista con el chat privado con el participante seleccionado actualizada
Flujo Principal	1. El coordinador pulsa en Mensaje sobre el participante. con el que desea hablar de la lista de participantes del grupo. 2. El sistema muestra la conversación con el participante.

Figura 6.72: Caso de uso CH05

6. Acceder al chat de un grupo siendo coordinador

Caso de uso	Acceder al chat de un grupo
Identificador	CU_CH_06
Objetivo en contexto	Acceder al chat de un grupo de los que lleva el coordinador.
Actores	Coordinador
Entidades que usa	Coordinador, Mensaje
Precondiciones	1. Que el coordinador esté en estado de Alta. 2. Que el coordinador se encuentre en la vista de un grupo
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Aparece la vista con el chat grupal actualizado
Flujo Principal	1. El sistema muestra la conversación grupal.

Figura 6.73: Caso de uso CH06

7. Enviar mensaje rápido a un usuario concreto siendo administrador

Caso de uso	Enviar mensaje rápido
Identificador	CU_CH_07
Objetivo en contexto	Enviar un mensaje rápido a un usuario guardándolo en la base de datos.
Actores	Administrador
Entidades que usa	Administrador, Usuario, MensajeAdmin
Precondiciones	1. Que el administrador esté en estado de Alta. 2. Que el administrador se encuentre en la vista del listado de usuarios.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Se guarda el mensaje en la base de datos y se redirige al listado de usuarios
Flujo Principal	1. El administrador selecciona el icono de enviar mensaje correspondiente al usuario con el que quiere conversar. 2. El sistema muestra el formulario. 3. El administrador introduce un mensaje. 4. El administrador pulsa Enviar. 5. El sistema comprueba que los datos introducidos son correctos. 6. Se envía el mensaje. 7. El sistema redirige a la vista del listado de usuarios.
Flujo Secundario	5. a) El sistema detecta un error en los datos introducidos. 5. b) El sistema pide que se rellene el mensaje.

Figura 6.74: Caso de uso CH07

8. Acceder a las conversaciones establecidas siendo administrador

Caso de uso	Acceder a las conversaciones
Identificador	CU_CH_08
Objetivo en contexto	Acceder a todas las conversaciones mantenidas por el administrador con usuarios de la aplicación.
Actores	Administrador
Entidades que usa	Usuario, MensajeAdmin
Precondiciones	1. Que el administrador esté en estado de Alta. 2. Que el administrador haya accedido a la aplicación.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Aparece la vista con la lista de conversaciones del administrador
Flujo Principal	1. El administrador pulsa en Mensajes. 1. El sistema muestra la vista de las conversaciones del administrador.

Figura 6.75: Caso de uso CH08

9. Enviar mensaje al coordinador siendo participante

Caso de uso	Enviar mensaje al coordinador
Identificador	CU_CH_09
Objetivo en contexto	Enviar un mensaje al coordinador guardándolo en la base de datos.
Actores	Participante, Coordinador
Entidades que usa	Participante, MensajePrivado
Precondiciones	<ul style="list-style-type: none"> 1. Que el participante esté en estado de Alta. 2. Que el participante se encuentre en la vista de chat.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Se guarda el mensaje en la base de datos
Flujo Principal	<ul style="list-style-type: none"> 1. El participante introduce un mensaje. 2. El participante pulsa enter o el icono de enviar. 3. El sistema comprueba que los datos introducidos son adecuados. 4. Se envía el mensaje. 5. El sistema redirige a la vista del chat.
Flujo Secundario	<ul style="list-style-type: none"> 3. a) El sistema detecta un error o palabra bloqueada en los datos introducidos. 3. b) El sistema pide que se rellene de nuevo el mensaje.

Figura 6.76: Caso de uso CH09

10. Enviar mensaje al administrador siendo participante

Caso de uso	Enviar mensaje al administrador
Identificador	CU_CH_10
Objetivo en contexto	Enviar un mensaje de ayuda al administrador de la aplicación.
Actores	Participante, Administrador
Entidades que usa	MensajeAdmin, Usuario
Precondiciones	1. Que el participante esté en estado de Alta. 2. Que el participante se encuentre en la vista de chatPartAdmin o de solicitar de ayuda.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Se guarda el mensaje en la base de datos
Flujo Principal	1. El participante introduce un mensaje. 2. El participante pulsa enter o el icono de enviar. 3. El sistema comprueba que los datos introducidos son adecuados. 4. Se envía el mensaje.
Flujo Secundario	3. a) El sistema detecta un error o palabra bloqueada en los datos introducidos. 3. b) El sistema pide que se rellene de nuevo el mensaje.

Figura 6.77: Caso de uso CH10

11. Imprimir el chat privado con el administrador siendo participante

Caso de uso	Imprimir chat de ayuda
Identificador	CU_CH_11
Objetivo en contexto	Imprimir la conversación de ayuda mantenida con el administrador.
Actores	Participante, Administrador
Entidades que usa	MensajeAdmin, Usuario
Precondiciones	1. Que el participante esté en estado de Alta. 2. Que el participante se encuentre en la vista de chatPartAdmin o de solicitar de ayuda.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Se devuelva la vista con la conversación y una interfaz para imprimir.
Flujo Principal	1. El participante pulsa en Imprimir.

Figura 6.78: Caso de uso CH11

12. Enviar mensaje al chat grupal siendo participante

Caso de uso	Enviar mensaje al chat grupal
Identificador	CU_CH_12
Objetivo en contexto	Enviar un mensaje al chat del grupo del participante guardándolo en la base de datos.
Actores	Participante
Entidades que usa	Participante, Grupo, Mensaje
Precondiciones	<ol style="list-style-type: none"> 1. Que el participante esté en estado de Alta. 2. Que el participante se encuentre en la vista de chat o mi grupo.
Postcondiciones	<p>Fallo: Se devuelve la vista principal.</p> <p>Éxito: Se guarda el mensaje en la base de datos.</p>
Flujo Principal	<ol style="list-style-type: none"> 1. El participante introduce un mensaje. 2. El participante pulsa enter o el icono de enviar. 3. El sistema comprueba que los datos introducidos son adecuados. 4. Se envía el mensaje.
Flujo Secundario	<ol style="list-style-type: none"> 3. a) El sistema detecta un error o palabra bloqueada en los datos introducidos. 3. b) El sistema pide que se rellene de nuevo el mensaje.

Figura 6.79: Caso de uso CH12

13. Enviar mensaje al administrador siendo coordinador

Caso de uso	Enviar mensaje al administrador
Identificador	CU_CH_13
Objetivo en contexto	Enviar un mensaje de ayuda al administrador de la aplicación.
Actores	Coordinador, Administrador
Entidades que usa	MensajeAdmin, Usuario
Precondiciones	<ol style="list-style-type: none"> 1. Que el coordinador esté en estado de Alta. 2. Que el coordinador se encuentre en la vista de chatCordAdmin o de solicitar de ayuda.
Postcondiciones	<p>Fallo: Se devuelve la vista principal.</p> <p>Éxito: Se guarda el mensaje en la base de datos</p>
Flujo Principal	<ol style="list-style-type: none"> 1. El coordinador introduce un mensaje. 2. El coordinador pulsa enter o el icono de enviar. 3. El sistema comprueba que los datos introducidos son adecuados. 4. Se envía el mensaje.
Flujo Secundario	<ol style="list-style-type: none"> 3. a) El sistema detecta un error o palabra bloqueada en los datos introducidos. 3. b) El sistema pide que se rellene de nuevo el mensaje.

Figura 6.80: Caso de uso CH13

14. Imprimir el chat privado con el administrador siendo coordinador

Caso de uso	Imprimir chat de ayuda
Identificador	CU_CH_14
Objetivo en contexto	Imprimir la conversación de ayuda mantenida con el administrador.
Actores	Coordinador, Administrador
Entidades que usa	MensajeAdmin, Usuario
Precondiciones	1. Que el coordinador esté en estado de Alta. 2. Que el coordinador se encuentre en la vista de chatCordAdmin o de solicitar de ayuda.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Se devuelva la vista con la conversación y una interfaz para imprimir.
Flujo Principal	1. El coordinador pulsa en Imprimir.

Figura 6.81: Caso de uso CH14

15. Enviar mensaje a un participante siendo coordinador

Caso de uso	Enviar mensaje a participante
Identificador	CU_CH_15
Objetivo en contexto	Enviar un mensaje a un participante concreto guardándolo en la base de datos.
Actores	Coordinador, Participante
Entidades que usa	Participante, Usuario, MensajePrivado
Precondiciones	1. Que el coordinador esté en estado de Alta. 2. Que el coordinador se encuentre en la vista de chat privado con un participante.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Se guarda el mensaje en la base de datos.
Flujo Principal	1. El coordinador introduce un mensaje. 2. El coordinador pulsa enter o el icono de enviar. 3. El sistema comprueba que los datos introducidos son adecuados. 4. Se envía el mensaje.
Flujo Secundario	3. a) El sistema detecta un error o palabra bloqueada en los datos introducidos. 3. b) El sistema pide que se rellene de nuevo el mensaje.

Figura 6.82: Caso de uso CH15

16. Imprimir el chat privado con un participante siendo coordinador

Caso de uso	Imprimir chat privado con participante
Identificador	CU_CH_16
Objetivo en contexto	Imprimir la conversación privada mantenida con un participante.
Actores	Coordinador, Participante
Entidades que usa	MensajePrivado, Usuario
Precondiciones	1. Que el coordinador esté en estado de Alta. 2. Que el coordinador se encuentre en la vista de chatPrivado o conversación con un participante.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Se devuelva la vista con la conversación y una interfaz para imprimir.
Flujo Principal	1. El coordinador pulsa en Imprimir.

Figura 6.83: Caso de uso CH16

17. Enviar mensaje al chat de un grupo siendo coordinador

Caso de uso	Enviar mensaje al chat de un grupo
Identificador	CU_CH_17
Objetivo en contexto	Enviar un mensaje a un grupo concreto guardándolo en la base de datos.
Actores	Coordinador
Entidades que usa	Usuario, Grupo, Mensaje
Precondiciones	1. Que el coordinador esté en estado de Alta. 2. Que el coordinador se encuentre en la vista de un grupo concreto.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Se guarda el mensaje en la base de datos.
Flujo Principal	1. El coordinador introduce un mensaje. 2. El coordinador pulsa enter o el icono de enviar. 3. El sistema comprueba que los datos introducidos son adecuados. 4. Se envía el mensaje.
Flujo Secundario	3. a) El sistema detecta un error o palabra bloqueada en los datos introducidos. 3. b) El sistema pide que se rellene de nuevo el mensaje.

Figura 6.84: Caso de uso CH17

18. Enviar mensaje a un usuario concreto siendo administrador

Caso de uso	Enviar mensaje a un usuario
Identificador	CU_CH_18
Objetivo en contexto	Enviar un mensaje a un usuario concreto guardándolo en la base de datos.
Actores	Administrador
Entidades que usa	Usuario, MensajeAdmin
Precondiciones	1. Que el administrador esté en estado de Alta. 2. Que el administrador se encuentre en la vista de mensajesAdmin con todas sus conversaciones.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Se guarda el mensaje en la base de datos.
Flujo Principal	1. El administrador introduce un mensaje. 2. El administrador pulsa enter o el icono de enviar. 3. El sistema comprueba que los datos introducidos son adecuados. 4. Se envía el mensaje.
Flujo Secundario	3. a) El sistema detecta un error o palabra bloqueada en los datos introducidos. 3. b) El sistema pide que se rellene de nuevo el mensaje.

Figura 6.85: Caso de uso CH18

6.3.6. Alimentación

1. Crear un nuevo alimento siendo Participante

Caso de uso	Crear nuevo alimento
Identificador	CU_AL_16
Objetivo en contexto	Añadir un nuevo alimento a la base de datos.
Actores	Participante
Entidades que usa	Alimentación, Participante
Precondiciones	1. Que el participante esté en estado de Alta. 2. Que el participante se encuentre en la vista de Alimentos.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Se crea el alimento correctamente y se redirige a la vista de Alimentos.
Flujo Principal	1. El participante pulsa en crear alimento. 2. El sistema muestra el formulario. 3. El participante rellena el formulario para crear el alimento 4. El sistema comprueba que los datos introducidos son correctos. 5. El sistema añade el alimento a la base de datos.
Flujo Secundario	5. a) El sistema detecta un error en los datos introducidos. 5. b) El sistema pide que cumplimenten bien esos datos.

Figura 6.86: Caso de uso AL16

2. Ver los alimentos y nutrientes que se han consumido durante el día siendo Participante

Caso de uso	Ver alimentos
Identificador	CU_AL_02
Objetivo en contexto	Ver los alimentos y nutrientes consumidos.
Actores	Participante
Entidades que usa	Alimentación, Participante, Alimentos Consumidos
Precondiciones	1. Que el participante esté en estado de Alta. 2. Que el participante haya accedido a la aplicación.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Se muestran los alimentos consumidos y nutrientes.
Flujo Principal	1. El participante pulsa en Alimentación. 2. El participante pulsa en Alimentos.

Figura 6.87: Caso de uso AL02

3. Añadir alimentos que haya consumido durante el día siendo Participante

Caso de uso	Añadir consumición
Identificador	CU_AL_14
Objetivo en contexto	Añadir un alimento consumido.
Actores	Participante
Entidades que usa	Alimentación, Participante, Alimento Consumido
Precondiciones	1. Que el participante esté en estado de Alta. 2. Que el participante se encuentre en la vista de Alimentos.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Se añade el alimento a la lista.
Flujo Principal	1. El participante rellena el número de raciones. 2. El sistema comprueba que los datos sean correctos. 3. El participante busca el alimento a añadir. 4. El participante pulsa en agregar.
Flujo Secundario	2. a) El sistema detecta un error en los datos introducidos. 2. b) El sistema pide que cumplimenten bien esos datos.

Figura 6.88: Caso de uso AL14

4. Eliminar alimentos que haya consumido

Caso de uso	Eliminar consumición
Identificador	CU_AL_15
Objetivo en contexto	Eliminar un alimento consumido.
Actores	Participante
Entidades que usa	Alimentación, Participante, Alimento Consumido
Precondiciones	1. Que el participante esté en estado de Alta. 2. Que el participante se encuentre en la vista de Alimentos.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Se elimina el alimento de la lista.
Flujo Principal	1. El participante pulsa el ícono de basura del alimento a borrar. 2. El sistema elimina el alimento.

Figura 6.89: Caso de uso AL15

5. Ver todas las recetas existentes siendo Participante

Caso de uso	Ver recetas
Identificador	CU_AL_01
Objetivo en contexto	Ver las recetas que hay en la base de datos.
Actores	Participante
Entidades que usa	Alimentación, Participante, Receta
Precondiciones	1. Que el participante esté en estado de Alta. 2. Que el participante haya accedido a la aplicación.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Se muestran las recetas existentes.
Flujo Principal	1. El participante pulsa en Alimentación. 2. El participante pulsa en Recetas.

Figura 6.90: Caso de uso AL01

6. Filtrar las recetas existentes siendo Participante

Caso de uso	Filtrar receta
Identificador	CU_AL_04
Objetivo en contexto	Buscar la receta indicada
Actores	Participante
Entidades que usa	Participante, Receta, Alimentación
Precondiciones	1. Que el participante haya accedido a la aplicación.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Se muestran las recetas con el filtro indicado.
Flujo Principal	1. El participante está en la página principal 2. Pulsa en "Alimentación" 3. Pulsa en "Recetas" 4. El sistema muestra las recetas 5. Escribe un texto de búsqueda 6. Selecciona los ingredientes que quiere que estén 7. Selecciona los ingredientes que no quiere 8. Se muestra una lista con los resultados devueltos por el filtro.
Flujo Secundario	8. a) Si no hay resultados muestra botón para ver recetas parecidas

Figura 6.91: Caso de uso AL04

7. Crear una nueva receta siendo Participante

Caso de uso	Crear nueva receta
Identificador	CU_AL_05
Objetivo en contexto	Añadir una nueva receta a la base de datos.
Actores	Participante
Entidades que usa	Alimentación, Participante, Receta
Precondiciones	1. Que el participante esté en estado de Alta. 2. Que el participante se encuentre en la vista de Recetas.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Se crea el alimento correctamente y se redirige a la vista de Alimentos.
Flujo Principal	1. El participante rellena el formulario para crear la receta 4. El sistema comprueba que los datos introducidos son correctos. 5. El sistema añade la receta a la base de datos.
Flujo Secundario	1. a) El participante pulsa en Crear nueva receta 1. b) El sistema muestra el formulario. 5. a) El sistema detecta un error en los datos introducidos. 5. b) El sistema pide que cumplimenten bien esos datos.

Figura 6.92: Caso de uso AL05

8. Ver una receta en concreto siendo Participante

Caso de uso	Ver receta
Identificador	CU_AL_06
Objetivo en contexto	Ver una receta en concreto.
Actores	Participante
Entidades que usa	Alimentación, Participante, Receta
Precondiciones	1. Que el participante esté en estado de Alta. 2. Que el participante se encuentre en la vista de Recetas.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Se redirige a la vista con la receta en concreto.
Flujo Principal	1. El participante pulsa en el icono de ver la receta 4. El sistema le redirige a la vista con la receta en concreto.

Figura 6.93: Caso de uso AL06

9. Ver recetas con ingredientes parecidos a los que quería cuando el filtro no encuentre ninguna siendo Participante

Caso de uso	Ver parecidas
Identificador	CU_AL_08
Objetivo en contexto	Ver una lista de recetas parecidas.
Actores	Participante
Entidades que usa	Alimentación, Participante, Receta
Precondiciones	1. Que el participante esté en estado de Alta. 2. Que el participante se encuentre en la vista de Recetas. 3. Que el filtro de recetas no haya devuelto ningún resultado.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Se redirige a la vista con las recetas parecidas.
Flujo Principal	1. El participante pulsa en el botón de ver recetas parecidas 4. El sistema le redirige a la vista con las recetas parecidas.

Figura 6.94: Caso de uso AL08

10. Filtrar las recetas parecidas siendo Participante

Caso de uso	Filtrar receta
Identificador	CU_AL_10
Objetivo en contexto	Buscar la receta indicada
Actores	Participante
Entidades que usa	Participante, Receta, Alimentación
Precondiciones	1. Que el participante haya accedido a la aplicación. 2. Que el participante se encuentre en la vista de recetas parecidas.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Se muestran las recetas con el filtro indicado.
Flujo Principal	1. Escribe un texto de búsqueda 2. Se muestra una lista con los resultados devueltos por el filtro.

Figura 6.95: Caso de uso AL10

11. Ver una receta parecida en concreto siendo Participante

Caso de uso	Ver receta
Identificador	CU_AL_11
Objetivo en contexto	Ver una receta en concreto.
Actores	Participante
Entidades que usa	Alimentación, Participante, Receta
Precondiciones	1. Que el participante esté en estado de Alta. 2. Que el participante se encuentre en la vista de Recetas Parecidas.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Se redirige a la vista con la receta en concreto.
Flujo Principal	1. El participante pulsa en el ícono de ver la receta 4. El sistema le redirige a la vista con la receta en concreto.

Figura 6.96: Caso de uso AL11

12. Ver recomendaciones de recetas según el Participante

Caso de uso	Ver recomendadas
Identificador	CU_AL_09
Objetivo en contexto	Ver listas de recetas recomendadas.
Actores	Participante
Entidades que usa	Alimentación, Participante, Receta
Precondiciones	1. Que el participante esté en estado de Alta. 2. Que el participante se encuentre en la vista de Recetas.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Se redirige a la vista con las recetas recomendadas.
Flujo Principal	1. El participante pulsa en el botón de ver recomendaciones 4. El sistema le redirige a la vista con las recetas recomendadas.

Figura 6.97: Caso de uso AL09

13. Filtrar las recetas recomendadas siendo Participante

Caso de uso	Filtrar receta
Identificador	CU_AL_12
Objetivo en contexto	Buscar la receta indicada
Actores	Participante
Entidades que usa	Participante, Receta, Alimentación
Precondiciones	1. Que el participante haya accedido a la aplicación. 2. Que el participante se encuentre en la vista de recetas recomendadas.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Se muestran las recetas con el filtro indicado.
Flujo Principal	1. Escribe un texto de búsqueda 2. Se muestra una lista con los resultados devueltos por el filtro.

Figura 6.98: Caso de uso AL12

14. Ver una receta recomendada siendo Participante

Caso de uso	Ver receta
Identificador	CU_AL_13
Objetivo en contexto	Ver una receta en concreto.
Actores	Participante
Entidades que usa	Alimentación, Participante, Receta
Precondiciones	1. Que el participante esté en estado de Alta. 2. Que el participante se encuentre en la vista de Recetas Recomendadas.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Se redirige a la vista con la receta en concreto.
Flujo Principal	1. El participante pulsa en el ícono de ver la receta 4. El sistema le redirige a la vista con la receta en concreto.

Figura 6.99: Caso de uso AL13

15. Ver nutrientes en dieta siendo Participante

Caso de uso	Ver nutrientes
Identificador	CU_AL_07
Objetivo en contexto	Ver listas de nutrientes para la dieta.
Actores	Participante
Entidades que usa	Alimentación, Participante, Receta
Precondiciones	1. Que el participante esté en estado de Alta. 2. Que el participante se encuentre en la vista de Recetas.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Se redirige a la vista con los nutrientes.
Flujo Principal	1. El participante pulsa en el botón de nutrientes 2. El sistema le redirige a la vista con los nutrientes.

Figura 6.100: Caso de uso AL07

16. Calcular icdr siendo Participante

Caso de uso	Calcular icdr
Identificador	CU_AL_21
Objetivo en contexto	Calcular el índice calórico diario recomendado.
Actores	Participante
Entidades que usa	AlimentosConsumidos, Participante, Alimentación
Precondiciones	1. Que el participante esté en estado de Alta. 2. Que el participante se encuentre en la vista de Nutrientes.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Sale una lista con los nutrientes que le faltan.
Flujo Principal	1. El participante rellena el formulario para calcular el icdr 2. El sistema comprueba que los datos introducidos sean correctos.
Flujo Secundario	2. a) El sistema detecta un error en los datos introducidos. 2. b) El sistema pide al participante que cumplimente bien los datos.

Figura 6.101: Caso de uso AL21

17. Ver juego de intercambiar alimentos siendo Participante

Caso de uso	Ver juego
Identificador	CU_AL_03
Objetivo en contexto	Acceder al juego de intercambio de alimentos.
Actores	Participante
Entidades que usa	Alimentos Intercambio, Participante
Precondiciones	1. Que el participante esté en estado de Alta. 2. Que el participante haya accedido a la aplicación.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Se muestran la vista del juego.
Flujo Principal	1. El participante pulsa en Alimentación. 2. El participante pulsa en Juego.

Figura 6.102: Caso de uso AL03

18. Añadir alimentos y calorías siendo Participante

Caso de uso	Añadir alimentos
Identificador	CU_AL_19
Objetivo en contexto	Añadir alimentos al juego.
Actores	Participante
Entidades que usa	Alimentos Intercambio, Participante
Precondiciones	1. Que el participante esté en estado de Alta. 2. Que el participante se encuentre en la vista del juego.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Se muestran la vista del juego actualizada.
Flujo Principal	1. El participante arrastra un alimento a la tabla del juego.

Figura 6.103: Caso de uso AL19

19. Filtrar alimentos a añadir siendo Participante

Caso de uso	Filtrar alimentos
Identificador	CU_AL_20
Objetivo en contexto	Filtrar alimentos a añadir.
Actores	Participante
Entidades que usa	Alimentos Intercambio, Participante
Precondiciones	1. Que el participante esté en estado de Alta. 2. Que el participante se encuentre en la vista del juego.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Se muestran la vista del juego actualizada.
Flujo Principal	1. El participante escribe un texto de búsqueda 2. Se muestra la vista con los resultados devueltos por el filtro.

Figura 6.104: Caso de uso AL20

20. Comprobar que esté bien siendo Participante

Caso de uso	Filtrar alimentos
Identificador	CU_AL_17
Objetivo en contexto	Comprobar que los alimentos estén bien situados.
Actores	Participante
Entidades que usa	Alimentos Intercambio, Participante
Precondiciones	1. Que el participante esté en estado de Alta. 2. Que el participante se encuentre en la vista del juego.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Se muestran la vista del juego actualizada.
Flujo Principal	1. El participante pulsa en Comprobar 2. El sistema comprueba que los datos introducidos estén bien situados 3. El sistema actualiza la vista indicando los alimentos que están mal.

Figura 6.105: Caso de uso AL17

21. Leer instrucciones siendo Participante

Caso de uso	Leer instrucciones
Identificador	CU_AL_18
Objetivo en contexto	Leer las instrucciones del juego.
Actores	Participante
Entidades que usa	Alimentos Intercambio, Participante
Precondiciones	1. Que el participante esté en estado de Alta. 2. Que el participante se encuentre en la vista del juego.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Se muestran la vista con las instrucciones.
Flujo Principal	1. El participante pulsa en Instrucciones.

Figura 6.106: Caso de uso AL18

6.3.7. Objetivos

1. Acceder a tus objetivos siendo participante

Caso de uso	Acceder a tus objetivos
Identificador	CU_OB_01
Objetivo en contexto	Acceder a los objetivos recomendados y personalizados del participante.
Actores	Participante
Entidades que usa	Objetivo, ObjetivoAgua, ObjetivoEjercicio, Objetivodescanso, ObjetivoEstadoAnimo, Participante
Precondiciones	1. Que el participante esté en estado de Alta. 2. Que el participante haya accedido a la aplicación.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Aparecen las zonas de objetivos recomendados y personalizados con la información y acceso a los recursos correspondientes
Flujo Principal	1. El participante está en la página principal. 2. El participante pulsa en Objetivos. 3. El sistema muestra los objetivos

Figura 6.107: Caso de uso OB01

2. Ver tus objetivos cumplidos siendo participante

Caso de uso	Visualizar los objetivos cumplidos por el participante
Identificador	CU_OB_02
Objetivo en contexto	Visualizar los objetivos diarios cumplidos por el participante.
Actores	Participante
Entidades que usa	Objetivo, ObjetivoAgua, ObjetivoEjercicio, Objetivodescanso, ObjetivoEstadoAnimo, Participante
Precondiciones	1. Que el participante esté en estado de Alta. 2. Que el participante haya accedido a la aplicación.
Postcondiciones	Fallo: Se devuelve la vista principal sin objetivos cumplidos. Éxito: Aparece información sobre los objetivos diarios recomendados cumplidos por el participante indicando cuáles le falta completar
Flujo Principal	1. El participante está en la página principal. 2. El sistema muestra los objetivos diarios recomendados distinguiendo entre completados y no completados

Figura 6.108: Caso de uso OB02

3. Acceder al objetivo de agua recomendado siendo participante

Caso de uso	Acceder al objetivo diario de agua recomendado.
Identificador	CU_OB_03
Objetivo en contexto	Acceder a la zona del objetivo diario de agua recomendado para llevar un seguimiento y control.
Actores	Participante
Entidades que usa	Objetivo, ObjetivoAgua, Participante
Precondiciones	1. Que el participante esté en estado de Alta. 2. Que el participante se encuentre en la vista de Objetivos.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Aparece información sobre el consumo diario de agua del participante que puede actualizar y hacer un seguimiento
Flujo Principal	1. El participante pulsa en Objetivos 2. El sistema muestra la información actualizada sobre el consumo de agua diario del participante

Figura 6.109: Caso de uso OB03

4. Acceder al objetivo de ejercicio recomendado siendo participante

Caso de uso	Acceder al objetivo diario de ejercicio recomendado.
Identificador	CU_OB_04
Objetivo en contexto	Acceder a la zona del objetivo diario de ejercicio recomendado para llevar un seguimiento.
Actores	Participante
Entidades que usa	Objetivo, ObjetivoEjercicio, Participante
Precondiciones	1. Que el participante esté en estado de Alta. 2. Que el participante se encuentre en la vista de Objetivos.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Aparece información sobre el ejercicio realizado por el participante ese día.
Flujo Principal	1. El participante pulsa en Ejercicio 2. El sistema muestra la información actualizada sobre el ejercicio diario realizado por el participante.

Figura 6.110: Caso de uso OB04

5. Acceder al objetivo de descanso recomendado siendo participante

Caso de uso	Acceder al objetivo diario de descanso recomendado.
Identificador	CU_OB_05
Objetivo en contexto	Acceder a la zona del objetivo diario de descanso recomendado para llevar un seguimiento.
Actores	Participante
Entidades que usa	Objetivo, ObjetivoDescanso, Participante
Precondiciones	1. Que el participante esté en estado de Alta. 2. Que el participante se encuentre en la vista de Objetivos.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Aparece información sobre las horas de sueño descansadas ese día por el participante.
Flujo Principal	1. El participante pulsa en Descanso 2. El sistema muestra la información actualizada sobre las horas de descanso diario del participante.

Figura 6.111: Caso de uso OB05

6. Acceder al objetivo de estado de ánimo siendo participante

Caso de uso	Acceder al objetivo diario de estado de ánimo.
Identificador	CU_OB_06
Objetivo en contexto	Acceder a la zona del objetivo diario de estado de ánimo para llevar un seguimiento.
Actores	Participante
Entidades que usa	Objetivo, ObjetivoEstadoAnimo, Participante
Precondiciones	1. Que el participante esté en estado de Alta. 2. Que el participante se encuentre en la vista de Objetivos.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Aparece información sobre el estado de ánimo del participante ese día.
Flujo Principal	1. El participante pulsa en Test Anímico 2. El sistema muestra la información actualizada sobre el estado de ánimo del participante.

Figura 6.112: Caso de uso OB06

7. Crear un objetivo personalizado siendo participante

Caso de uso	Crear nuevo objetivo personalizado
Identificador	CU_OB_07
Objetivo en contexto	Añadir un nuevo objetivo personalizado a la base de datos.
Actores	Participante
Entidades que usa	Objetivo, Participante
Precondiciones	1. Que el participante esté en estado de Alta. 2. Que el participante se encuentre en la vista de Objetivos.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Se crea el objetivo correctamente y se visualiza en el calendario y tabla asociados.
Flujo Principal	1. El participante pulsa en Crear un objetivo nuevo 2. El sistema muestra el formulario 3. El participante rellena el formulario para crear el objetivo. 4. El sistema comprueba que los datos introducidos son correctos. 5. El sistema añade el objetivo a la base de datos. 6. El sistema muestra el calendario y tabla de objetivos personalizados actualizados.
Flujo Secundario	3. a) El participante pulsa en Añadir 4. a) El sistema detecta un error en los datos introducidos. 4. b) El sistema pide al participante que cumplimente bien los datos.

Figura 6.113: Caso de uso OB07

8. Editar objetivo personalizado siendo participante

Caso de uso	Editar un objetivo personalizado
Identificador	CU_OB_08
Objetivo en contexto	Editar un objetivo personalizado de la lista de objetivos.
Actores	Participante
Entidades que usa	Objetivo, Participante
Precondiciones	1. Que el participante esté en estado de Alta. 2. Que el participante se encuentre en la vista de Objetivos.
Postcondiciones	Fallo: Se muestran los errores en el formulario. Éxito: Se modifica el objetivo correctamente y se redirige a la vista de objetivos actualizada.
Flujo Principal	1. El participante pulsa en el icono de editar objetivo 2. El sistema redirige al formulario 3. El participante rellena el formulario para editar el objetivo. 4. El participante pulsa Editar. 5. El sistema comprueba que los datos introducidos son correctos. 6. El sistema actualiza el objetivo en la base de datos. 7. El sistema muestra el calendario y tabla de objetivos personalizados actualizados.
Flujo Secundario	6. a) El sistema detecta un error en los datos introducidos. 6. b) El sistema pide que se rellenen bien los datos.

Figura 6.114: Caso de uso OB08

9. Eliminar objetivo personalizado siendo participante

Caso de uso	Eliminar objetivo personalizado
Identificador	CU_OB_09
Objetivo en contexto	Eliminar el objetivo seleccionado
Actores	Participante
Entidades que usa	Objetivo, Participante
Precondiciones	1. Que el participante esté en estado de Alta. 2. Que el participante se encuentre en la vista de Objetivos.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Se elimina el objetivo correctamente.
Flujo Principal	1. El participante pulsa en el ícono de Eliminar de un objetivo concreto. 2. Se elimina el objetivo. 3. El sistema muestra el calendario y tabla de objetivos personalizados actualizados.

Figura 6.115: Caso de uso OB09

10. Filtrar objetivos personalizados siendo participante

Caso de uso	Filtrar objetivos personalizados
Identificador	CU_OB_10
Objetivo en contexto	Filtrar para mostrar solo los objetivos indicados
Actores	Participante
Entidades que usa	Objetivo, Participante
Precondiciones	1. Que el participante esté en estado de Alta. 2. Que el participante se encuentre en la vista de Objetivos.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Se muestra la vista de objetivos actualizada.
Flujo Principal	1. El participante selecciona si quiere mostrar todos los objetivos o solo los pendientes o completados. 2. Se muestra la vista actualizada con los resultados devueltos por el filtro.

Figura 6.116: Caso de uso OB10

11. Ver los objetivos de un día siendo participante

Caso de uso	Ver objetivos de dia concreto
Identificador	CU_OB_11
Objetivo en contexto	Seleccionar un día concreto para visualizar los objetivos establecidos en esa fecha.
Actores	Participante
Entidades que usa	Objetivo, Participante
Precondiciones	1. Que el participante esté en estado de Alta. 2. Que el participante se encuentre en la vista de Objetivos.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Se muestra la vista de objetivos actualizada.
Flujo Principal	1. El participante navega por el calendario hasta encontrar el día del que dese ver los objetivos. 2. El participante pulsa sobre el día escogido 3. El sistema muestra los objetivos correspondientes a la derecha del calendario.

Figura 6.117: Caso de uso OB11

12. Añadir un vaso de agua consumido al objetivo diario de agua siendo participante

Caso de uso	Añadir un vaso de agua consumido
Identificador	CU_OB_12
Objetivo en contexto	Añadir un vaso más al total de vasos de agua diarios consumidos.
Actores	Participante
Entidades que usa	ObjetivoAgua, Participante
Precondiciones	1. Que el participante esté en estado de Alta. 2. Que el participante se encuentre en la vista de Objetivos en la zona del objetivo de agua .
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Se muestra la vista de objetivos actualizada.
Flujo Principal	1. El participante pulsa en Añadir un vaso 2. El sistema muestra los iconos de los vasos y la información del consumo de agua actualizados.

Figura 6.118: Caso de uso OB12

13. Eliminar un vaso de agua consumido del objetivo diario de agua siendo participante

Caso de uso	Eliminar vaso de agua consumido
Identificador	CU_OB_13
Objetivo en contexto	Eliminar un vaso de agua del total de vasos de agua diarios consumidos.
Actores	Participante
Entidades que usa	ObjetivoAgua, Participante
Precondiciones	<ol style="list-style-type: none"> 1. Que el participante esté en estado de Alta. 2. Que el participante se encuentre en la vista de Objetivos en la zona del objetivo de agua .
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Se muestra la vista de objetivos actualizada.
Flujo Principal	<ol style="list-style-type: none"> 1. El participante pulsa en Eliminar un vaso 2. El sistema muestra los iconos de los vasos y la información del consumo de agua actualizados.

Figura 6.119: Caso de uso OB13

14. Añadir un ejercicio realizado al objetivo diario de ejercicio siendo participante

Caso de uso	Añadir ejercicio realizado
Identificador	CU_OB_14
Objetivo en contexto	Añadir la realización de ejercicio al objetivo diario.
Actores	Participante
Entidades que usa	ObjetivoEjercicio, Participante
Precondiciones	<ol style="list-style-type: none"> 1. Que el participante esté en estado de Alta. 2. Que el participante se encuentre en la vista de Objetivos en la zona del objetivo de ejercicio .
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Se muestra la vista de objetivos actualizada.
Flujo Principal	<ol style="list-style-type: none"> 1. El participante selecciona el tipo de ejercicio que ha realizado. 2. El participante agrega la duración del ejercicio. 3. El participante pulsa en Agregar ejercicio. 4. El sistema comprueba que los datos introducidos son correctos. 5. El sistema añade el ejercicio a la base de datos. 6. El sistema muestra la tabla de ejercicio diario actualizada.
Flujo Secundario	<ol style="list-style-type: none"> 4. a) El sistema detecta un error en los datos introducidos. 4. b) El sistema pide que se rellenen bien los datos.

Figura 6.120: Caso de uso OB14

15. Eliminar un ejercicio realizado del objetivo diario de ejercicio siendo partici-

pante

Caso de uso	Eliminar ejercicio realizado
Identificador	CU_OB_15
Objetivo en contexto	Eliminar un ejercicio de la lista de ejercicio diario realizado.
Actores	Participante
Entidades que usa	ObjetivoEjercicio, Participante
Precondiciones	1. Que el participante esté en estado de Alta. 2. Que el participante se encuentre en la vista de Objetivos en la zona del objetivo de ejercicio.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Se muestra la vista de objetivos actualizada.
Flujo Principal	1. El participante pulsa en el icono de Eliminar de un ejercicio concreto. 2. Se elimina el ejercicio. 3. El sistema muestra la tabla de ejercicio diario actualizada.

Figura 6.121: Caso de uso OB15

16. Registrar el tiempo de sueño en el objetivo diario de descanso siendo participante

Caso de uso	Registrar tiempo de sueño
Identificador	CU_OB_16
Objetivo en contexto	Añadir el tiempo descansado por el participante en un día para realizar un seguimiento.
Actores	Participante
Entidades que usa	ObjetivoDescanso, Participante
Precondiciones	1. Que el participante esté en estado de Alta. 2. Que el participante se encuentre en la vista de Objetivos en la zona del objetivo de descanso .
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Se muestra la vista de objetivos actualizada.
Flujo Principal	1. El participante introduce las horas descansadas. 2. El participante introduce los minutos descansados. 3. El participante pulsa en Registrar. 4. El sistema comprueba que los datos introducidos son correctos. 5. El sistema añade el tiempo descansado a la base de datos. 6. El sistema muestra la vista de objetivos actualizada.
Flujo Secundario	4. a) El sistema detecta un error en los datos introducidos. 4. b) El sistema pide que se rellenen bien los datos.

Figura 6.122: Caso de uso OB16

17. Eliminar el registro de sueño del objetivo diario de descanso siendo participante

Caso de uso	Eliminar tiempo descansado
Identificador	CU_OB_17
Objetivo en contexto	Eliminar el tiempo de descanso en un día por el participante.
Actores	Participante
Entidades que usa	ObjetivoDescanso, Participante
Precondiciones	1. Que el participante esté en estado de Alta. 2. Que el participante se encuentre en la vista de Objetivos en la zona del objetivo de descanso.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Se muestra la vista de objetivos actualizada.
Flujo Principal	1. El participante pulsa en Eliminar registro 2. Se elimina el tiempo descansado. 3. El sistema muestra la vista de objetivos actualizada.

Figura 6.123: Caso de uso OB17

18. Registrar el estado de ánimo en el objetivo de estado de ánimo siendo participante

Caso de uso	Registrar estado de ánimo
Identificador	CU_OB_18
Objetivo en contexto	Registrar el estado de ánimo del participante en un día para realizar un seguimiento.
Actores	Participante
Entidades que usa	ObjetivoEstadoAnimo, Participante
Precondiciones	1. Que el participante esté en estado de Alta. 2. Que el participante se encuentre en la vista de Objetivos en la zona del objetivo de estado de ánimo .
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Se muestra la vista de objetivos actualizada.
Flujo Principal	1. El participante selecciona el ícono correspondiente con su estado de ánimo. 3. El participante pulsa en Guardar. 5. El sistema añade el estado de ánimo a la base de datos. 6. El sistema muestra la vista de objetivos actualizada.

Figura 6.124: Caso de uso OB18

19. Eliminar el registro del estado de ánimo del objetivo de estado de ánimo siendo participante

Caso de uso	Eliminar estado de ánimo
Identificador	CU_OB_19
Objetivo en contexto	Eliminar el estado de ánimo en un día por el participante.
Actores	Participante
Entidades que usa	ObjetivoEstadoAnimo, Participante
Precondiciones	1. Que el participante esté en estado de Alta. 2. Que el participante se encuentre en la vista de Objetivos en la zona de estado de ánimo.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Se muestra la vista de objetivos actualizada.
Flujo Principal	1. El participante pulsa en Eliminar registro 2. Se elimina el estado de ánimo. 3. El sistema muestra la vista de objetivos actualizada.

Figura 6.125: Caso de uso OB19

6.3.8. Cuaderno y sesiones

1. Acceder al cuaderno siendo Participante

Caso de uso	Acceder al cuaderno
Identificador	CU_S_01
Objetivo en contexto	Acceder al cuaderno con las diapositivas.
Actores	Participante
Entidades que usa	Participante
Precondiciones	1. Que el participante esté en estado de Alta. 2. Que el participante haya accedido a la aplicación.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Se muestran las diapositivas.
Flujo Principal	1. El participante pulsa en Cuaderno. 2. El participante pulsa en Información.

Figura 6.126: Caso de uso S01

2. Acceder a las fichas siendo Participante

Caso de uso	Acceder a las fichas
Identificador	CU_S_02
Objetivo en contexto	Acceder a las fichas.
Actores	Participante
Entidades que usa	Participante
Precondiciones	1. Que el participante esté en estado de Alta. 2. Que el participante haya accedido a la aplicación.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Se muestran las diapositivas.
Flujo Principal	1. El participante pulsa en Cuaderno. 2. El participante pulsa en Fichas.

Figura 6.127: Caso de uso S02

3. Rellenar la ficha taller siendo Participante

Caso de uso	Rellenar ficha taller
Identificador	CU_S_05
Objetivo en contexto	Rellenar la ficha taller.
Actores	Participante
Entidades que usa	Participante, Ficha Taller
Precondiciones	1. Que el participante esté en estado de Alta. 2. Que el participante esté en la vista de fichas.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Se muestran las fichas.
Flujo Principal	1. El participante pulsa en Ficha Taller. 2. El participante rellena el formulario. 3. El sistema comprueba que los datos introducidos sean correctos. 4. El sistema actualiza los datos de la ficha Taller.
Flujo Secundario	3. a) El sistema detecta un error en los datos introducidos. 3. b) El sistema pide que cumplimenten bien esos datos.

Figura 6.128: Caso de uso S05

4. Rellenar la ficha objetivo siendo Participante

Caso de uso	Rellenar ficha Objetivo
Identificador	CU_S_06
Objetivo en contexto	Rellenar la ficha Objetivo.
Actores	Participante
Entidades que usa	Participante, Ficha Objetivo
Precondiciones	1. Que el participante esté en estado de Alta. 2. Que el participante esté en la vista de fichas.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Se muestran las fichas.
Flujo Principal	1. El participante pulsa en Ficha Objetivo. 2. El participante rellena el formulario. 3. El sistema comprueba que los datos introducidos sean correctos. 4. El sistema actualiza los datos de la ficha Objetivo.
Flujo Secundario	3. a) El sistema detecta un error en los datos introducidos. 3. b) El sistema pide que cumplimenten bien esos datos.

Figura 6.129: Caso de uso S06

5. Acceder a las fichas de elección siendo Participante

Caso de uso	Acceder a las fichas de elección
Identificador	CU_S_07
Objetivo en contexto	Acceder a las fichas de elección.
Actores	Participante
Entidades que usa	Participante
Precondiciones	1. Que el participante esté en estado de Alta. 2. Que el participante esté en la vista de cuaderno.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Se muestra la ficha de elección.
Flujo Principal	1. El participante pulsa en Ficha Elección. 2. El participante pulsa en el número de ficha elección a la que quiere acceder.

Figura 6.130: Caso de uso S07

6. Rellenar una ficha de elección en concreto siendo Participante

Caso de uso	Rellenar ficha Elección
Identificador	CU_S_08
Objetivo en contexto	Rellenar la ficha Elección.
Actores	Participante
Entidades que usa	Participante, Ficha Elección
Precondiciones	1. Que el participante esté en estado de Alta. 2. Que el participante esté en la vista de fichas Elección.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Se muestran las fichas Elección.
Flujo Principal	1. El participante pulsa la ficha Elección que desea. 2. El participante rellena el formulario. 3. El sistema comprueba que los datos introducidos sean correctos. 4. El sistema actualiza los datos de la ficha Elección.
Flujo Secundario	3. a) El sistema detecta un error en los datos introducidos. 3. b) El sistema pide que cumplimenten bien esos datos.

Figura 6.131: Caso de uso S08

7. Acceder a las sesiones del programa siendo Participante

Caso de uso	Acceder a las sesiones
Identificador	CU_S_03
Objetivo en contexto	Acceder a las sesiones.
Actores	Participante
Entidades que usa	Participante
Precondiciones	1. Que el participante esté en estado de Alta. 2. Que el participante haya accedido a la aplicación.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Se muestran la sesión en concreto.
Flujo Principal	1. El participante pulsa en Sesiones. 2. El participante pulsa el número de sesión a la que quiere acceder.

Figura 6.132: Caso de uso S03

8. Rellenar una sesión en concreto siendo Participante

Caso de uso	Rellenar sesión
Identificador	CU_S_09
Objetivo en contexto	Rellenar la sesión.
Actores	Participante
Entidades que usa	Participante, Sesión
Precondiciones	1. Que el participante esté en estado de Alta. 2. Que el participante esté en la vista de la sesión a llenar.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Se muestran las sesiones.
Flujo Principal	1. El participante rellena el formulario. 2. El sistema comprueba que los datos introducidos sean correctos. 3. El sistema actualiza los datos de la sesión.
Flujo Secundario	2. a) El sistema detecta un error en los datos introducidos. 2. b) El sistema pide que cumplimenten bien esos datos.

Figura 6.133: Caso de uso S09

9. Terminar una sesión en concreto siendo Participante

Caso de uso	Terminar sesión
Identificador	CU_S_10
Objetivo en contexto	Terminar la sesión.
Actores	Participante
Entidades que usa	Participante, Sesión
Precondiciones	1. Que el participante esté en estado de Alta. 2. Que el participante esté en la vista de la sesión a terminar.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Se muestran las sesiones.
Flujo Principal	1. El participante pulsa en terminar sesión. 2. El participante rellena el formulario. 3. El sistema comprueba que los datos introducidos sean correctos. 4. El sistema actualiza los datos de la sesión.
Flujo Secundario	3. a) El sistema detecta un error en los datos introducidos. 3. b) El sistema pide que cumplimenten bien esos datos.

Figura 6.134: Caso de uso S10

10. Hacer el cuestionario de una sesión en concreto siendo Participante

Caso de uso	Hacer cuestionario
Identificador	CU_S_12
Objetivo en contexto	Hacer el cuestionario.
Actores	Participante
Entidades que usa	Participante, Sesión
Precondiciones	1. Que el participante esté en estado de Alta. 2. Que el participante esté en la vista de la sesión concreta.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Se muestran las sesiones y se abre el cuestionario en una ventana nueva.
Flujo Principal	1. El participante pulsa en el enlace del cuestionario.

Figura 6.135: Caso de uso S12

11. Descargar las diapositivas de una sesión en concreto siendo Participante

Caso de uso	Descargar diapositivas
Identificador	CU_S_11
Objetivo en contexto	Descargar las diapositivas.
Actores	Participante
Entidades que usa	Participante, Sesión
Precondiciones	1. Que el participante esté en estado de Alta. 2. Que el participante esté en la vista de la sesión concreta.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Se muestran las sesiones y se descarga un pdf con las diapositivas.
Flujo Principal	1. El participante pulsa en el enlace de las diapositivas.

Figura 6.136: Caso de uso S11

12. Acceder a las sesiones de un participante en concreto siendo Coordinador

Caso de uso	Acceder a sesiones
Identificador	CU_S_04
Objetivo en contexto	Acceder a las sesiones de un participante.
Actores	Participante, Coordinador
Entidades que usa	Participante, Sesión, Coordinador
Precondiciones	1. Que el participante esté en estado de Alta. 2. Que el coordinador esté en la vista del expediente del participante.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Se muestran la sesión concreta a la que quiere acceder.
Flujo Principal	1. El coordinador pulsa en Fichas. 2. Selecciona la sesión a la que quiere acceder.

Figura 6.137: Caso de uso S04

13. Ver los datos rellenados por el participante de una sesión en concreto siendo Coordinador

Caso de uso	Ver datos
Identificador	CU_S_13
Objetivo en contexto	Ver los datos rellenados por un participante.
Actores	Participante, Coordinador
Entidades que usa	Participante, Sesión, Coordinador
Precondiciones	1. Que el participante esté en estado de Alta. 2. Que el coordinador esté en la vista de la sesión concreta del participante.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Se muestra la sesión concreta y los datos.
Flujo Principal	1. El coordinador pulsa en Terminar Sesión.

Figura 6.138: Caso de uso S13

6.3.9. Gestión de participantes

1. Recibir notificaciones siendo Participante

Caso de uso	Recibir notificaciones
Identificador	CU_PA_01
Objetivo en contexto	El participante recibe y mira sus notificaciones.
Actores	Participante
Entidades que usa	Participante
Precondiciones	1. Que el participante haya accedido a la aplicación.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Se muestran las notificaciones.
Flujo Principal	1. El participante pulsa en la campana de la cabecera.

Figura 6.139: Caso de uso PA01

2. Rellenar el test de Findrisc antes de registrarse

Caso de uso	Rellenar test de Findrisc
Identificador	CU_PA_02
Objetivo en contexto	Rellenar el test de Findrisc sin registrarte.
Actores	Participante
Entidades que usa	
Precondiciones	1. El usuario, futuro participante, está en la vista de iniciar sesión.
Postcondiciones	Fallo: Se devuelve la vista de iniciar sesión. Éxito: Se devuelve la vista del test actualizada.
Flujo Principal	1. El usuario pulsa en "TEST DE FINDRISC". 2. El usuario rellena el formulario. 3. El sistema comprueba si los datos son correctos. 4. El sistema actualiza la vista con un enlace a las recomendaciones o al registro en función de la puntuación obtenida.
Flujo Secundario	3. a) El sistema detecta un error en los datos introducidos. 3. b) El sistema pide que cumplimenten bien esos datos.

Figura 6.140: Caso de uso PA02

3. Ver las recomendaciones de hábitos saludables

Caso de uso	Ver recomendaciones
Identificador	CU_PA_03
Objetivo en contexto	Ver recomendaciones de hábitos saludables.
Actores	Participante
Entidades que usa	
Precondiciones	1. El usuario, futuro participante, está en la vista de test de Findrisc. 2. Dada su baja puntuación en el test, el sistema le ha mostrado un enlace a Recomendaciones.
Postcondiciones	Fallo: Se devuelve la vista de iniciar sesión. Éxito: Se devuelve la vista de las recomendaciones.
Flujo Principal	1. El usuario pulsa en "Ver recomendaciones".

Figura 6.141: Caso de uso PA03

4. Acceder al listado de participantes de Baja o del coordinador siendo Coordinador

Caso de uso	Acceder al listado
Identificador	CU_PA_04
Objetivo en contexto	Acceder al listado de participantes.
Actores	Participante, Coordinador
Entidades que usa	Coordinador, Participante
Precondiciones	1. El coordinador debe de haber accedido a la página web.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Se devuelve la vista con el listado.
Flujo Principal	1. El coordinador pulsa en Participantes.

Figura 6.142: Caso de uso PA04

5. Filtrar los participantes siendo Coordinador

Caso de uso	Filtrar los participantes
Identificador	CU_PA_05
Objetivo en contexto	Filtrar el listado de participantes.
Actores	Participante, Coordinador
Entidades que usa	Coordinador, Participante, Usuario
Precondiciones	1. El coordinador debe estar en la vista de participantes.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Se devuelve la vista con el listado actualizado.
Flujo Principal	1. El coordinador escribe un texto de búsqueda. 2. Se muestra una lista con los resultados devueltos por el filtro.

Figura 6.143: Caso de uso PA05

6. Imprimir el listado de participante siendo Coordinador

Caso de uso	Imprimir el listado
Identificador	CU_PA_06
Objetivo en contexto	Imprimir el listado de participantes.
Actores	Participante, Coordinador
Entidades que usa	Coordinador, Participante
Precondiciones	1. El coordinador debe estar en la vista de participantes.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Se devuelve la vista con el listado y una interfaz para imprimir.
Flujo Principal	1. El coordinador pulsa en Imprimir.

Figura 6.144: Caso de uso PA06

7. Acceder al expediente de un participante en concreto siendo Coordinador

Caso de uso	Acceder al expediente
Identificador	CU_PA_07
Objetivo en contexto	Acceder al expedientes de un participante en concreto.
Actores	Participante, Coordinador
Entidades que usa	Coordinador, Participante
Precondiciones	1. El coordinador debe estar en la vista de participantes.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Se devuelve la vista con el expediente.
Flujo Principal	1. El coordinador pulsa en el icono de expediente de un participante.

Figura 6.145: Caso de uso PA07

8. Mandar una analítica a un participante en concreto siendo Coordinador

Caso de uso	Mandar analítica
Identificador	CU_PA_08
Objetivo en contexto	Mandar analítica a un participante en concreto.
Actores	Participante, Coordinador
Entidades que usa	Coordinador, Participante
Precondiciones	1. El coordinador debe estar en la vista de expediente de un participante.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Se devuelve la vista con el expediente.
Flujo Principal	1. El coordinador pulsa en Mandar Analítica.

Figura 6.146: Caso de uso PA08

9. Acceder a la fase de valoración de un participante siendo Coordinador

Caso de uso	Acceder a fase de valoración
Identificador	CU_PA_09
Objetivo en contexto	Acceder a la fase de valoración de un participante en concreto.
Actores	Participante, Coordinador
Entidades que usa	Coordinador, Participante
Precondiciones	1. El coordinador debe estar en la vista de expediente de un participante.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Se devuelve la vista con la fase de valoración.
Flujo Principal	1. El coordinador pulsa en Fase de Valoración.

Figura 6.147: Caso de uso PA09

10. Dar de alta a un participante siendo Coordinador

Caso de uso	Dar de alta
Identificador	CU_PA_11
Objetivo en contexto	Dar de alta a un participante en concreto.
Actores	Participante, Coordinador
Entidades que usa	Coordinador, Participante, Usuario
Precondiciones	1. El coordinador debe estar en la vista de fase de valoración.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Se devuelve la vista con la fase de valoración.
Flujo Principal	1. El coordinador pulsa en Activar Cuenta. 2. El sistema comprueba que el participante cumpla los requisitos para darse de ALTA. 3. El sistema da de ALTA al participante.
Flujo Secundario	2. a) El sistema detecta un error. 2. b) El sistema pide que cumplimenten bien los requisitos para dar de ALTA la cuenta.

Figura 6.148: Caso de uso PA11

11. Eliminar la cuenta de un participante siendo Coordinador

Caso de uso	Eliminar la cuenta
Identificador	CU_PA_10
Objetivo en contexto	Eliminar la cuenta de un participante en concreto.
Actores	Participante, Coordinador
Entidades que usa	Coordinador, Participante, Usuario
Precondiciones	1. El coordinador debe estar en la vista de fase valoración de un participante.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Se devuelve la vista con la fase de valoración.
Flujo Principal	1. El coordinador pulsa en Eliminar Cuenta.

Figura 6.149: Caso de uso PA10

12. Imprimir la fase de valoración de un participante siendo Coordinador

Caso de uso	Imprimir fase de valoración
Identificador	CU_PA_12
Objetivo en contexto	Imprimir la fase de valoración.
Actores	Coordinador, Participante
Entidades que usa	Participante, Coordinador, Fase de Valoración
Precondiciones	1. Que el coordinador se encuentre en la vista de la fase de valoración de un participante.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Se muestran interfaz para imprimir la página.
Flujo Principal	1. El coordinador pulsa en imprimir.

Figura 6.150: Caso de uso PA12

13. Rellenar formulario de Exploración de un participante siendo Coordinador

Caso de uso	Rellenar Exploración
Identificador	CU_PA_13
Objetivo en contexto	Rellenar el formulario de Exploración.
Actores	Coordinador, Participante
Entidades que usa	Participante, Coordinador, Exploración
Precondiciones	1. Que el coordinador se encuentre en la vista del formulario de Exploración de un participante.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Se devuelve la vista de la fase de valoración.
Flujo Principal	1. El coordinador rellena el formulario. 2. El sistema comprueba que los datos introducidos sean correctos. 3. El sistema actualiza los datos del formulario de Exploración.
Flujo Secundario	2. a) El sistema detecta un error en los datos introducidos. 2. b) El sistema pide que cumplimenten bien esos datos.

Figura 6.151: Caso de uso PA13

14. Imprimir formulario de Exploración de un participante siendo Coordinador

Caso de uso	Imprimir exploración
Identificador	CU_PA_19
Objetivo en contexto	Imprimir el formulario de exploración.
Actores	Coordinador, Participante
Entidades que usa	Participante, Coordinador, Exploración
Precondiciones	1. Que el coordinador se encuentre en la vista del formulario de exploración de un participante.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Se muestran interfaz para imprimir la página.
Flujo Principal	1. El coordinador pulsa en imprimir.

Figura 6.152: Caso de uso PA19

15. Rellenar formulario de Findrisc de un participante siendo Coordinador

Caso de uso	Rellenar Findrisc
Identificador	CU_PA_14
Objetivo en contexto	Rellenar el formulario de Findrisc.
Actores	Coordinador, Participante
Entidades que usa	Participante, Coordinador, Findrisc
Precondiciones	1. Que el coordinador se encuentre en la vista del formulario de Findrisc de un participante.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Se devuelve la vista de la fase de valoración.
Flujo Principal	1. El coordinador rellena el formulario. 2. El sistema comprueba que los datos introducidos sean correctos. 3. El sistema actualiza los datos del formulario de Findrisc.
Flujo Secundario	2. a) El sistema detecta un error en los datos introducidos. 2. b) El sistema pide que cumplimenten bien esos datos.

Figura 6.153: Caso de uso PA14

16. Imprimir formulario de Findrisc de un participante siendo Coordinador

Caso de uso	Imprimir findrisc
Identificador	CU_PA_20
Objetivo en contexto	Imprimir el formulario de findrisc.
Actores	Coordinador, Participante
Entidades que usa	Participante, Coordinador, Findrisc
Precondiciones	1. Que el coordinador se encuentre en la vista del formulario de findrisc de un participante.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Se muestran interfaz para imprimir la página.
Flujo Principal	1. El coordinador pulsa en imprimir.

Figura 6.154: Caso de uso PA20

17. Rellenar formulario de Antecedentes de un participante siendo Coordinador

Caso de uso	Rellenar Antecedentes
Identificador	CU_PA_15
Objetivo en contexto	Rellenar el formulario de Antecedentes.
Actores	Coordinador, Participante
Entidades que usa	Participante, Coordinador, Antecedentes
Precondiciones	1. Que el coordinador se encuentre en la vista del formulario de Antecedentes de un participante.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Se devuelve la vista de la fase de valoración.
Flujo Principal	1. El coordinador rellena el formulario. 2. El sistema comprueba que los datos introducidos sean correctos. 3. El sistema actualiza los datos del formulario de Antecedentes.
Flujo Secundario	2. a) El sistema detecta un error en los datos introducidos. 2. b) El sistema pide que cumplimenten bien esos datos.

Figura 6.155: Caso de uso PA15

18. Imprimir formulario de Antecedentes de un participante siendo Coordinador

Caso de uso	Imprimir Antecedentes
Identificador	CU_PA_21
Objetivo en contexto	Imprimir el formulario de Antecedentes.
Actores	Coordinador, Participante
Entidades que usa	Participante, Coordinador, Antecedentes
Precondiciones	1. Que el coordinador se encuentre en la vista del formulario de Antecedentes de un participante.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Se muestran interfaz para imprimir la página.
Flujo Principal	1. El coordinador pulsa en imprimir.

Figura 6.156: Caso de uso PA21

19. Rellenar formulario de Adherencia a la dieta mediterránea de un participante siendo Coordinador

Caso de uso	Rellenar Adherencia
Identificador	CU_PA_16
Objetivo en contexto	Rellenar el formulario de Adherencia.
Actores	Coordinador, Participante
Entidades que usa	Participante, Coordinador, Adherencia
Precondiciones	1. Que el coordinador se encuentre en la vista del formulario de Adherencia de un participante.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Se devuelve la vista de la fase de valoración.
Flujo Principal	1. El coordinador rellena el formulario. 2. El sistema comprueba que los datos introducidos sean correctos. 3. El sistema actualiza los datos del formulario de Adherencia.
Flujo Secundario	2. a) El sistema detecta un error en los datos introducidos. 2. b) El sistema pide que cumplimenten bien esos datos.

Figura 6.157: Caso de uso PA16

20. Imprimir formulario de Adherencia a la dieta mediterránea de un participante siendo Coordinador

Caso de uso	Imprimir Adherencia
Identificador	CU_PA_22
Objetivo en contexto	Imprimir el formulario de Adherencia.
Actores	Coordinador, Participante
Entidades que usa	Participante, Coordinador, AlimentacionVal
Precondiciones	1. Que el coordinador se encuentre en la vista del formulario de Adherencia de un participante.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Se muestran interfaz para imprimir la página.
Flujo Principal	1. El coordinador pulsa en imprimir.

Figura 6.158: Caso de uso PA22

21. Rellenar formulario de Actividad Física de un participante siendo Coordinador

Caso de uso	Rellenar Actividad Física
Identificador	CU_PA_17
Objetivo en contexto	Rellenar el formulario de Actividad Física.
Actores	Coordinador, Participante
Entidades que usa	Participante, Coordinador, Actividad Física
Precondiciones	1. Que el coordinador se encuentre en la vista del formulario de Actividad Física de un participante.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Se devuelve la vista de la fase de valoración.
Flujo Principal	1. El coordinador rellena el formulario. 2. El sistema comprueba que los datos introducidos sean correctos. 3. El sistema actualiza los datos del formulario de Actividad Física.
Flujo Secundario	2. a) El sistema detecta un error en los datos introducidos. 2. b) El sistema pide que cumplimenten bien esos datos.

Figura 6.159: Caso de uso PA17

22. Imprimir formulario de Actividad Física de un participante siendo Coordinador

Caso de uso	Imprimir Actividad Física
Identificador	CU_PA_23
Objetivo en contexto	Imprimir el formulario de Actividad Física.
Actores	Coordinador, Participante
Entidades que usa	Participante, Coordinador, ActFisica
Precondiciones	1. Que el coordinador se encuentre en la vista del formulario de Actividad Física de un participante.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Se muestran interfaz para imprimir la página.
Flujo Principal	1. El coordinador pulsa en imprimir.

Figura 6.160: Caso de uso PA23

23. Rellenar formulario de Clasificación de un participante siendo Coordinador

Caso de uso	Rellenar Clasificación
Identificador	CU_PA_18
Objetivo en contexto	Rellenar el formulario de Clasificación.
Actores	Coordinador, Participante
Entidades que usa	Participante, Coordinador, Clasificación
Precondiciones	1. Que el coordinador se encuentre en la vista del formulario de Clasificación de un participante.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Se devuelve la vista de la fase de valoración.
Flujo Principal	1. El coordinador rellena el formulario. 2. El sistema comprueba que los datos introducidos sean correctos. 3. El sistema actualiza los datos del formulario de Clasificación.
Flujo Secundario	2. a) El sistema detecta un error en los datos introducidos. 2. b) El sistema pide que cumplimenten bien esos datos.

Figura 6.161: Caso de uso PA18

24. Imprimir formulario de Clasificación de un participante siendo Coordinador

Caso de uso	Imprimir Clasificación
Identificador	CU_PA_24
Objetivo en contexto	Imprimir el formulario de Clasificación.
Actores	Coordinador, Participante
Entidades que usa	Participante, Coordinador, Clasificación
Precondiciones	1. Que el coordinador se encuentre en la vista del formulario de Clasificación de un participante.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Se muestran interfaz para imprimir la página.
Flujo Principal	1. El coordinador pulsa en imprimir.

Figura 6.162: Caso de uso PA24

6.3.10. Gestión de grupos

1. Acceder a mi grupo siendo participante

Caso de uso	Acceder a mi grupo
Identificador	CU_GR_01
Objetivo en contexto	Acceder al grupo del participante
Actores	Participante
Entidades que usa	Participante, Grupo, Mensaje
Precondiciones	1. Que el participante esté en estado de Alta. 2. Que el participante haya accedido a la aplicación.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Aparece la vista con la información del grupo actualizada
Flujo Principal	1. El participante pulsa en Mi grupo. 2. El sistema muestra la información actualizada

Figura 6.163: Caso de uso GR01

2. Acceder al listado de grupos siendo coordinador

Caso de uso	Acceder al listado de grupos
Identificador	CU_GR_02
Objetivo en contexto	Acceder al listado de grupos gestionados por el coordinador.
Actores	Coordinador, Grupo
Entidades que usa	Coordinador, Grupo
Precondiciones	1. El coordinador debe de haber accedido a la aplicación.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Se devuelve la vista con el listado de grupos
Flujo Principal	1. El coordinador pulsa en Grupos.

Figura 6.164: Caso de uso GR02

3. Añadir participante a un grupo siendo coordinador

Caso de uso	Añadir participante a grupo
Identificador	CU_GR_03
Objetivo en contexto	Añadir un participante a un grupo existente actualizando la base de datos.
Actores	Coordinador, Participante
Entidades que usa	Coordinador, Participante, Grupo
Precondiciones	1. Que el coordinador esté en estado de Alta. 2. Que el coordinador se encuentre en la vista de listado de participantes.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Se actualiza la base de datos con el participante en el grupo correspondiente.
Flujo Principal	1. El coordinador pulsa el icono de Añadir a un grupo de un participante concreto. 2. El sistema muestra un listado con los grupos en los que puede añadirse al participante. 3. El coordinador selecciona el grupo al que quiere añadir al participante. 4. El sistema redirige a la vista del listado de grupos actualizada.
Flujo Secundario	1. a) El sistema detecta que el participante ya pertenece a otro grupo y lo muestra en un modal.

Figura 6.165: Caso de uso GR03

4. Crear un nuevo grupo siendo coordinador

Caso de uso	Crear nuevo grupo
Identificador	CU_GR_04
Objetivo en contexto	Añadir un nuevo grupo a la base de datos.
Actores	Coordinador
Entidades que usa	Grupo, Coordinador
Precondiciones	1. Que el coordinador esté en estado de Alta. 2. Que el coordinador se encuentre en la vista del listado de grupos.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Se crea el grupo correctamente
Flujo Principal	1. El coordinador pulsa en Crear un grupo nuevo 2. El sistema muestra el formulario en un modal 3. El coordinador rellena el formulario para crear el grupo. 4. El sistema comprueba que los datos introducidos son correctos. 5. El sistema añade el grupo a la base de datos. 6. El sistema muestra el listado de grupos actualizado.
Flujo Secundario	3. a) El coordinador pulsa en Añadir 4. a) El sistema detecta un error en los datos introducidos. 4. b) El sistema pide al participante que cumplimente bien los datos.

Figura 6.166: Caso de uso GR04

5. Ver un grupo concreto siendo coordinador

Caso de uso	Ver un grupo
Identificador	CU_GR_05
Objetivo en contexto	Visualizar toda la información de un grupo concreto dirigido por el coordinador.
Actores	Coordinador
Entidades que usa	Grupo, Observaciones, Coordinador
Precondiciones	1. Que el coordinador esté en estado de Alta. 2. Que el coordinador se encuentre en la vista del listado de grupos.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Se devuelve la vista con la información de grupo.
Flujo Principal	1. El coordinador pulsa en el ícono de ver un grupo de un grupo concreto de la lista. 2. El sistema muestra la información de ese grupo

Figura 6.167: Caso de uso GR05

6. Crear una nota nueva de observación de un grupo siendo coordinador

Caso de uso	Crear nueva observación
Identificador	CU_GR_06
Objetivo en contexto	Añadir una nueva observación de un grupo a la base de datos.
Actores	Coordinador
Entidades que usa	Observaciones, Grupo, Coordinador
Precondiciones	1. Que el coordinador esté en estado de Alta. 2. Que el coordinador se encuentre en la vista de un grupo concreto.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Se crea la observación correctamente
Flujo Principal	1. El coordinador pulsa en Crear una nota nueva 2. El sistema muestra el formulario en un modal 3. El coordinador rellena el formulario para crear la nota. 4. El sistema comprueba que los datos introducidos son correctos. 5. El sistema añade la observación a la base de datos. 6. El sistema muestra la vista del grupo actualizada.
Flujo Secundario	3. a) El coordinador pulsa en Añadir 4. a) El sistema detecta un error en los datos introducidos. 4. b) El sistema pide al participante que cumplimente bien los datos.

Figura 6.168: Caso de uso GR06

7. Editar una nota concreta de observación de un grupo siendo coordinador

Caso de uso	Editar una observación
Identificador	CU_GR_06
Objetivo en contexto	Editar una observación de un grupo actualizando la base de datos.
Actores	Coordinador
Entidades que usa	Observaciones, Grupo, Coordinador
Precondiciones	1. Que el coordinador esté en estado de Alta. 2. Que el coordinador se encuentre en la vista de un grupo concreto en la zona de notas y observaciones.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Se modifica la observación correctamente
Flujo Principal	1. El coordinador pulsa en el ícono de editar observación 2. El sistema muestra el formulario en un modal 3. El coordinador rellena el formulario para editar la nota. 4. El sistema comprueba que los datos introducidos son correctos. 5. El sistema modifica la observación de la base de datos. 6. El sistema muestra la vista del grupo actualizada.
Flujo Secundario	3. a) El coordinador pulsa en Añadir 4. a) El sistema detecta un error en los datos introducidos. 4. b) El sistema pide al participante que cumplimente bien los datos.

Figura 6.169: Caso de uso GR07

8. Eliminar una nota concreta de observación de un grupo siendo coordinador

Caso de uso	Eliminar observación
Identificador	CU_GR_08
Objetivo en contexto	Eliminar la observación seleccionada
Actores	Coordinador
Entidades que usa	Observaciones, Grupo, Coordinador
Precondiciones	1. Que el coordinador esté en estado de Alta. 2. Que el coordinador se encuentre en la vista de un grupo concreto en la zona de notas y observaciones.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Se elimina la observación correctamente.
Flujo Principal	1. El coordinador pulsa en el ícono de Eliminar de una nota u observación concreta. 2. El sistema muestra un modal de confirmación. 3. Se elimina la observación. 4. El sistema muestra la zona de notas y observaciones actualizada.
Flujo Secundario	1. a) El coordinador pulsa en eliminar.

Figura 6.170: Caso de uso GR08

9. Editar la información de un grupo concreto siendo coordinador

Caso de uso	Editar un grupo
Identificador	CU_GR_09
Objetivo en contexto	Editar la información de un grupo actualizando la base de datos.
Actores	Coordinador
Entidades que usa	Grupo, Coordinador, Participante
Precondiciones	1. Que el coordinador esté en estado de Alta. 2. Que el coordinador se encuentre en la vista del listado de grupos.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Se modifica el grupo correctamente
Flujo Principal	1. El coordinador pulsa en el ícono de Editar grupo. 2. El sistema muestra el formulario. 3. El coordinador modifica el formulario para editar el grupo. 4. El sistema comprueba que los datos introducidos son correctos. 5. El sistema modifica el grupo de la base de datos. 6. El sistema muestra la vista de listado de grupos actualizada.
Flujo Secundario	3. a) El coordinador pulsa en Editar 4. a) El sistema detecta un error en los datos introducidos. 4. b) El sistema pide al coordinador que cumplimente bien los datos.

Figura 6.171: Caso de uso GR09

10. Eliminar un grupo siendo coordinador

Caso de uso	Eliminar grupo
Identificador	CU_GR_10
Objetivo en contexto	Eliminar el grupo seleccionada
Actores	Coordinador
Entidades que usa	Grupo, Coordinador
Precondiciones	1. Que el coordinador esté en estado de Alta. 2. Que el coordinador se encuentre en la vista del listado de grupos.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Se elimina el grupo correctamente.
Flujo Principal	1. El coordinador pulsa en el icono de Eliminar de un grupo concreto. 2. El sistema muestra un modal de confirmación. 3. Se elimina el grupo. 4. El sistema muestra el listado de grupos actualizado.
Flujo Secundario	1. a) El coordinador pulsa en eliminar.

Figura 6.172: Caso de uso GR10

11. Buscar un grupo concreto en la lista de grupos siendo coordinador

Caso de uso	Buscar un grupo
Identificador	CU_GR_11
Objetivo en contexto	Buscar un grupo concreto
Actores	Coordinador
Entidades que usa	Grupo, Coordinador
Precondiciones	1. Que el coordinador esté en estado de Alta. 2. Que el coordinador se encuentre en la vista del listado de grupos.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Se muestra el grupo buscado.
Flujo Principal	1. El coordinador introduce texto para buscar al elegido. 2. Se muestra la vista actualizada con los resultados devueltos por el buscador.

Figura 6.173: Caso de uso GR11

12. Filtrar grupos activos y terminados en la lista de grupos siendo coordinador

Caso de uso	Filtrar grupos
Identificador	CU_GR_12
Objetivo en contexto	Filtrar para mostrar solo los grupos indicados
Actores	Coordinador
Entidades que usa	Grupo, Coordinador
Precondiciones	1. Que el coordinador esté en estado de Alta. 2. Que el coordinador se encuentre en la vista del listado de grupos.
Postcondiciones	Fallo: Se devuelve la vista principal. Éxito: Se muestra la vista del listado actualizada.
Flujo Principal	1. El coordinador selecciona si quiere mostrar todos los grupos o solo los activos o terminados. 2. Se muestra la vista actualizada con los resultados devueltos por el filtro.

Figura 6.174: Caso de uso GR12

Capítulo 7

Implementación de la aplicación

7.1. Diagramas de actividad

7.1.1. Gestión de usuarios

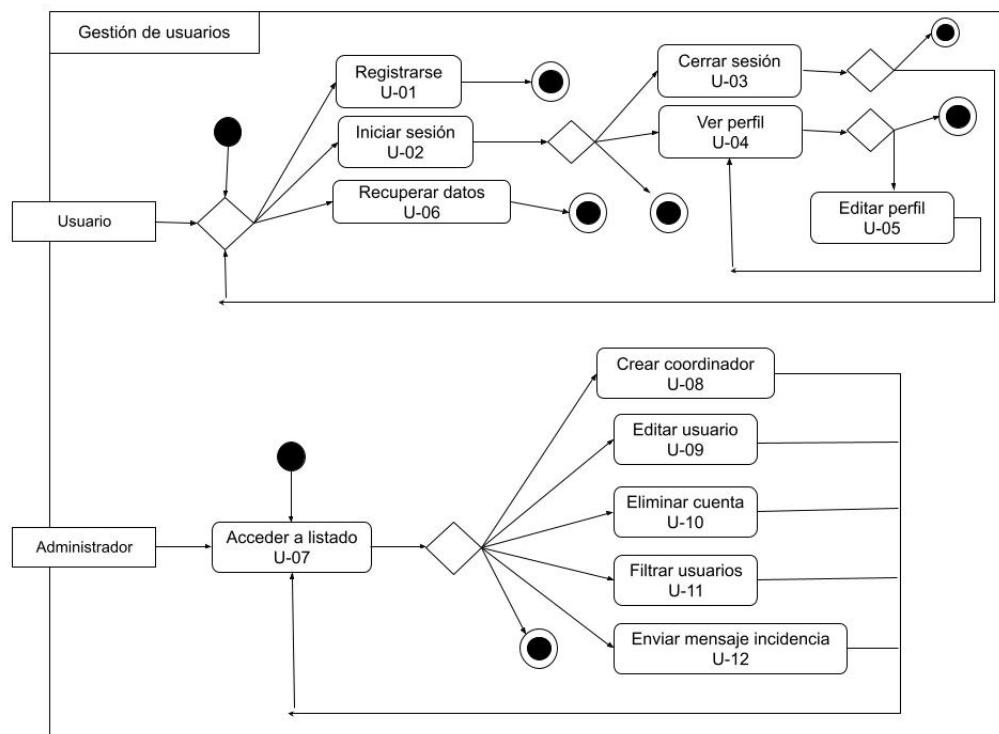


Figura 7.1: Diagrama de actividades de gestión de usuarios

7.1.2. Actividades

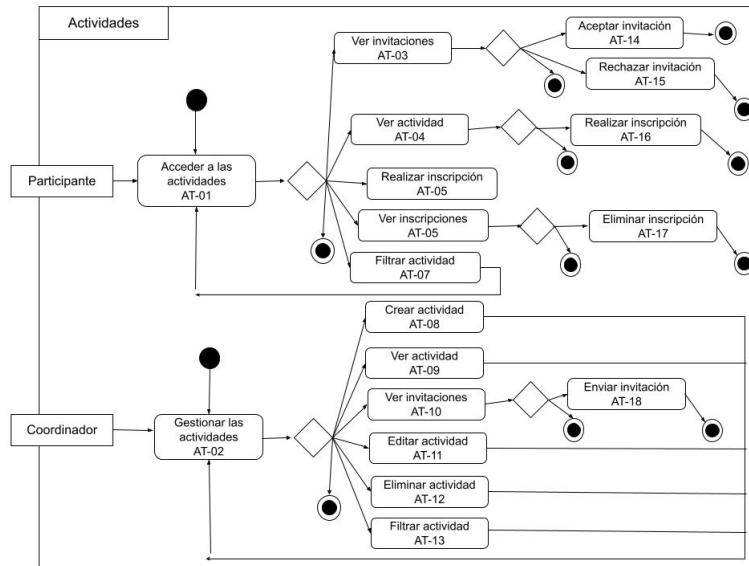


Figura 7.2: Diagrama de actividades de gestión de actividades

7.1.3. Materiales

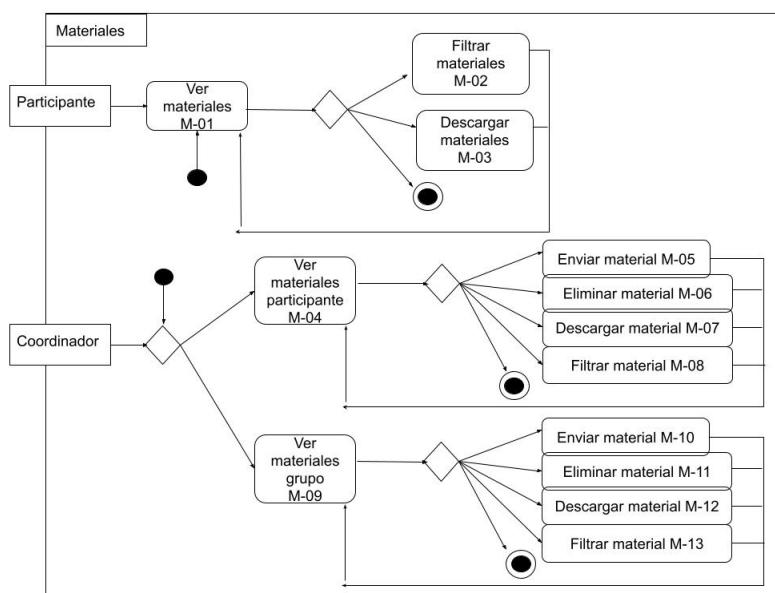


Figura 7.3: Diagrama de actividades de materiales

7.1.4. Progreso

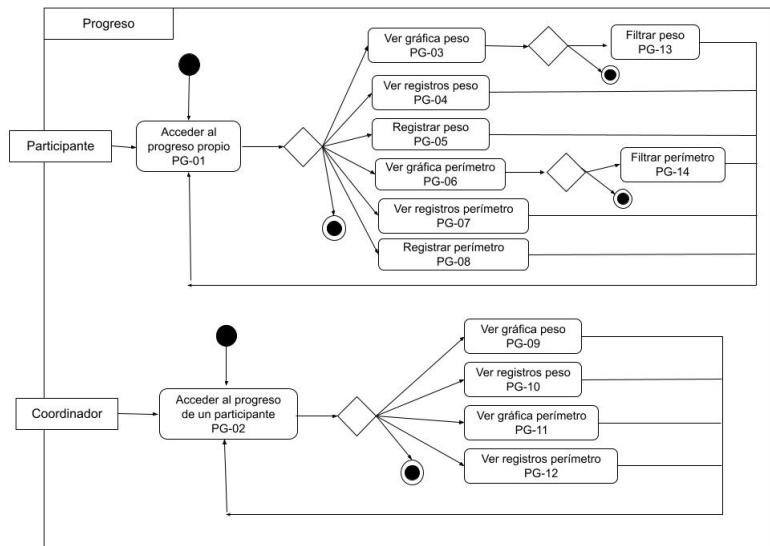


Figura 7.4: Diagrama de actividades de progreso

7.1.5. Chat

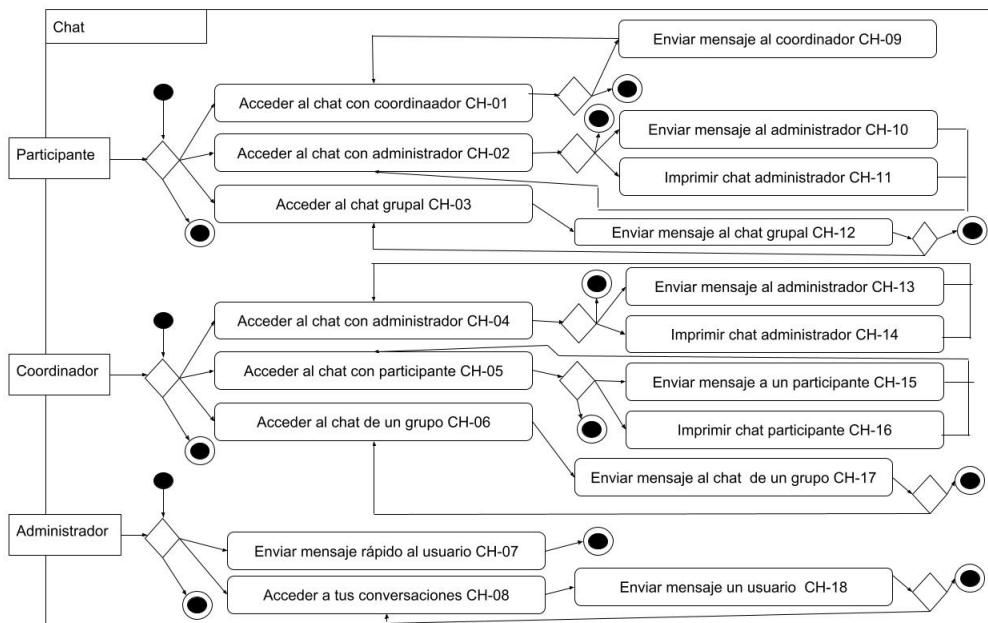


Figura 7.5: Diagrama de actividades de chat

7.1.6. Alimentación

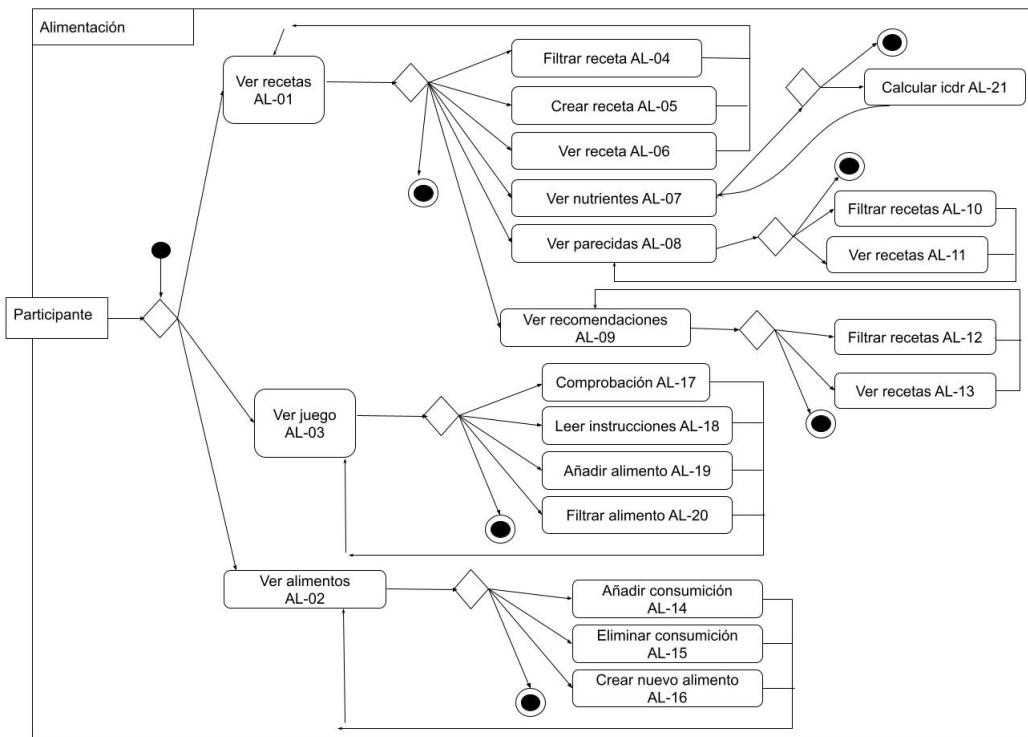


Figura 7.6: Diagrama de actividades de alimentación

7.1.7. Objetivos

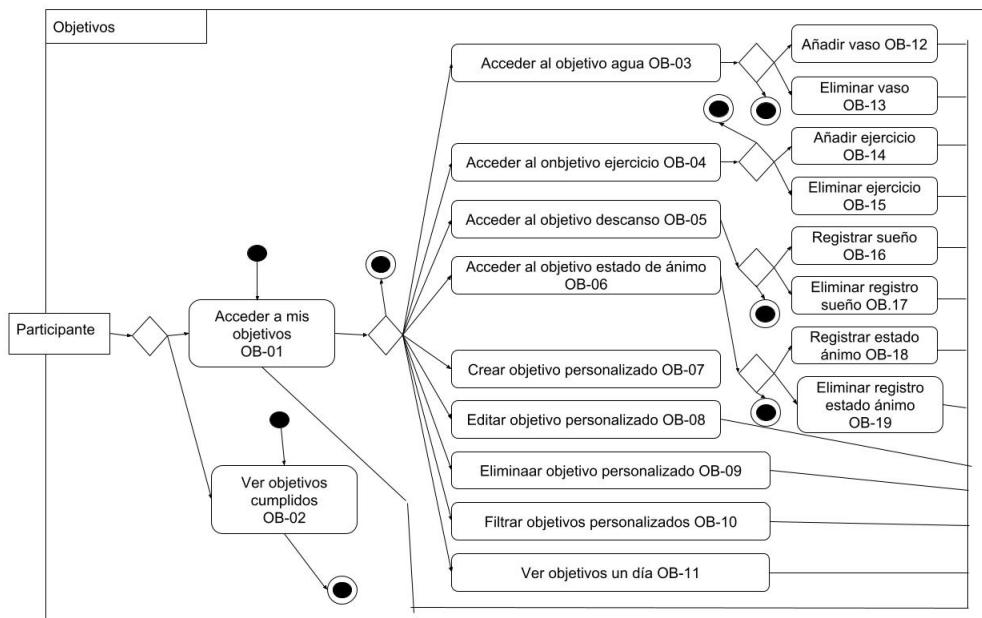


Figura 7.7: Diagrama de actividades de objetivos

7.1.8. Cuaderno y sesiones

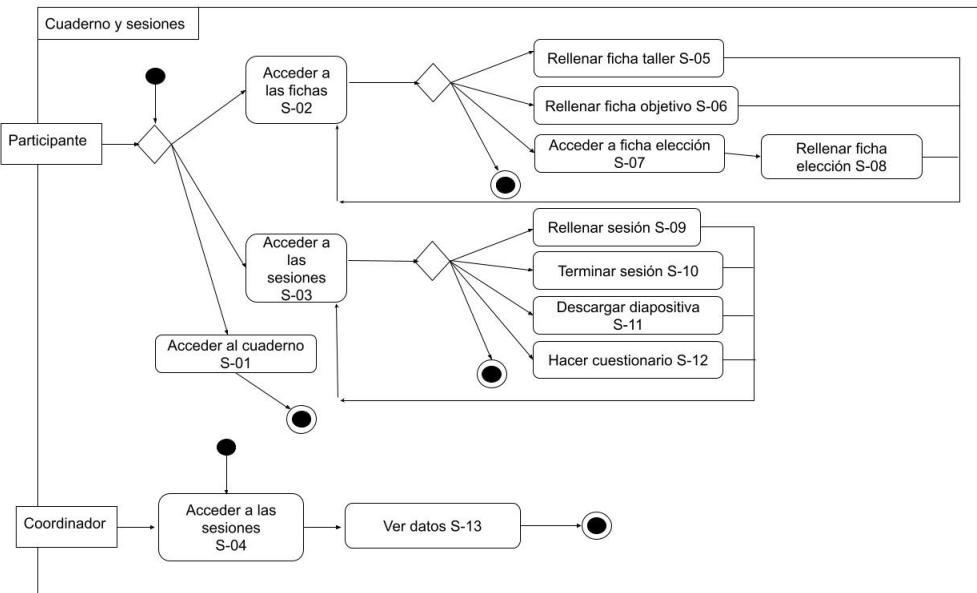


Figura 7.8: Diagrama de actividades de cuaderno y sesiones

7.1.9. Gestión de participantes

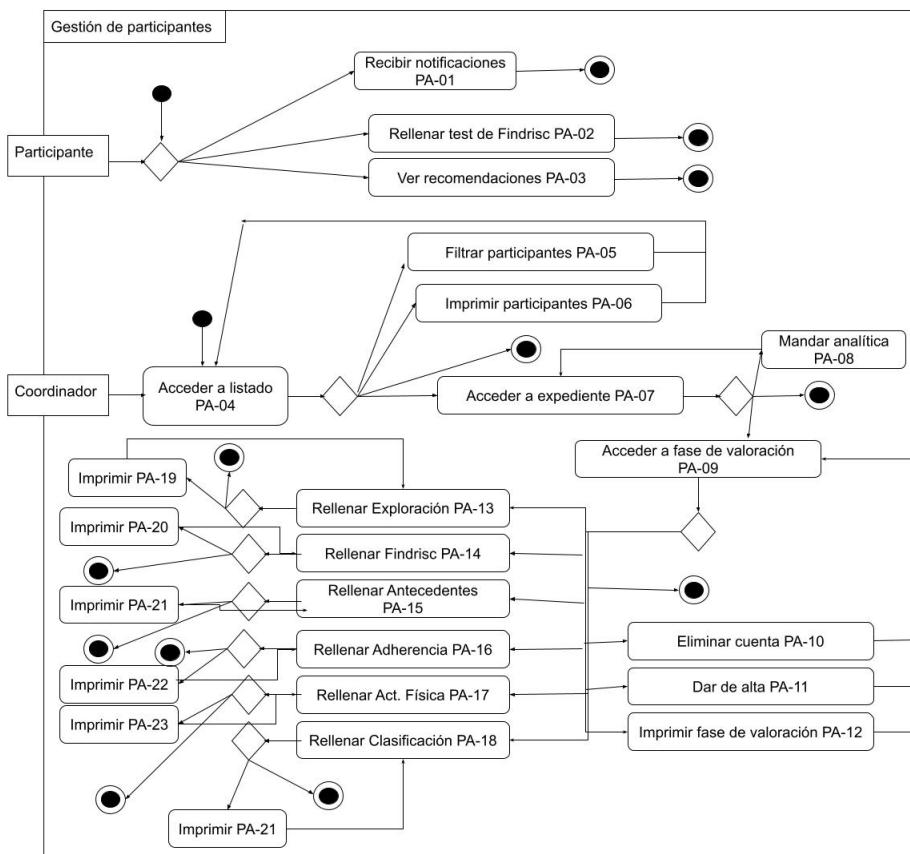


Figura 7.9: Diagrama de actividades de gestión de participantes

7.1.10. Gestión de grupos

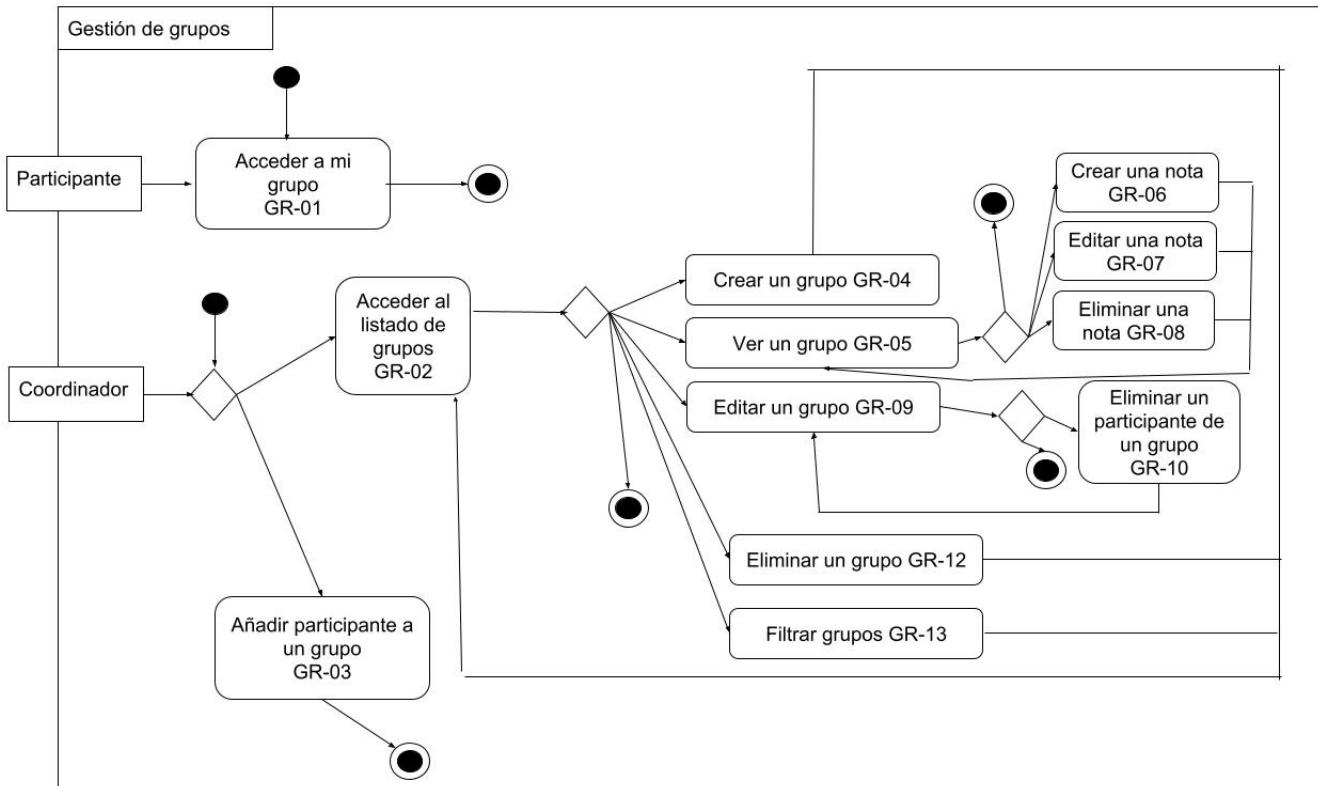


Figura 7.10: Diagrama de actividades de grupos

7.2. Back-End o lado del servidor

7.2.1. Gestión de usuarios

1. Registrarse siendo Usuario

Para realizar el registro en la aplicación, se ha implementado la función `showRegistrationForm()` que devuelve como resultado el script `registro.html`, donde se ha creado un formulario con los campos alias, correo electrónico, contraseña y confirmar contraseña. Al llenar los datos, se realiza una validación de los mismos con el archivo `validaciones.js` para comprobar que tiene la longitud de caracteres adecuada o sea un email válido. Además de llenar los datos, es necesario aceptar los términos y condiciones, que son las cookies de la aplicación. Una cookie es un pequeño archivo que se almacena en el navegador del usuario, y que se envía junto con las solicitudes HTTP a un servidor web determinado.

Al pulsar en el botón de Registrarse se redirige a la función `processRegister()` de `HomeController.java`. Para la contraseña se realiza una instancia del objeto `BCryptPasswordEncoder`, que es una clase que se utiliza para codificar las contraseñas de manera segura. Para el código de identificación

del usuario se genera un número aleatorio entre 0 y 100000 con la función Random(), ya que es el número máximo de usuarios que puede tener la aplicación y se verifica que no existe otro usuario con el mismo código. Además, se inicializan algunos atributos como el estado del participante a BAJA.

Posteriormente se comprueba si las cookies han sido aceptadas. Por lo tanto, si el participante ha dado su consentimiento al realizar el registro, se crea una cookie llamada consentimiento con un valor igual al código de identificación del usuario. La cookie se configura para que expire en un año y se asegura de que solo se envía a través de HTTPS para mejorar la seguridad. Además, se establece la propiedad HttpOnly en la cookie para protegerla contra los ataques de scripting malintencionados.

Por último, se crean los formularios de la fase de valoración y se redirige al script registersuccess.html donde aparece el código generado.

2. Iniciar sesión siendo Usuario

Para llevar a cabo el inicio de sesión en la aplicación, se ha implementado un formulario con los campos código y contraseña. Como el login es una vista que ya la tiene desarrollada Spring Boot, es necesario que en la función configure() de SpringSecurity.java se indique el nombre de la página del login personalizado y donde se procesa, en este caso en la ruta /autenticacion. Esta ruta es la URL a la que se envían los datos de inicio de sesión del usuario cuando hace clic en el botón de enviar. En Spring Security, esta URL está reservada y se utiliza para procesar la autenticación del usuario, que implica validar las credenciales del usuario, en este caso el código y la contraseña, comparándolas con los datos almacenados en la base de datos.

Si los credenciales son correctos se redirige a la función index() de HomeController.java y se comprueba que tipo de usuario es.

En el caso de ser un coordinador o un administrador y tengan el estado en BAJA, la función devolverá el script bajaUsuario.html donde solo tienen acceso a escribir a un administrador para que le den de alta en la aplicación. Por el contrario si tienen el estado en ALTA se redirige a las vistas coordinador.html y admin.html respectivamente.

Por otro lado, si es un participante, primero se comprueba si tiene cookies mirando si entre las cookies existentes hay una con el nombre consentimiento con un valor igual al código único de usuario. Si no tiene esa cookie, se redirige al script de términos y condiciones donde se ha implementado un botón para que el participante acepte el consentimiento de las cookies y pueda iniciar sesión correctamente. Esta cookie tiene las mismas características que la descrita en el caso anterior.

Mientras que si tiene el consentimiento aceptado y está dado de alta, se realiza una consulta para comprobar si el participante ha registrado el peso en la última semana y en caso de que no, mostrarle un recordatorio al iniciar sesión.

Después se hacen varias consultas para obtener las notificaciones, los objetivos, el porcentaje del progreso y las sesiones completadas. El porcentaje del

progreso se calcula dividiendo el peso que ha perdido con el peso que quiere perder.

La función devuelve el script index.html para el usuario participante.

3. Cerrar sesión siendo Usuario

En la parte superior derecha de cada vista de la aplicación, aparece un desplegable con varias opciones en las que se encuentra Cerrar sesión.

Cuando se pulsa sobre este enlace, se realiza una petición POST HTTP a la ruta /logout. Una petición HTTP POST es una solicitud que se hace a un servidor web para enviar datos al servidor y procesarlos o almacenarlos. Spring Security invalidará la sesión iniciada y redirigirá al usuario a la página de inicio de sesión.

Para habilitar el logout, se ha agregado en la función configure() de SecurityConfig.java la configuración. Se especifica que se le permite a todos los usuarios cerrar sesión con http.logout(logout -> logout.permitAll()).

4. Ver perfil siendo Usuario

Como se comenta en el caso anterior, en la parte superior derecha de cada script de la aplicación hay un desplegable con una serie de opciones en la que se encuentra Mi perfil.

Al pulsar en este enlace se redirige a la función mostrarperfil() implementada en HomeController.java. Se comprueba si el perfil al que se quiere acceder es del usuario que ha iniciado sesión y luego se diferencia por tipo de rol.

En el caso de ser un participante se inicializan todos los atributos. Si el participante no ha llenado la edad y el sexo o no pertenece a ningún grupo, se le asignará un texto de No asignado por cada atributo. Despues esos valores se le pasan al model para mostrarlos en la vista mostrarperfilParticipante.html.

Por último se hace una búsqueda de todas las notificaciones del participante y se pasan también por el model a la vista.

Por otro lado, si el usuario es un coordinador o un administrador, se redirige al script mostrarperfil.html.

5. Editar perfil siendo Usuario

En la vista de Ver perfil se diseñó un botón para editar el perfil. Al pulsar sobre él se redirige a la función perfil() de HomeController.java, en la cual se diferencia por el tipo de usuario para devolver como resultado el script editarperfilParticipante.html si es un participante o editarperfil.html si es un coordinador o un administrador.

En la vista de editar el perfil se muestra un formulario con los campos editables. Mientras el usuario edita los campos, son validados en el archivo validaciones.js para comprobar que cumpla con los requisitos definidos. Cuando no existan errores, el usuario al pulsar en el botón de Guardar, se redirige a la función processPerfil() de HomeController.java, donde se actualizan los datos modificados y se guarda en la base de datos.

6. Recuperar datos siendo Usuario

Una vez el usuario se encuentre en la vista recuperarCodigo.html, rellena el formulario y pulsa en el botón de enviar. Este botón le redirige a la función de HomeController llamada procesoRecuperarCodigo() que, con el correo del usuario, se encarga de generar una nueva contraseña aleatoria usando la función generarNuevaContrasenia(), que crea un string, escogiendo ocho posiciones aleatoriamente que seleccionan cada carácter que forma el string. Después, se encripta esta contraseña usando el algoritmo de encriptación de contraseñas BCrypt. BCrypt es un algoritmo de hashing de contraseñas que utiliza una función de hash unidireccional para convertir una contraseña en un valor de hash. Por último, se llama a la función recuperarCodigo() de UsuarioService.

En la función recuperarCodigo() de UsuarioService, se comprueba que el correo electrónico introducido coincida con algún participante de la base de datos. Si no es así el mensaje del correo electrónico es "Ha intentado recuperar sus datos de Estepper sin estar registrado. Acceda a estepper.com para registrarse.". Mientras que si ese correo electrónico existe en la base de datos se le remite en el mensaje los datos con su código y la nueva contraseña generada, y se actualiza el participante en la base de datos con la nueva contraseña. Después se llama a la función de mandarcorreo(), dentro de esta, se crea un servicio de Gmail y un mensaje con MimeMessage, luego se trabaja con el mensaje para pasarlo a la clase Message y se envía usando service. Para ello hemos usado las bibliotecas de javax.mail, com.google.api.client y com.google.api.services.gmail.

7. Acceder al listado de usuarios

La página principal del usuario con rol de Administrador es el script admin.html, donde aparece el listado de todos los usuarios. Por lo tanto, para acceder a admin.html el administrador inicia sesión y en la función index() de HomeController() se redirige al script donde aparece el listado en el caso de que su estado sea ALTA.

8. Crear coordinador siendo Administrador

Este caso de uso se realiza a partir del script admin.html. En la parte superior derecha de esta vista se ha implementado un botón llamado Nuevo coordinador. Al pulsar en él, aparece un modal (ventana emergente) que se abre a través de la función abrirIframe() del archivo admin.js.

Seguidamente aparece un formulario con los campos a llenar por el administrador. Mientras se completan los campos, se realiza una validación de cada uno de ellos en el archivo validaciones.js. Se comprueba que el alias del coordinador tenga mínimo 3 caracteres, que el correo electrónico sea válido y que la contraseña tenga un mínimo de ocho caracteres. En el caso de que no exista ningún error, el administrador al pulsar en el botón Guardar se redirige a la función crear() de AdminController.java.

Primero se genera una instancia de un objeto BCryptPasswordEncoder, que es una clase que se utiliza para codificar las contraseñas de los usuarios de manera segura. Se genera un número aleatorio que se utiliza como el código

de identificación del coordinador, y se verifica que no existe otro usuario con el mismo código. Se establecen otros atributos del coordinador como el estado de la cuenta a ALTA y la foto de perfil por defecto, y se guarda en la base de datos.

Por último, se agregan algunos elementos al modelo para redirigirlo al script principal admin.html.

9. Editar usuario siendo Administrador

Para que el administrador pueda editar un usuario se ha desarrollado un botón que aparece para cada usuario. Al pulsar en él se llama a la función habilitarEdicion() del JavaScript admin.js, la cual se encarga de mostrar el botón de guardar la edición y cancelar la edición, además de convertir los campos alias, correo y estado en editables.

Si se pulsa en el botón de Guardar para confirmar la edición, se redirige a la función actualizarUsuario() de AdminController.java y se actualiza el usuario en la base de datos.

10. Eliminar cuenta siendo Administrador

En la vista admin.html, donde aparecen todos los usuarios, se ha implementado un botón para cada usuario con el objetivo de que el administrador pueda eliminarlo.

Desde el archivo admin.js, a través de un evento de click, se detecta cuando el administrador pulsa en el botón de Eliminar y aparece un modal para confirmar si se quiere realizar la acción. En ese caso se redirige a la función eliminarUsuario() de AdminController.java.

Si el usuario que se está eliminando es un participante se eliminan los datos de sus materiales, sesiones, objetivos, las fichas realizadas, los mensajes que tenga asignados. Además se le elimina del grupo al que estaba asignado y todas sus notificaciones, entre otras cosas.

Si es un coordinador se eliminan los mensajes, las observaciones escritas, etc.

Por último, si es un administrador se eliminan los mensajes.

11. Filtrar usuarios siendo Administrador

Dentro del script admin.html se ha agregado un input de búsqueda en la parte superior. Para implementarlo se ha definido una función de jQuery en admin.js, en la cual se asigna un evento keyup, que se ejecuta cada vez que el usuario suelta una tecla después de haber escrito en el campo de texto.

Se crea una expresión regular para que el texto escrito no sea sensible a mayúsculas y minúsculas. Después oculta todas las filas de la tabla y muestra solo las que coinciden con la expresión regular.

Esta función de filtrado se ejecuta cada vez que el administrador modifica el campo de entrada del buscador, permitiendo una búsqueda dinámica a tiempo real en la tabla.

12. Enviar mensaje incidencia siendo Administrador

En la vista principal del administrador, admin.html, aparece un listado de todos los usuarios de la aplicación. En cada uno de ellos se ha agregado un botón para enviar un mensaje de incidencia. Al hacer click sobre él, se redirige a la función abrirIframe() que se encarga de abrir un modal donde aparece un formulario para escribir el mensaje y enviarlo. Cuando pulsa en Enviar se redirige a la función guardarMensajeAdmin() implementada en AdminController.java.

En ella se comprueba si el mensaje no está vacío y se crea un diccionario de palabras ofensivas que se cambian por asteriscos en el caso de ser alguna de ellas. Para comprobar si es una palabra ofensiva, se separa el mensaje en palabras individuales y se hace un bucle por cada palabra para censurarlas en caso de ser necesario.

Luego se obtiene el usuario que ha iniciado sesión y se establecen algunos atributos como el emisor, la fecha y la hora de envío y el destinatario.

Por último, se guarda el mensaje en la base de datos y se redirige al script mensajesAdmin.html.

7.2.2. Actividades

1. Acceder a las actividades siendo Participante

Para llevar a cabo este caso de uso, desde el controlador ParticipanteController.java se ha implementado la función actividades(Model model). En ella se realiza una comprobación del tipo de usuario para devolver un script distinto según el rol. Por ejemplo, en el caso de ser un participante, a través del Service de actividades se realizan tres consultas: buscar todas las actividades que aun no han sido realizadas, obtener todas las actividades a las que el participante ha confirmado su asistencia y el listado total de las notificaciones correspondientes al participante. En el caso del administrador se devolverá la vista principal, ya que no tiene acceso a esta funcionalidad.

2. Ver invitaciones siendo Participante

Una vez el usuario se encuentra en la vista del script actividades.html, tiene la opción de pulsar el botón Invitaciones. Al realizar dicha acción, redirige al participante a la función invitacionesPart(Model model) del ParticipanteController.java.

En la función, primero se comprueba que el usuario sea un participante además de estar dado de alta, ya que se necesita para tener acceso a esta sección. Seguidamente se realizan varias consultas a la base de datos a través del Service de invitaciones, para buscar las invitaciones pendientes de contestar, las rechazadas, las aceptadas y las notificaciones del usuario. Una vez obtenidos todos los datos necesarios, éstos se pasan a través del model al script devuelto por la función, invitacionesPart.html, el cual es un panel de invitaciones.

En el caso de que el usuario no sea un participante, se devuelve el script de la vista principal.

3. Aceptar una invitación siendo Participante

Como se comenta en el caso anterior, un tipo de invitación son las invitaciones aceptadas. Desde el script de invitacionesPart.html, al lado de cada invitación pendiente aparece el botón para aceptar. Si el participante pulsa en el botón, se le indexa el id de la invitación y le redirige a la función confirmarInvitacion() de ParticipanteController.java.

Primero se comprueba que sea un participante y que esté dado de alta. Para poder realizar las actualizaciones necesarias, se hacen consultas para obtener la invitación que se quiere aceptar, la actividad a la que corresponde la invitación y el participante. A través de getters (función para acceder a un valor privado de una clase) y setters (función para modificar un valor privado de una clase), se llevan a cabo las actualizaciones oportunas como cambiar el número de participantes que asisten a la actividad a través de un setter al atributo numParticipantes. Una vez realizadas todas las actualizaciones, se realiza una consulta a la base de datos de update para la actividad y otra para la invitación.

Por último, en el caso de que el número de plazas de la actividad se haya quedado a cero, hay que enviar una notificación con el contenido “Invitación eliminada a la actividad: [nombre actividad]. Sin disponibilidad” a todos los participantes que tuvieran una invitación a esa actividad, ya que ya no van a poder asistir. Por tanto, a través del listado de todas las invitaciones a la actividad concreta que estén pendientes, se crea una notificación y se guarda en la base de datos. Como resultado, la función se redirige al panel de invitaciones.

4. Rechazar una invitación siendo Participante

Otro tipo de invitación, son las invitaciones rechazadas. Como en el caso anterior, en cada invitación hay un botón Rechazar. Si el participante pulsa en él, la aplicación le redirige a la función rechazarInvitacion() del controlador del participante, con el id de la invitación indexado.

Se comprueba si el usuario es un participante además de estar dado de alta y posteriormente se actualiza el estado de la invitación a Rechazada con una consulta a la base de datos. La función devuelve el script panelInvitaciones.html como resultado.

5. Ver una actividad siendo Participante

Para ver la información de una actividad, en el script de actividades.html cada actividad tiene un botón que le permite al participante ver la información, además de poder pulsar directamente en el enlace del nombre de la actividad.

Una vez se pulsa en el botón, se le redirige a la función actividad() del controlador del Participante al que se le pasa por @PathVariable el id de la actividad seleccionada.

En primer lugar, se distingue entre si es un participante o un coordinador, ya que un administrador no puede acceder.

Si es un participante y está dado de alta, se implementa una variable booleana la cual será true si el participante ya ha confirmado la asistencia a esa actividad o false si por el contrario no lo ha hecho. Esto sirve para que no se le permita volver a confirmar y se le cuente más de una vez su participación. Además, como en todas las vistas del participante, se realiza una consulta a la base de datos para obtener todas las notificaciones del participante.

Como resultado de la función, se devuelve el script actividad.html donde aparece la descripción, los detalles y la acción de inscribirse de la actividad.

En el caso del coordinador se comentará en el caso de uso correspondiente.

6. Realizar inscripción a una actividad siendo Participante desde la vista de Actividades

Estando en el mismo script que en el anterior caso, actividades.html, cada actividad tiene el botón que le permite ver la información, pero al mismo tiempo puede realizar la inscripción directamente.

Cuando el participante pulsa en ese botón, el sistema le lleva a la función de JavaScript inscripcion() al que le pasa como parámetro el id de la actividad. Esta función muestra un modal para que afirmes si quieras realizar la inscripción. En el caso de darle a Confirmar, se redirige a la función confirmar() de ParticipanteController.java, pasándole el id de la actividad.

Se comprueba si el participante está dado de alta y se realizan los siguientes pasos.

En primer lugar, se hace una consulta a la base de datos a través del Service de actividades para obtener la entidad de la actividad correspondiente y poder acceder a sus datos a través de setters y getters, y se comprueba si la actividad tiene plazas disponibles.

En el caso de que sí que haya disponibilidad, se actualizan los participantes de la actividad, y el número de plazas. Como pasa con el caso de uso “Aceptar una invitación”, es necesario comprobar si la actividad se ha quedado sin disponibilidad para avisar a todos los participantes que hubieran recibido una invitación. Por lo tanto, se obtienen las invitaciones a esa actividad concreta con el método findByAct() del Service y a las que tengan el estado como Pendiente, se les enviará una notificación de la invitación ha sido eliminada.

Por el contrario se devolverá el script de actividades.html.

7. Realizar inscripción a una actividad siendo Participante desde una actividad concreta

El procedimiento anterior, también se puede realizar desde el script de actividad.html. Para llegar a esa sección es necesario navegar por las 3 opciones superiores: descripción, detalles e inscripción. Para cambiar de opción se utiliza en el javascript actividades.js una serie de eventos, en concreto .click, para que según haga click en una opción le muestre la información correspondiente.

Por lo tanto, una vez dentro de “Inscripción”, el participante tiene que pulsar en el botón de “Confirmar asistencia” si desea inscribirse a la actividad. En ese

caso se redirige a la función inscripcion() del controlador del participante y se hace la misma implementación descrita en el caso anterior.

8. Ver inscripciones siendo Participante

En el script de actividades.html, en la parte superior derecha aparece el botón de “Asistencia”. Si el participante pulsa, aparece un modal (ventana emergente) a través de la función abrirIframe() de JavaScript.

La información que se muestra es una especie de calendario compuesto por listado de todas las actividades a las que el participante se ha inscrito, con la opción de poder eliminarlas.

9. Eliminar inscripción a una actividad siendo Participante

Como se explica en el caso anterior, en el calendario del listado de actividades inscritas está la opción de eliminar una inscripción. Por lo tanto, cuando se pulsa en el ícono de Eliminar, se redirige a la función desinscripcion() de actividades.js, que es un evento de click. En ella se implementa el modal que aparece donde te pide la confirmación de eliminar la inscripción.

Por lo tanto, si el participante pulsa en Eliminar, se redirige a la función de eliminarinscripcion() del controlador del participante, con el id de la actividad indexado. Primero se realizan dos consultas, una para obtener el participante y otra para la actividad.

Luego se actualizan tres atributos de la actividad a través de setters. Se elimina el participante para eliminar su asistencia, el número de participantes disminuye y el número de plazas aumenta.

Una vez realizado esto, se guarda la actividad en la base de datos con una consulta de update.

10. Filtrar actividades siendo Participante

Para filtrar las actividades se ha hecho uso de un buscador y un desplegable de categorías, que trabajan en conjunto. El funcionamiento consiste en buscar una actividad y la aplicación te devuelve el resultado a la vez que estás escribiendo.

Se ha implementado una función en JavaScript en actividades.js, en la que se aplica un evento de keyup. Este evento se dispara cuando el usuario suelta una tecla del teclado después de haberla presionado. La función obtiene el valor a tiempo real del valor buscado, además de la categoría que hay seleccionada en el select (desplegable). A través de un bucle, va recorriendo todas las actividades que se muestran y según el valor buscado y la categoría, filtra el listado.

11. Gestionar las actividades siendo Coordinador

Para que el coordinador acceda a las gestión de las actividades, desde el controlador del participante está implementada la función actividades(). Primero se hace una distinción por roles de usuario, por lo que en el caso del coordinador, se obtienen todas las actividades creadas en la base de datos y se pasan a través del model a la vista actividadesCoor.html.

Además, para poder crear una nueva actividad en la vista, se crea un nuevo objeto de la entidad Actividad.

12. Crear actividad siendo Coordinador

Para que el coordinador pueda crear una actividad, tiene que pulsar en el botón de Nueva actividad situado en la parte superior derecha del script actividadesCoor.html.

Al realizar la acción, a través de la función abrirIframe() de JavaScript, se crea un modal donde aparece el formulario con los distintos campos para crear una actividad.

Mientras el coordinador rellena el formulario, se hace una validación de los datos con las implementaciones realizadas en validaciones.js. Para este caso, se comprueba que la fecha de realización sea posterior a la actual, que el número de plazas se encuentre entre 1 y 150 y que el nombre de actividad cumpla con los requisitos de longitud de caracteres. También hay una función implementada en JavaScript que comprueba que el formato de la imagen sea PNG, JPEG o JPG. Después de dar al botón de guardar sin tener errores en los campos, se redirige a la función processActividad() de CoordinadorController.java, donde se inicializa el número de participantes a cero y se guarda la actividad en la base de datos.

13. Ver una actividad siendo Coordinador

En el caso del coordinador, para ver una actividad tiene que estar en la vista de actividadesCoor.html, donde se muestra un listado con todas las actividades. Para ver una actividad, es necesario pulsar en el botón Información.

Al realizar esta acción, se redirige a la función actividad() con el id indexado, donde se lleva a cabo una consulta para obtener los asistentes a esa actividad. Como resultado se devuelve la vista de actividad.html.

14. Ver invitaciones de una actividad siendo Coordinador

Para ver las invitaciones, el coordinador tiene que situarse en el script de actividadesCoor.html y pulsar en el botón de Invitaciones de la actividad seleccionada. Seguidamente se redirige a la función invitaciones() del controlador del coordinador con el identificador de la actividad indexado.

Primero se comprueba que el usuario es un coordinador. Después, a través de consultas llamando a los Service correspondientes, se obtiene la actividad, los grupos pertenecientes al coordinador y las invitaciones que ha realizado el coordinador a la actividad elegida. Por último, la función devuelve la vista de invitaciones.html.

15. Enviar invitación de una actividad siendo Coordinador

Dentro del script de invitaciones.html, se encuentra un formulario donde se llevan a cabo las invitaciones a las actividades. Aparece un desplegable para elegir entre enviar una invitación grupal o una invitación individual.

Una vez seleccionado el tipo de invitación, a través de eventos de JavaScript en actividades.js, si es una invitación grupal se muestra un desplegable con el nombre de los grupos al que enviar la invitación. Por el contrario, si es una invitación individual, muestra un cuadro de texto donde hay que introducir el código del participante destinatario.

Al pulsar en el botón de Enviar, se redirige a la función processinvitacion() del CoordinadorController.java.

En el caso de ser una invitación grupal, se obtiene con una consulta el grupo y los participantes correspondientes. Seguidamente, por cada participante del grupo, se comprueba si ya tiene una invitación a la actividad seleccionada y si no es el caso, se envía una invitación además de una notificación para avisar al participante.

Por otro lado, si es una invitación individual, lo primero que se hace es comprobar si el código introducido pertenece a un usuario y si éste es un participante. Si no existe se muestra un modal advirtiendo al coordinador. Pero si el código si que es de un participante, se hace el mismo procedimiento que para los grupos, se crea una nueva invitación con estado pendiente y se envía una notificación.

16. Editar actividad siendo Coordinador

Desde el script de actividadesCoor.html, donde se muestra el listado de todas las actividades, aparece un ícono de Editar para cada actividad. En el momento en el que se pulsa en él, se redirige a la función editarActividad() del controlador del coordinador.

En la implementación, primero se comprueba si el usuario es un coordinador. De este modo, si cumple con la condición, se pasa la actividad seleccionada al script editarActividad.html, donde se realiza la modificación de los datos.

En este script, aparece un formulario con toda la información de la actividad que se puede editar. Como se ha comentado en casos de uso anteriores, mientras el coordinador rellena los campos, se realiza una validación a través de validaciones.js, para comprobar si existen errores. Algunas de las comprobaciones son que la fecha de realización sea posterior a la actual o que el número de plazas se encuentre entre 1 y 150.

Cuando los campos modificados no den ningún error, se puede pulsar el botón de Guardar y se redirige a la función guardaractividad() implementada en CoordinadorController.java.

En esta segunda función del caso de uso, se actualizan los atributos de la actividad que hayan sido modificados a través de setters.

17. Eliminar actividad siendo Coordinador

Para realizar este caso de uso, se ha implementado un botón de Editar en cada actividad del listado mostrado en actividadesCoor.html. Una vez se pulsa en este botón aparece un modal (ventana emergente) de JavaScript, para confirmar que se quiere realizar la acción.

En el caso de pulsar en Confirmar, se redirige a la función eliminarActividad() implementada en CoordinadorController(), donde se realizan una serie de operaciones.

En primer lugar, se comprueba que el usuario sea un Coordinador. Después, se eliminan todas las invitaciones que existan a esa actividad, ya que no será posible asistir a ella. Por último, se elimina la actividad de la base de datos con el método delete.

18. Filtrar actividades siendo Coordinador

Para realizar la filtración de las actividades, se ha seguido la misma implementación que para el buscador de las actividades por parte del participante. La diferencia es la vista donde se encuentra cada uno de los buscadores y que el total de actividades en la vista del coordinador es mayor que en la del participante, debido a que aparecen todas las actividades sin filtrar por fecha ya realizada.

7.2.3. Materiales

1. Ver materiales siendo Participante

Para hacer este caso de uso, en primer lugar, hemos implementado en la clase HomeController.java una función llamada mostrarMateriales(). En ella, primero se hace el control de acceso y se devuelve un script de vista distinto en función del actor que esté accediendo. Por ejemplo, si el que está accediendo es un coordinador, le devuelve el script materialesCoor.html. Si el que está accediendo es un participante, distingue si el estado de su cuenta es ALTA o BAJA y en función del resultado devuelve el script correspondiente. Para el caso del coordinador y del participante que está de ALTA, se añade al modelo la lista con los materiales que están asignados al participante. Dentro del script materialesPart.html se muestran la lista que se había añadido al modelo, un filtro y la opción de descargar cada material.

2. Descargar un material siendo un Participante

Dentro del script materialesPart.html se encuentra la opción de descargar un material. Una vez se pulse en el ícono de descargar, se le indexa el id del material y le dirige a la función de descargar() dentro de CoordinadorController. Esta función, comprueba en primer lugar si la persona que ha intentado descargar el archivo tiene permiso para hacerlo. Luego, utiliza la clase 'Path' para obtener la ruta del archivo en el sistema de archivos y la clase 'Files' para leer todos los bytes del archivo. A continuación, establece los encabezados HTTP apropiados, incluyendo el tipo de contenido y la longitud del contenido, y utiliza la clase 'ContentDisposition' para establecer el nombre de archivo original en el encabezado Content-Disposition. Finalmente, la función devuelve una respuesta HTTP con el cuerpo del archivo como un array de bytes, junto con los encabezados establecidos anteriormente y se descarga el archivo. Para implementar la función hemos usado varias clases de la biblioteca org.springframework.http.

3. Filtrar materiales siendo Participante

Dentro del script materialesPart.html, hay un filtro que permite al usuario buscar los materiales en función de su nombre o su descripción. Para ello, hemos implementado un buscador con javascript que compara el texto introducido con el que se encuentra dentro de la clase html llamada material, y devuelve los resultados obtenidos.

4. Enviar un material a un participante en concreto siendo Coordinador

Dentro del script materialesCoor.html hemos implementado dos formas para poder enviar un material, desde un botón o directamente desde el formulario que aparece al final de la página. Desde un botón se muestra el formulario creando un modal, usando jquery y javascript. Cuando el coordinador rellena el formulario y le da a Añadir, le dirige a una función del HomeController llamada procesoMaterial(). Dentro de esta función se modifican los datos del material, por ejemplo poniendo el atributo grupo a null, ya que se lo está enviando a una persona concreta y no a un grupo. A continuación, usamos Path para modificar la ruta del material enviado y Files para guardarla en la carpeta de materiales dentro de la aplicación. Por último, guardamos el nuevo material en la base de datos.

5. Ver los materiales enviados a un participante en concreto siendo Coordinador

Dentro de la clase HomeController.java usamos una función llamada mostrarMateriales(). En ella, primero se hace el control de acceso y se comprueba que sea el coordinador el que esté accediendo. Luego, se añade al modelo la lista con los materiales que están asignados al participante. Y devuelve el script materialesCoor.html.

6. Filtrar los materiales enviados a un participante en concreto siendo Coordinador

Dentro del script materialesCoor.html, hay un filtro que permite al coordinador buscar los materiales en función de su nombre o su descripción. Para ello, hemos implementado un buscador con javascript y jquery que compara el texto introducido con el que se encuentra dentro de la clase html llamada buscarParticipantes, y devuelve los resultados obtenidos.

7. Descargar un material de los que se han enviado a un participante siendo Coordinador

Dentro del script materialesCoor.html se encuentra la opción de descargar un material. Una vez se pulse en el icono de descargar, se le indexa el id del material y le dirige a la función de descargar() dentro de CoordinadorController explicada anteriormente.

8. Eliminar un material de los que se han enviado a un participante siendo Coordinador

Dentro del script materialesCoor.html se encuentra la opción de eliminar un material. Una vez se pulse en el ícono de eliminar, se abre una alerta usando sweetalert y javascript que te da la opción de confirmar si de verdad quieres eliminar el material. Si confirmas la operación, te dirige a la función de procesElimMaterial() dentro de ParticipanteController que elimina el material de la base de datos.

9. Enviar un material a un grupo en concreto siendo Coordinador

Dentro del script materialesGrupo.html hemos implementado dos formas para poder enviar un material, desde un botón o directamente desde el formulario que aparece al final de la página. Desde un botón se muestra el formulario creando un modal, usando jquery y javascript. Cuando el coordinador rellena el formulario y le da a Añadir, le dirige a una función del GruposController llamada procesoMaterial(). Dentro de esta función se modifican los datos del material, por ejemplo poniendo el atributo grupo al grupo indicado. A continuación, usamos Path para modificar la ruta del material enviado y Files para guardarla en la carpeta de materiales dentro de la aplicación. Por último, guardamos el nuevo material en la base de datos de todos los participantes del grupo.

10. Ver los materiales enviados a un grupo en concreto siendo Coordinador

Dentro de la clase GruposController.java usamos una función llamada mostrarMateriales(). En ella, primero se hace el control de acceso y se comprueba que sea el coordinador el que esté accediendo. Luego, se añade al modelo la lista con los materiales que están asignados al grupo, para ello se hace una búsqueda de todos los materiales de los participantes que tienen asignado ese grupo y se eliminan los duplicados. Y devuelve el script materialesGrupo.html.

11. Filtrar los materiales enviados a un grupo en concreto siendo Coordinador

Dentro del script materialesGrupo.html, hay un filtro que permite al coordinador buscar los materiales en función de su nombre o su descripción. Para ello, hemos implementado un buscador con javascript y jquery que compara el texto introducido con el que se encuentra dentro de la clase html llamada buscarParticipantes, y devuelve los resultados obtenidos.

12. Descargar un material de los que se han enviado a un grupo siendo Coordinador

Dentro del script materialesGrupo.html se encuentra la opción de descargar un material. Una vez se pulse en el ícono de descargar, se le indexa el id del material y le dirige a la función de descargar() dentro de CoordinadorController explicada anteriormente.

13. Eliminar un material de los que se han enviado a un grupo siendo Coordinador

Dentro del script materialesGrupo.html se encuentra la opción de eliminar un material. Una vez se pulse en el ícono de eliminar, se abre una alerta usando sweetalert y javascript que te da la opción de confirmar si de verdad quieres eliminar el material. Si confirmas la operación, te dirige a la función de processElimMaterial() dentro de GruposController que elimina el material de la base de datos, buscando todos los materiales de los participantes y eliminando los que tengan el grupo asignado y el mismo link, nombre y descripción.

7.2.4. Progreso

1. Acceder a progreso propio siendo Participante

La implementación de este caso de uso comienza en la función progreso() de ParticipanteController.java. Primero se obtiene el participante al que pertenece el progreso con el objetivo de comprobar si existe y si está dado de alta. Seguidamente se diferencia entre si el usuario que ha iniciado la sesión es un participante o un coordinador, por el contrario se devolverá como resultado la vista principal.

En el caso de ser un participante, primero se comprueba que el progreso al que está accediendo sea del mismo usuario que ha iniciado la sesión, ya que el participante solo puede ver su progreso propio. Despues, se realizan varias consultas a la base de datos para obtener los listados del progreso del peso, del progreso del perímetro y de las sesiones, que se pasan a través del model al script progreso.html.

Para calcular el índice de masa corporal que aparece en la vista del progreso, se divide el peso en kilogramos por la estatura en metros cuadrados. De este modo, primero se hace un bucle por los formularios de la fase de valoración para obtener el formulario de exploración. Luego se hace un get de exploración para coger el dato de la altura, que posteriormente se dividirá entre 100 para pasarla a metros, ya que en la base de datos está en centímetros.

El siguiente paso que se implementa es buscar el último registro del peso con una consulta que realiza una búsqueda por fecha y tipo de progreso. Si el participante no ha registrado ningún dato en el progreso, el dato que se coge para realizar el cálculo del IMC (índice de masa corporal) es el que se introdujo en el formulario de exploración.

Por último, se realiza el cálculo del peso entre la altura al cuadrado y se le aplica el formato de dos decimales. Además, se hace una consulta para coger todas las notificaciones del participante y pasárselas al script.

Los datos del IMC se muestran a través de un canvas de JavaScript. Primero se obtiene en progreso.js el dato del índice de masa corporal para clasificarlo en normopeso, sobrepeso u obesidad. Luego se crea un gráfico de tipo donut realizando una instancia a la clase Chart, donde se añade el dato del IMC y el color correspondiente a la clasificación.

2. Ver gráfica de peso siendo Participante

Todas las acciones del dashboard de progreso se realizan en la misma vista, existiendo una diferente para participante y coordinador.

Para ver la gráfica primero se ha implementado un botón en la sección Peso del dashboard. Al realizar un evento de click sobre el botón, se detecta en el archivo JavaScript progreso.js, donde se hace un show (mostrar elemento HTML) de la gráfica y un hide (ocultar elemento HTML) de la tabla que contiene los registros del peso. De esta forma el participante ya puede ver la gráfica del peso.

Por otro lado, para crear la gráfica se hace en un elemento canvas, que es una etiqueta HTML para crear gráficos en JavaScript. Se crea una instancia de la gráfica en una variable y se realiza su configuración especificando el tipo de gráfica.

Además, se realiza una petición HTTP con la función fetch para llenar los datos en la gráfica. Seguidamente, se redirige a la función datosPeso() implementada en ApiRestController.java, la cual devuelve como resultado los datos del peso correspondientes al participante. Al terminar la petición, en la función then se transforma el resultado obtenido en formato JSON y por último a través del método push se añaden los datos actualizados en la gráfica.

3. Filtrar peso siendo Participante

En la sección del peso, aparecen justo arriba dos campos de tipo Date para indicar el intervalo de fechas de los que se quiere obtener los datos del peso. Por lo tanto, en el archivo progreso.js existen dos variables, iniPeso y finPeso, que están inicializadas a cero. En el caso de introducir una fecha se detecta con el evento change en el JavaScript y se llama a la función actualizarPeso() de progreso.js.

Lo primero que se hace es una petición HTTP realizada mediante la función fetch de JavaScript. Una petición HTTP es una solicitud que se realiza desde el cliente al servidor para obtener los resultados. Para realizarla se obtiene primero la url correspondiente y se redirige a la función datosPeso() del ApiRestController.java.

En esta función se comprueba si se ha seleccionado alguna fecha en uno de los campos o en ambos, y según eso se realiza la consulta correspondiente a la base de datos.

Después de realizar la petición, se procesa la respuesta recibida en formato JSON mediante una serie de operaciones para transformar y agregar los datos recibidos al gráfico de peso. Por lo tanto, la función then primero convierte los datos a formato JSON y después se itera sobre los elementos para obtener los datos necesarios y se agrega cada dato recibido al gráfico utilizando el método push.

Finalmente, se llama al método update del gráfico del peso para actualizar la visualización con los datos agregados.

4. Ver registros de peso siendo Participante

Para observar los registros del peso siendo un Participante, se realiza un procedimiento muy parecido que el de ver la gráfica. Se ha implementado un botón con el nombre Ver registros, el cual al hacer click sobre él, a través de eventos desarrollados en el archivo progreso.js, se oculta el elemento de la gráfica y se muestra la tabla con todos los registros del peso, apareciendo el dato, la fecha y la hora en la que se ha introducido.

5. Registrar peso siendo Participante

Con el objetivo de que el participante pueda introducir un nuevo dato del peso, se ha desarrollado un botón con un icono de (+), ubicado en la parte superior derecha de la sección del Peso en el dashboard del progreso.

Al hacer click sobre él, se redirige a la función abrirIframe() del archivo de JavaScript progreso.js, la cual se encarga de mostrar un modal (ventana emergente) que se compone de un formulario con los campos fecha y peso. Mientras el participante rellena los datos, al mismo tiempo se realiza una validación de los campos implementada en validaciones.js, que se encarga de comprobar que la fecha sea actual o anterior, ya que no puedes saber lo que pesas en el futuro, y además comprueba que el dato del peso introducido sea mayor a 0.

Para realizar estas validaciones, en validaciones.js se valida el campo de fecha cada vez que cambia con el evento change, mientras que el campo dato a través del evento keyup comprueba mientras el participante está escribiendo si está correcto o no.

Por lo tanto, para procesar los datos introducidos válidos, hay un botón Guardar que envía los datos a la función processpeso() de ParticipanteController.java.

En esta función se crea el nuevo registro introducido, por lo que primero se necesita hacer un set del progreso creado para añadir el participante correspondiente a ese registro, además de indicar que el progreso es de tipo PESO. Con el fin de obtener el valor del peso inicial que se llenó en el formulario de exploración, se realiza un bucle por todos los formularios de la fase de valoración hasta obtener el de exploración.

El siguiente paso que se implementa es obtener el dato del peso con un get para después compararlo con el nuevo peso registrado y ver si el participante ha perdido peso o ha ganado. Por lo tanto, se realiza una resta entre el peso introducido y el peso inicial que en el caso de dar positivo habrá ganado peso y si por el contrario da negativo, habrá perdido peso.

Por último, se actualiza al participante con el nuevo dato del atributo perdidaPeso a través de una consulta SQL desde el Service de Participante y se añade a la base de datos el registro del progreso del peso.

6. Ver gráfica del perímetro siendo Participante

Para implementar este caso de uso, en la sección del perímetro en el script progreso.html aparece un botón en la parte inferior con el nombre Ver registros. Al hacer click sobre él, en el archivo progreso.js se detecta la acción con un

evento de click. Por lo tanto, con el método hide (ocultar elemento HTML) se oculta la tabla de registros del perímetro y con show se muestra la gráfica del perímetro.

7. Filtrar perímetro siendo Participante

Como sucede con el peso, en el script progreso.html, arriba de la sección del perímetro aparecen dos campos de tipo Date, que sirven para que el participante indique el intervalo de fechas de los que quiere obtener datos del perímetro. Entonces, en el archivo progreso.js se definen las variables iniPerimetro y finPerimetro inicializadas a cero.

El procedimiento es el mismo que en "Filtrar peso siendo Participante", por lo que al introducir una fecha se detecta con un evento change en progreso.js y se llama a la función actualizarPerimetro().

Por lo tanto, se define la variable url inicializada con la ruta correspondiente para realizar la petición HTTP mediante la función fetch. Esta petición se redirige a la función datosPerimetro() del controlador APIRest para realizar la consulta correspondiente según se haya añadido una fecha u otra y obtener los datos.

Después, se procesan los datos recibidos para convertirlos a formato JSON en la función de JavaScript actualizarPerimetro() y luego se itera por cada elemento obtenido para agregarlo al gráfico con el método push. Y por último se actualizan los datos de la gráfica haciendo un update.

8. Ver registros de perímetro siendo Participante

En el script progreso.html, en concreto en la sección del perímetro, se ha implementado el botón Ver registros. Cuando el participante pulsa en él, a través del archivo progreso.js se detecta el evento click. Por lo tanto, con el método hide se oculta el elemento de la gráfica del perímetro y con el método show, se muestra la tabla con los registros del perímetro que contiene el dato, la fecha y la hora.

9. Registrar perímetro siendo Participante

La implementación de este caso es exactamente igual que el caso de registrar peso siendo participante, con la única diferencia de que se aplica a los datos del perímetro en vez de a los del peso.

10. Acceder al progreso de un participante siendo Coordinador

Para este caso de uso se ha implementado la función progreso() en ParticipanteController.java. Primero se obtiene el participante haciendo una consulta a la base de datos y se comprueba si existe y si está dado de alta.

Luego se diferencia entre si el usuario que ha iniciado sesión es un participante o un coordinador.

En el caso de ser un coordinador, se verifica que el participante del que quiere ver el progreso le pertenece al coordinador haciendo un getCoordinador del

participante, ya que solo puede acceder a los progresos de los participantes que gestione.

Por último, se le pasa al script `progresoCoor.html` el peso perdido del participante y el perímetro perdido, además del participante.

11. Ver gráfica de peso siendo Coordinador

Para implementar este caso, desde la vista de `progresoCoor.html` se ha implementado en la sección del Peso el botón Ver gráfica. En `progresoCoordinador.js` se detecta el evento click cuando se pulsa en el botón, donde se hace un show de la gráfica para mostrarla y un hide de la tabla para ocultarla.

Para crear la gráfica en un elemento canvas (etiqueta HTML que permite crear gráficos y animaciones a través de scripting en JavaScript), se utiliza la librería `Chart.js`. En la variable `pesoCoor` se crea la instancia de la gráfica y se le pasa un objeto de configuración donde se especifica que la gráfica es lineal.

Por otro lado, para llenar los datos en la gráfica, se realiza una petición HTTP mediante la función `fetch` de JavaScript. Se redirige a la función `datosPeso()` implementada en el Rest Controller `ApiRestController.java`, que se encarga de devolver los datos correspondientes al participante seleccionado.

Después de la petición, se transforma el resultado en formato JSON y con el método `push` se van metiendo los datos en la gráfica.

12. Ver registros de peso siendo Coordinador

En la vista de progreso individual a la que accede el coordinador, para que éste pueda ver los registros del peso con la fecha y hora, se ha implementado un botón con el nombre Ver registros en la parte inferior de la sección del peso. Al hacer click sobre él, se detecta el evento click en `progresoCoor.js`, por lo que primero se oculta la gráfica con un `hide` y después se muestra la tabla con los registros de cada dato con un `show`.

13. Ver gráfica de perímetro siendo Coordinador

En la vista `progresoCoor.html` aparece un botón Ver gráfica en la parte de los datos del perímetro. A través del evento click de JavaScript, cuando el coordinador pulsa en el botón se realizan dos acciones: un `show` de la gráfica y un `hide` de la tabla de datos.

Como se realiza con el peso, para crear la gráfica en un elemento canvas, se crea una instancia en una variable en la cual se implementa la configuración del formato de la gráfica.

Para llenar la gráfica con la información de los datos del perímetro, se ejecuta una petición HTTP utilizando la función `fetch` de JavaScript. Esta petición redirige a la función `datosPerimetro()` dentro de `ApiRestController.java`, la cual se encarga de devolver como resultado los datos del perímetro. Posteriormente, el resultado se transforma en formato JSON y se introducen los datos en la gráfica con el método `push`.

14. Ver registros de perímetro siendo Coordinador

En la parte inferior de la sección del perímetro, se ha añadido un botón para poder ver los registros del perímetro. En el momento en el que el coordinador pulsa en él, a través del evento click implementado en el archivo progreso-Coor.js, se oculta la gráfica del perímetro y se muestra la tabla con los datos del perímetro.

7.2.5. Chat

1. Acceder al chat privado con el coordinador siendo participante

Para implementar este caso de uso, en el script GruposController.java se ha realizado la función chat(Model model). En ella, tras comprobar que el usuario está dado de alta como participante, se hacen diversas consultas a la base de datos mediante servicios. Por ejemplo, para obtener la lista de mensajes intercambiados entre el participante y su coordinador. De este modo, se pasan las variables necesarias a la vista chat.html donde se muestra un botón para seleccionar que se visualice el chat privado con el coordinador. En este html, el diseño del chat se divide en distintas secciones. En primer lugar, se muestra una cabecera para indicar que se trata de la conversación con el coordinador. En la zona central, se muestran los mensajes intercambiados. Para ello, se recorre la lista de mensajes privados recibida desde el controlador diferenciando entre los mensajes enviados por el participante y los enviados por su coordinador, para poder distinguir el aspecto visual (css) de los mismos. Así, para cada mensaje, se indica el emisor, contenido y día y hora en que se ha enviado junto con el avatar del usuario correspondiente. La tercera zona del chat es la encargada de enviar los mensajes, y se comentará con detalle más adelante.

2. Acceder al chat privado con el administrador siendo participante

En caso de que el participante quiera solicitar ayuda, puede hacerlo a través de este caso de uso. Para implementarlo, se ha creado una función chatPartAdmin en el HomeController.java en la que primero se comprueba que el rol del usuario sea el de participante y después, a través de servicios, se obtiene la lista de mensajes intercambiados con el administrador de la aplicación. Esta lista, es la que se pasa al chatPartAdmin.html donde se muestra el chat correspondiente. Para que el participante pueda acceder a ella sin importar en qué parte de la aplicación se encuentre navegando, se ha añadido en todos los scripts del participante un apartado de ayuda que referencia a la vista del chatPartAdmin en el desplegable del perfil.

3. Acceder al chat grupal siendo participante

El proceso seguido para implementar este caso de uso es muy similar al realizado para implementar el chat privado con el coordinador. De este modo, en la función chat(Model model) de GruposController.java, tras comprobar que el usuario está dado de alta como participante, se hacen consultas mediante servicios a la base de datos. Así, se obtienen los mensajes intercambiados por todos los participantes miembros de un grupo y su coordinador para después

pasarlos al html. Para realizar esta consulta, es necesario pasar como argumento el grupo del que se obtienen los mensajes. Finalmente, en el chat.html se recorre la lista de mensajes diferenciando entre enviados y recibidos por el participante. Además, es esta conversación la que aparece por defecto, pero para volver a mostrarla en caso de que se esté visualizando el chat con el coordinador, también se incluye un botón para el chat grupal. De hecho, aunque se comentará en la parte de enviar mensajes, con javascript se regula qué chat de los dos se va a mostrar.

4. Acceder al chat privado con el administrador siendo coordinador

Para solicitar ayuda al administrador siendo coordinador, se ha implementado la función chatCordAdmin en el HomeController.java. En esta función, tras realizar el control de acceso asegurando que el usuario que accede es de tipo coordinador, se obtiene la lista de mensajes intercambiados entre éste y el administrador mediante consultas a la base de datos llevadas a cabo mediante servicios. A continuación se envía la lista a la vista html chatCordAdmin para poder visualizar en ella el chat. Además, puesto que el coordinador puede necesitar ayuda en cualquier momento, se ha incorporado en todos los scripts del coordinador un apartado de ayuda que le redirige al chatPartAdmin.html en el desplegable del perfil.

5. Acceder al chat privado con un participante siendo coordinador

Para que un coordinador pueda escribir a un participante que forma parte de uno de los grupos que dirige, se han implementado varias funciones. Si se parte de la vista del listado de grupos, en el script grupos.html en la parte de la tabla aparece un botón por cada grupo que permite redirigir la vista a la de un grupo concreto. Para que esto funcione, en el GruposController.java se crea la función unGrupo a la que se le pasa el id del grupo correspondiente. En ella, primero se comprueba que el usuario sea de tipo coordinador, y después, mediante consultas a la base de datos por servicios y sus repositorios respectivos, se obtienen distintos datos acerca del grupo como la lista de participantes. A su vez, en el html de un grupo, se recorre la lista de participantes de manera que para cada uno, se comprueba si está dado de alta o de baja para activar o no el botón Mensaje que va a permitir acceder al chat privado con dicho participante. Así, en caso de que el participante esté dado de alta, se llama a la función chatPrivado de GruposController.java donde, tras comprobar de nuevo que el usuario es un coordinador y que quiere escribir a un participante que ya existe, se obtiene por servicios la lista de mensajes intercambiados entre ambos como mensajesPrivados que redirige a la vista chatPrivado con el participante indicado donde ya se muestra la conversación.

6. Acceder al chat de un grupo siendo coordinador

Para llevar a cabo este caso de uso, puesto que el chat del grupo aparece en la vista de un grupo anteriormente mencionada, solo es necesario acceder a ella. A modo resumen, por medio de la función unGrupo que toma como argumento el id del grupo y se implementa en el GruposController.java, se comprueba que el rol del usuario coincide con coordinador y se obtienen mediante servicios

los mensajes intercambiados en el chat grupal por todos los usuarios a través de la función obtenerMensajes. Finalmente, estos mensajes se pasan a la vista y en el html junto con css se construye el chat.

7. Enviar mensaje rápido a un usuario concreto siendo administrador

Este caso de uso se realiza a partir de la vista principal del administrador. En concreto, en la función index del HomeController.java, tras comprobar que el usuario sea de tipo administrador, se pasan a la vista tanto la lista de usuarios de la aplicación (salvo él mismo) y los mensajes intercambiados con dicho administrador. De este modo, en la vista admin.html, dentro de la tabla del listado de usuarios, se implementa un botón para enviar mensajes rápidos de manera que al hacer click en él, se llama a una función abrirIframe a la que se le pasan tanto el id como el nickname del usuario al que el administrador quiere escribir. Esta función abre un modal que contiene un formulario para enviar los mensajes. Así, cuando se hace el submit, se llama a la función guardarMensajeAdmin del AdminController.java a la que se le pasa el id del usuario al que se manda el mensaje. En esta función, primero se hace un filtro de palabras bloqueadas, de manera que se comprueba que el contenido del mensaje sea adecuado, no admitiendo una serie de insultos ni palabras concretas. Después, se establecen los campos de la entidad MensajeAdmin como el emisor o fecha y hora del envío. Posteriormente, a través de los servicios y repositorio de dicha entidad, se guarda el mensaje en la base de datos. Finalmente, se redirige a la vista de mensajesAdmin con la conversación abierta, por si el administrador decide enviar más mensajes al mismo usuario.

8. Acceder a las conversaciones establecidas siendo administrador

En el caso de que el administrador quiera ver todas las conversaciones que ha establecido, puede hacerlo desde la vista de mensajesAdmin.html. Para lograr esto, se ha implementado la función mensajesAdmin en el HomeController.java. En ella, mediante el uso de servicios se obtienen tanto el listado con el resto de usuarios como el de los mensajes intercambiados. Además, se distingue si la llamada a esta función viene tras pulsar en la opción de hablar con un usuario concreto o simplemente pinchando en la zona mensajes del menú, para decidir qué chat es el que se mostrará abierto. Por último, la función redirige al html donde aparecen la tabla de usuarios con un chat para que, dependiendo del usuario escogido, se muestre una conversación u otra.

9. Enviar mensaje al coordinador siendo participante

Para implementar este caso de uso, se parte del acceso a la vista de chat.html comentado anteriormente. En particular, puesto que en la vista hay dos opciones de chat, para enviar el mensaje al coordinador lo primero que se ha implementado es una clase con los dos botones que te permite escoger un chat u otro. Una vez escogido el chat con el coordinador, éste se muestra y en la parte inferior tiene una zona, que es la que nos interesa, para enviar los mensajes. De este modo, la zona inferior contiene un formulario con el input para escribir el mensaje que se desea enviar y el botón de submit. Así, cuando

el participante escriba, tanto pulsando la tecla enter como dándole al icono para enviar, se llama a la función guardarMensajePrivado del GruposController.java a la que se le pasa el id del emisor del mensaje, además del contenido del mismo. El siguiente paso que se realiza en la función anterior es comprobar que el contenido del mensaje cumpla unos requisitos, es decir, que no contenga palabras bloqueadas. Después, a través de los servicios y repositorio respectivos, se establecen elementos del mensaje como la fecha y hora de envío para finalmente guardarla en la base de datos llamando a la función saveMensaje-Privado. Por último, se hace una comprobación para ver quién es el que está enviando el mensaje para redirigir la aplicación a la vista indicada. En este caso, se redirige al participante a la vista del chat.

Además, cabe destacar que para que al enviar el mensaje o refrescar la página se mantenga el chat escogido, ya que por defecto se muestra el chat grupal, se ha utilizado javascript. De este modo, en el script grupos.js se han implementado varias funciones. En primer lugar, una función iniciarChat que hace que por defecto, la primera vez que el participante abre la vista, se muestre el chat grupal. Esto se hace a través del uso del local storage para guardar una variable que indica qué chat tiene que estar abierto. Por otro lado, en la función cambiarChat, dependiendo del valor de la variable (grupal o coordinador) se modifica el color del botón seleccionado y con el uso de show() y hide() se selecciona el chat que se va a mostrar, ocultado el otro. Esta función, se llama siempre que se refresca la página y siempre que se hace click en uno de los botones. Asimismo, al hacer click en cada botón, con localStorage.setItem se va modificando el valor de la variable (grupal o coordinador).

10. Enviar mensaje al administrador siendo participante

Para llevar a cabo este caso de uso, es necesario haber accedido al chat privado con el administrador como se detalla en el caso de uso 2. De esta manera, partiendo de la vista chatPartAdmin.html donde se muestra el chat con el administrador, como en todos los chats, la parte inferior contiene un formulario con un input para introducir el contenido del mensaje y un botón submit para enviarlo. Así, el action del formulario consiste en hacer un post a la función guardarMensajeAdmin2 donde se comprueba que el contenido del mensaje pase el filtro de palabras bloqueadas, se establecen los campos de la entidad mensaje correspondientes mediante servicios y se actualiza la base de datos con el nuevo mensaje. Finalmente, se redirige al participante a la misma página para poder seguir en la conversación y enviar tantos mensajes como desee.

11. Imprimir el chat privado con el administrador siendo participante

Dentro del script chatPartAdmin.html debajo del chat hay un botón Imprimir. Así, cuando el participante pulsa en él se dirige a una función javascript que abre una interfaz para imprimir la pantalla actual.

12. Enviar mensaje al chat grupal siendo participante

Para enviar un mensaje al char grupal, la implementación es muy similar a la del caso de uso 9 explicado en este apartado. En este caso, partiendo de la

vista del grupo del participante, se ha implementado un botón para escoger el chat grupal en lugar del del coordinador. No obstante, la primera vez que se accede a esa página el chat mostrado por defecto es el grupal. De este modo, siguiendo el proceso ya mencionado para elegir el chat grupal y ocultar el del coordinador se puede enviar el mensaje. Para ello, en la zona inferior de la conversación se muestra un formulario con un input para escribir el mensaje y el icono para enviarlo. Al hacer el submit del formulario se va a la función guardarMensaje desarrollada en el GruposController.java. En esta función, tras la comprobación de que el mensaje es adecuado, se establecen los campos del mensaje mediante servicios y finalmente se guarda en la base de datos a través de la función save. Por último se redirige al usuario a la misma vista para poder seguir enviando y recibiendo mensajes. Cabe señalar, que a diferencia del caso de uso 9, ahora no solo se incluyen los enviados entre participante y coordinador sino todos los enviados por los integrantes del grupo del participante así como del coordinador, por lo que la lista de mensajes recorridos es distinta, siendo en este caso la lista mensajes.

13. Enviar mensaje al administrador siendo coordinador

Para enviar al mensaje al administrador siendo coordinador, primero es necesario acceder al script chatCordAdmin.html como se detalla en el caso de uso 4. Una vez ahí, aparece el chat privado con el administrador, por lo que en la parte inferior del mismo se ha implementado un formulario con un input y un botón de submit para escribir y enviar el mensaje. Así, el formulario llama a guardarMensajeAdmin2 del AdminController.java. En esta función se establecen los campos correspondientes y se guarda el mensaje en la base de datos. Finalmente, redirige al coordinador a la vista actual para que pueda seguir visualizando la conversación y resolver sus dudas.

14. Imprimir el chat privado con el administrador siendo coordinador

Dentro del script chatCordAdmin.html, debajo del chat hay un botón Imprimir. Así, cuando el coordinador pulsa en él se dirige a una función javascript que abre una interfaz para imprimir la pantalla actual, la del chat.

15. Enviar mensaje a un participante siendo coordinador

Para enviar un mensaje a un participante concreto es necesario que el participante pertenezca a uno de los grupos dirigidos por el coordinador. De este modo, en caso de querer escribir a uno de los participantes el coordinador necesita acceder al chat correspondiente como se indica en el caso 5, es decir, a la vista chatPrivado.html. En ella, se muestra un chat, que al igual que los anteriores tiene una zona inferior donde se manda el mensaje. En este caso, la función que llama el formulario es guardarMensajePrivado. En ella, se hace la comprobación del contenido adecuado, se establecen los respectivos valores y se llama a saveMensajePrivado para actualizar la base de datos. Por último se comprueba que el usuario sea coordinador para redirigirle de nuevo a la vista de chatPrivado con el participante para seguir manteniendo la conversación.

16. Imprimir el chat privado con un participante siendo coordinador

Dentro del script chatPrivado.html, debajo del chat hay un botón Imprimir. Así, cuando el coordinador pulsa en él se dirige a una función javascript que abre una interfaz para imprimir la pantalla actual, la del chat privado con el participante.

17. Enviar mensaje al chat de un grupo siendo coordinador

Para enviar un mensaje al chat de un grupo concreto es necesario haber accedido a la vista donde se muestra dicho chat como se indica en el caso de uso 6. En esta vista en la parte derecha aparece el chat grupal. En su zona inferior, está el formulario con el input y el botón de submit que, en este caso, llama a la función guardarMensaje de GruposController.java que pasa el filtro de palabras bloqueadas, establece los valores y actualiza la base de datos. Por último, se asegura de que el usuario sea de tipo coordinador para redirigirle a la misma vista de ese grupo y que pueda seguir enviando y recibiendo los mensajes por el chat grupal además de visualizar otra información pertinente del grupo seleccionado.

18. Enviar mensaje a un usuario concreto siendo administrador

Para llevar a cabo este caso de uso es necesario haber accedido a la vista mensajesAdmin.html como se indica en el caso de uso 8. Una vez ahí, se muestran por un lado una tabla que contiene a los usuarios con los que el administrador mantiene una conversación y por otro la conversación que tiene abierta. Este mecanismo se implementa mediante una serie de llamadas, de modo que cuando el administrador pulsa sobre el icono de enviar un mensaje en uno de los usuarios, se actualiza la vista modificando el id al del usuario que se quiere escribir. Así, se abre el chat correspondiente y en la zona inferior se encuentran el input y botón submit del formulario que llama por método post a la función guardarMensajeAdmin. Esta función comprueba el contenido del mensaje, establece los valores y actualiza la base de datos para finalmente redirigir al administrador a la misma vista con el chat abierto del usuario pasado como parámetro.

7.2.6. Alimentación

1. Crear un nuevo alimento siendo Participante

Dentro del script nuevoalimento.html se encuentra un formulario a llenar, cuando se pulsa en Crear, pasa a la función de process_al() que agrega el nuevo alimento a la base de datos.

2. Ver los alimentos y nutrientes que se han consumido durante el día siendo Participante

Dentro de la clase ParticipanteController hemos implementado la función llamada alimentos() esta, en primer lugar, hace un control de acceso para cerciorarse de que sólo puedan acceder los participantes que están dados de ALTA al script alimentos.html. Una vez pasado ese control, se añade al modelo una lista con los alimentos que ha consumido durante el día y se calculan los nutrientes

que se han consumido en total. Además, se borran los alimentos consumidos de hace más de una semana que se encuentren en la base de datos para liberar espacio en la base de datos, ya que son datos que no vamos a usar más dentro de la aplicación. Finalmente, se accede al script de alimentos.html.

3. Añadir alimentos que haya consumido durante el día siendo Participante

Desde el script de alimentos.html hay un formulario al final de la página en el que se puede añadir un alimento que se haya consumido en el día. Dentro de este formulario hemos implementado un filtro en javascript para ayudar a los usuarios a seleccionar los alimentos si existen muchos en la base de datos. Para ello, se compara el texto introducido con el texto de los alimentos que hay en el enum del select, es decir, los alimentos que aparecen para seleccionar. Este filtro devuelve los índices de los alimentos que coinciden y muestra solo esos. A continuación, el usuario puede pulsar en Añadir y le dirige a la función process_alimento() de ParticipanteController, que, básicamente, agrega el alimento consumido a ese participante en la base de datos.

4. Eliminar alimentos que haya consumido

Desde el script de alimentos.html el participante puede seleccionar el alimento consumido a eliminar. Cuando pulse en el icono de la papelera se abrirá una alerta que hemos implementado usando sweetalert y javascript que te da la opción de confirmar si de verdad quieres eliminar el alimento. Si confirmas la operación, te dirige a la función de process_alimentoCeliminar() dentro de ParticipanteController que elimina el alimento de la base de datos.

5. Ver todas las recetas existentes siendo Participante

Dentro de la clase ParticipanteController hemos implementado la función llamada recetas() esta, en primer lugar, hace un control de acceso para cerciorarse de que sólo puedan acceder los participantes que están dados de ALTA al script recetas.html. Una vez pasado ese control, se añade al modelo una lista con las recetas existentes. Finalmente, se accede al script de alimentos.html.

6. Filtrar las recetas existentes siendo Participante (por nombre y por ingredientes)

Dentro del script recetas.html hemos implementado dos filtros. En primer lugar, uno que permite al usuario buscar las recetas en función de su nombre y sus ingredientes. Para ello, hemos implementado un buscador con javascript que compara el texto introducido con el que se encuentra dentro de la clase html llamada material, y devuelve los resultados obtenidos. En segundo lugar, un filtro que permite al participante seleccionar los ingredientes que quiere y que no quiere en la receta, para que se muestren sólo las que cumplen esas condiciones. Para eso, usando javascript mostramos solamente las recetas que tienen los alimentos que están entre los seleccionados que quieren y que no tienen los que no quieren.

7. Crear una nueva receta siendo Participante

Dentro del script recetas.html hemos implementado dos formas para poder crear una nueva receta, desde un botón o directamente desde el formulario que aparece al final de la página. Desde un botón se muestra el formulario creando un modal, usando jquery y javascript. Dentro del formulario, hemos implementado un filtro en javascript para ayudar a los usuarios a seleccionar los ingredientes que deseen. Para ello, se compara el texto introducido con el texto de los ingredientes que hay en el enum del select, es decir, los alimentos que aparecen para seleccionar. Este filtro devuelve los índices de los ingredientes que coinciden y muestra solo esos. Además, al seleccionar el archivo hemos añadido una validación en javascript para controlar que solamente se puedan subir archivos de tipo pdf. Cuando el participante rellena el formulario y le da a Crear, le dirige a una función del ParticipanteController llamada procesoReceta(). En esta función se inicializan los datos de la receta. A continuación, usamos Path para modificar la ruta del pdf de la receta y Files para guardarla en la carpeta de recetas dentro de la aplicación. Por último, guardamos la nueva receta en la base de datos.

8. Ver una receta en concreto siendo Participante

Desde el script de recetas.html está la opción de ver una receta en concreto pulsando en el icono del ojo. Desde ahí se le indexa el enlace de esa receta y se dirige a la función de unareceta() de ParticipanteController. Esta función abre el script de unareceta.html que usa un iframe para mostrar la receta con ese enlace.

9. Ver recetas con ingredientes parecidos a los que quería cuando el filtro no encuentre ninguna siendo Participante

Desde el script de recetas.html hemos agregado un botón que aparece solamente si el filtro no devuelve ninguna receta, para que el usuario pueda encontrar una receta con ingredientes parecidos a los que ha seleccionado que quería en el filtro. Para ello, en javascript hemos implementado la opción de que aparezca el botón cuando no se devuelvan resultado y que cuando se pulse en él se le indexen los ingredientes seleccionados que quiere y que no quiere. Una vez pulsado se le dirige a la función recetasParecidas() de ParticipanteController. En esta función, en primer lugar se controla que el usuario que esté accediendo sea un participante de ALTA. Luego se le añade al modelo un mensaje si el usuario no ha seleccionado ningún ingrediente que quiere. En segundo lugar, se usa python para encontrar recetas parecidas, llamando a recetasParecidas de PythonServiceImpl. Para hacer complementario y compatible el uso de python y java dentro de nuestra aplicación, y procurando seguir la misma estructura usada en java y respetando el modelo vista controlador, hemos usado la biblioteca de jython y hemos implementado la factoría de PythonServiceFactory para implementar manualmente el que la aplicación de java reconozca y cree el script de PythonServiceImpl en python. Además, hemos agregado las anotaciones @bean necesarias para su reconocimiento. Luego, dentro de PythonServiceImpl está la función de recetasparecidas(). En la misma, primero

se trabaja con los datos de want y dontwant para poder trabajar con ellos en python. Luego se usa la biblioteca de jaydebeapi para conectar la base de datos que usa Java Database Connectivity, con python. Recogemos los datos que nos interesan y luego usamos machine learning para encontrar las recetas que queremos. Más concretamente, usamos el algoritmo del coseno de similitud. Al usar la versión de jython 2.7.3, sólo podemos usar las bibliotecas de python 2.7.3, por lo que no hemos podido usar la biblioteca de sklearn para implementar esta función, y por ello, en vez de usar kmeans y el coseno de similitud, hemos usado solo el coseno, ya que obteníamos el mismo resultado y podíamos codificar esa función en concreto sin usar bibliotecas de python. Por lo que, con la similitud del coseno obtenemos una matriz con lo que se parecen los alimentos que hay en la base de datos entre ellos, luego seleccionamos los que se parecen a los que el usuario ha seleccionado que quiere, y, finalmente, buscamos las recetas con esos ingredientes parecidos. También tenemos que trabajar con los datos al final para que sean compatibles con java y que pueda interpretar bien la lista devuelta. Se agrega al modelo la lista devuelta y se muestra el script de recetasparecidas.html.

10. Filtrar las recetas parecidas siendo Participante

Dentro del script recetasparecidas.html hemos implementado un filtro que permite al usuario buscar las recetas en función de su nombre y sus ingredientes. Para ello, hemos implementado un buscador con javascript que compara el texto introducido con el que se encuentra dentro de la clase html llamada material, y devuelve los resultados obtenidos.

11. Ver una receta parecida en concreto siendo Participante

Desde el script de recetasparecidas.html está la opción de ver una receta en concreto pulsando en el icono del ojo. Desde ahí se le indexa el enlace de esa receta y se dirige a la función de unareceta() de ParticipanteController. Esta función abre el script de unareceta.html que usa un iframe para mostrar la receta con ese enlace.

12. Ver recomendaciones de recetas según el Participante

Desde el script de recetas.html hemos agregado un botón llamado Recomendaciones. Una vez pulsado se le dirige a la función recomendaciones() de ParticipanteController. En esta función, en primer lugar se controla que el usuario que esté accediendo sea un participante de ALTA. Luego se usa python para implementar un content based filter y un collaborative filter. El content based filter, es un algoritmo de recomendación que utiliza la información de las interacciones de un usuario para recomendar elementos. Este devuelve una lista de recetas llamando a recomendacionesindividuales() de PythonServiceImpl. En la misma, primero se usa la biblioteca de jaydebeapi para conectar la base de datos que usa Java Database Connectivity, con python. Recogemos los datos que nos interesan y luego hacemos un conteo de todos los alimentos consumidos y cogemos los cuatro que han sido más consumidos. Luego, buscamos las recetas con esos ingredientes y devolvemos esa lista. También tenemos que

trabajar con los datos al final para que sean compatibles con java y que pueda interpretar bien la lista devuelta. Por otro lado, el collaborative filter, es un algoritmo de recomendación que utiliza la información de las interacciones de varios usuarios para recomendar elementos. Este devuelve una lista de recetas llamando a `recomendacionesglobales()` de `PythonServiceImpl`. En la misma, se hace prácticamente lo mismo que con la función de `recomendacionesindividuales()`. Finalmente, se agregan al modelos las dos listas devueltas y se muestra el script de `recetasrecomendadas.html`.

13. Filtrar las recetas recomendadas siendo Participante

Dentro del script `recetasrecomendadas.html` hemos implementado un filtro que permite al usuario buscar las recetas en función de su nombre y sus ingredientes. Para ello, hemos implementado un buscador con javascript que compara el texto introducido con el que se encuentra dentro de la clase html llamada `material`, y devuelve los resultados obtenidos.

14. Ver una receta recomendada siendo Participante

Desde el script de `recetasrecomendadas.html` está la opción de ver una receta en concreto pulsando en el icono del ojo. Desde ahí se le indexa el enlace de esa receta y se dirige a la función de `unareceta()` de `ParticipanteController`. Esta función abre el script de `unareceta.html` que usa un iframe para mostrar la receta con ese enlace.

15. Ver nutrientes en dieta siendo Participante

Desde el script de `recetas.html`, se ha implementado un botón Nutrientes de manera que al pulsarlo se redirige al participante a la vista `nutrientes.html` en la que, desde la función `nutrientes()` de `ParticipanteController`, se visualiza el total de nutrientes consumidos por dicho participante en un día concreto, multiplicando los nutrientes de cada alimento consumido, como en la función de `alimentos()` de `ParticipanteController`. Además de mostrar los nutrientes, muestra un formulario para calcular el índice calórico diario recomendado y poder acceder así al caso de uso de la figura 6.101.

16. Calcular icdr siendo Participante

El cálculo del icdr se realiza en el script `nutrientes.html`. Para ello, se ha implementado un formulario de manera que al pulsar en Calcular, mediante javascript se llama a una función `validarFormulario` que comprueba que los datos introducidos son correctos (en caso contrario muestra una alerta), y posteriormente llama a otra función de javascript llamada `calcular`. Es en esta función, en la que se realiza el cálculo de los nutrientes que tiene que ingenier el participante separando en distintas categorías. Además, muestra dicha información al participante indicando si se encuentra en niveles adecuados, superiores o inferiores. También se ha implementado un modal que contiene información relativa al ICDR para que el participante tenga conocimiento sobre dicho índice.

17. Ver juego de intercambiar alimentos siendo Participante

Desde la página alimentacion.html, se muestran tres botones, uno de los cuales permite acceder al juego. Al hacer clic en este botón, se redirige a la función juego() implementada en el controlador ParticipanteController.java. Esto permite al participante iniciar el juego y disfrutar de la experiencia interactiva proporcionada.

La implementación que se lleva a cabo es realizar una búsqueda de todas las notificaciones que tiene el participante y por otro lado obtener toda la base de datos de los alimentos de intercambio para poder seleccionarlos en el juego. Y como resultado de la función se devuelve el script juego.html.

18. Añadir alimentos y calorías siendo Participante

Desde el script juego.html se ha implementado un desplegable con cuatro cantidades de calorías que indican que tipo de dieta tiene que llenar el participante en la tabla del juego.

En el archivo juego.js se definen 4 matrices que representan cada tipo de dieta e informan de la cantidad de unidades de intercambio que se tiene que llenar en cada casilla.

Para poder arrastrar los alimentos a la casilla correspondiente, se ha utilizado la función drag and drop (arrastrar y soltar). Para habilitarla en HTML, se utilizan los atributos draggable en los elementos que se pueden arrastrar y ondragstart en el elemento que inicia el arrastre.

19. Filtrar alimentos a añadir siendo Participante

Dentro del script juego.html hemos implementado un filtro que permite al usuario buscar los alimentos a insertar en función de su nombre y su grupo. Para ello, hemos implementado un buscador con javascript que compara el texto introducido con el que se encuentra dentro de la clase html llamada elemento-intercambio, y devuelve los resultados obtenidos. Además, se puede filtrar según el grupo de alimentos que quiera el participante y se muestran los resultados de los alimentos que tengan ese grupo.

20. Comprobar que esté bien siendo Participante

En la parte superior de la tabla del juego, aparece un botón para comprobar el resultado obtenido. En la función de javascript comprobarResultado(), se realiza un bucle por toda la matriz para ir verificando si el resultado de la casilla es correcto, comparando con el número de unidades que tiene que haber y el grupo correspondiente en esa casilla.

Una vez terminado el bucle, se indica con un mensaje el resultado obtenido en el juego.

21. Leer instrucciones siendo Participante

Dentro del script juego.html hay un botón de Leer instrucciones que, cuando pulsas, abre un modal usando javascript en el que se muestra el texto con las instrucciones del juego.

7.2.7. Objetivos

1. Acceder a tus objetivos siendo participante

Para llevar a cabo la implementación de este caso de uso se ha realizado la función objetivos(Model model) en el script de ParticipanteController.java. En esta función, a la que se accede al pulsar sobre el botón objetivos de la vista principal del participante, el primer paso es comprobar que el usuario que la llama sea un participante mediante el uso de instanceof. Entonces, una vez realizada esta comprobación, se utilizan los servicios y repositorios de las distintas entidades para mandarle a la vista de objetivos.html el contenido necesario para realizar diversas acciones sobre una serie de objetivos recomendados y personalizados. Por ejemplo, se pasan la lista de objetivos personalizados, los objetivos diarios recomendados de agua, ejercicio, descanso y estado de ánimo o la lista de objetivos por mes. Finalmente, la función redirige a la vista de objetivos.html donde el participante puede visualizar toda la información correspondiente a los mismos.

2. Ver tus objetivos cumplidos siendo participante

Para ver cuáles de los objetivos diarios recomendados han sido cumplidos o aún están pendientes de un participante, es necesario que éste acceda a la pantalla principal de la aplicación. El acceso se hace a través de la función index implementada en el HomeController.java. En ella, tras comprobar que el rol del usuario es de tipo participante, se envían distintos elementos a la vista a través del uso de Model. De esta manera, para llevar la cuenta de objetivos cumplidos, en la función se crea una variable contador de objetivos de tipo integer. Además, puesto que no solo se indica cuántos objetivos ha cumplido, sino también una barra de progreso con el porcentaje que lleva en un día concreto, se utiliza una variable llamada porcentajeObjetivos en la que se hace el cálculo. Por otro lado, para indicar qué objetivos son exactamente los que se han cumplido y los que no, se pasan a la vista las variables aguaCompletado, ejercicioCompletado, descansoCompletado y estadoAnimoCompletado. El valor de dichas variables se actualiza en la función dependiendo de si el estado de dichos objetivos se considera completado.

El objetivo de agua recomendado implica beber 2 litros de agua diarios, o lo que es lo mismo, 8 vasos de 0,25 litros cada uno. De este modo, en la función index en caso de que el participante no haya introducido ningún vaso o haya introducido menos de 8 (ha bebido menos de 2 litros), el valor de aguaCompletado es false. En caso contrario, el objetivo está completado y se suma 1 al contador de objetivos y se pasa la variable aguaCompletado como true.

El objetivo de ejercicio recomendado consiste en registrar al menos la realización de un ejercicio al día. De este modo, en la función index se comprueba si la listaEjercicioParticipante se encuentra o no vacía. Si está vacía se devuelve ejercicioCompletado como false y en caso contrario se suma 1 al contador de objetivos y se pasa la variable ejercicioCompletado como true.

En cuanto al objetivo de descanso recomendado, muestra que lo ideal es dormir 8 horas diarias. De este modo, en la función index se comparan los valores actuales con el ideal y en función de si se ha cumplido o no dicho objetivo se modifica el valor del contador y de la variable descansoCompletado.

Por su parte, el objetivo del estado de ánimo recomendado implica registrar cómo se siente el participante cada día. De este modo, se ve si se ha registrado o no para actualizar el valor del contador de objetivos y de la variable estadoAnimoCompletado.

Finalmente la función index redirige al participante a la vista principal que se corresponde con el script index.html al que se le pasan todos los elementos mencionados, entre otros. Así, es en la vista html donde se regula qué mensajes se muestran al participante en función de los objetivos que éste ha cumplido. Además, acompañando a estos mensajes se muestra una barra de progreso en la parte superior que se crea y actualiza en el html mediante los valores de porcentajeObjetivos y contadorObjetivos recibidos.

3. Acceder al objetivo de agua recomendado siendo participante

Para implementar este caso de uso es necesario seguir lo indicado en el caso de uso 1 de este apartado para acceder a la vista objetivos.html.

En el caso concreto del objetivo recomendado de agua, a través del model, se pasa el objetivoAgua del participante del controlador a la vista. De esta manera, dentro de la vista objetivos.html, en la zona superior que es donde se encuentra la información sobre los 4 objetivos recomendados, se crea un contenedor de botones correspondiente a cada uno de ellos, de forma que con javascript se muestra la información relativa al objetivo del botón pulsado. Este javascript se encuentra en el archivo objetivos.js. Respecto a la información del objetivo de agua se muestran 2 zonas, una superior con el encabezado y los vasos y una inferior con un resumen del consumo diario además de los botones para añadir o eliminar un vaso. Esta implementación se detalla más adelante.

4. Acceder al objetivo de ejercicio recomendado siendo participante

Para implementar este caso de uso es necesario seguir lo indicado en el caso de uso 1 de este apartado para acceder a la vista objetivos.html.

En el caso concreto del objetivo recomendado de ejercicio, a través del model se pasa la listaObjetivoEjercicio del participante. De esta manera, dentro de la vista objetivos.html, en la parte del contenedor de botones, es necesario pinchar sobre el botón Ejercicio para acceder a la información correspondiente, proceso implementado con javascript. El funcionamiento de este punto se detalla más adelante, pero de forma general, se compone de dos partes, un formulario para registrar ejercicio y una tabla donde visualizar y eliminar el ejercicio registrado.

5. Acceder al objetivo de descanso recomendado siendo participante

Para implementar este caso de uso es necesario seguir lo indicado en el caso de uso 1 de este apartado para acceder a la vista objetivos.html.

En el caso concreto del objetivo recomendado de descanso, a través del model se pasa el objetivoDescanso del participante. De esta manera, dentro de la vista objetivos.html, en la parte del contenedor de botones, es necesario pinchar sobre el botón Descanso para acceder a la información correspondiente, proceso implementado con javascript. El funcionamiento de este punto se detalla más adelante, pero de forma resumida contiene un formulario donde se registran las horas y minutos dormidos por el participante.

6. Acceder al objetivo de estado de ánimo siendo participante

Para implementar este caso de uso es necesario seguir lo indicado en el caso de uso 1 de este apartado para acceder a la vista objetivos.html.

En el caso concreto del objetivo de destado de ánimo, a través del model se pasa el objetivoEstadoAnimo del participante. De esta manera, dentro de la vista objetivos.html, en la parte del contenedor de botones, es necesario pinchar sobre el botón Test anímico para acceder a la información correspondiente, proceso implementado con javascript. El funcionamiento de este punto se detalla más adelante aunque consiste en un formulario donde se selecciona cómo se siente el participante.

7. Crear un objetivo personalizado siendo participante

Para crear un objetivo personalizado, el primer paso es acceder al script objetivos.html tal y como se comenta en el caso de uso 1 de este apartado. Así, en la vista de objetivos, en la zona inferior de objetivos personalizados se ha incluido el botón Crear un objetivo nuevo que al hacer click en él, llama a una función de javascript que permite mostrar un modal donde el usuario puede crear el objetivo. Este modal, implementado en el mismo html, contiene un formulario con distintos campos (título, fecha tope y repetición) acerca del nuevo objetivo que el participante debe llenar. Para guardar el nuevo objetivo en la base de datos, el formulario llama a la función guardarObjetivo implementada en ParticipanteController.java. Finalmente, dicha función redirige al participante a la vista de objetivos.html en la que ya se encuentra, de forma que si el objetivo se encuentra en el mes indicado por el calendario, aparece reflejado en la tabla inferior. Para poder mostrar el contenido de esta tabla de manera dinámica y sincronizada con el calendario superior, se ha renderizado el html correspondiente en javascript, concretamente en el archivo objetivos.js.

8. Editar objetivo personalizado siendo participante

La implementación de este caso de uso parte del acceso a la vista de objetivos explicada en el punto 1 de este apartado. De este modo, en la parte más baja de la vista se muestra una tabla en la que se muestran los objetivos personalizados ya creados por el participante. Para poder editarlos, dentro de la tabla se incorpora un campo de acciones que contiene los iconos de editar y eliminar. Al hacer click sobre el de editar, se llama a la función editarObjetivo del script de ParticipanteController.java a la que se le pasa como parámetro el id del objetivo correspondiente. En esa función, tras varias comprobaciones como que el usuario sea un participante, el objetivo que quiere editar sea

distinto de null, es decir, que exista, y que el objetivo que trata de editar sea un objetivo que él mismo haya creado, se redirige a una nueva vista para editar el objetivo a la que se le pasan el usuario y el objetivo. De este modo, en la vista para editar el objetivo (editarobjetivo.html) se muestra un formulario con los campos ya rellenados, para que el participante pueda modificar lo que desee. Al hacer el submit, se llama a la función processEditarObjetivo implementada en ParticipanteController.java que asegura el rol del participante y actualiza el objetivo en la base de datos. Finalmente, redirige a la vista de objetivos.

9. Eliminar objetivo personalizado siendo participante

El proceso de eliminar un objetivo personalizado tiene una parte común con el de editar en lo que respecta a la forma de acceder al botón de eliminar. De este modo, una vez que se hace click en el ícono para eliminar un objetivo, implementado en el script objetivos.js, se muestra una alerta utilizando Swal.fire con javascript que solicita confirmar que se desea eliminar el objetivo para continuar con la operación. Así, si en la alerta el usuario pulsa en Eliminar, se llama a la función eliminarObjetivo del script ParticipanteController.java que, tras comprobar que el usuario es un participante, y que coincide con el del creador del objetivo, mediante servicios se acaba llamando a la función delete que elimina dicho objetivo de la base de datos. Finalmente, se redirige a la vista de objetivos actualizada.

10. Filtrar objetivos personalizados siendo participante

Para implementar este caso de uso, en la parte final del archivo objetivos.js se incluye un div con id filtro cuya implementación se realiza en el script objetivos.js en la función renderDatos. En esta función, primero se comprueba que haya datos (objetivos) que mostrar, pues en caso contrario no se visualiza el filtro. Si hay objetivos, primero se renderiza el html que muestra el filtro, de manera que permite seleccionar entre mostrar todos o solo los pendientes o completados, asignando lo escogido al id estadoObjetivo. A continuación, se agrega un event listener al elemento con dicho id para ejecutar una función que va a mostrar los objetivos correspondientes en cada caso. En ella, se realiza una solicitud fetch para buscar los objetivos del mes que se encuentre abierto en el calendario mediante una url a la que se pasan las variables mesUsado y anioUsado. Después, se utiliza la respuesta del fetch para filtrar los datos recibidos. De esta manera, si el valor es TODOS se devuelven todos los objetivos, pero en caso contrario, sólo se devuelven los objetivos con estadoObjetivo similar al del valor del elemento. Finalmente se llama a la función renderTabla para que se muestren los objetivos filtrados en la tabla del final de la vista.

11. Ver los objetivos de un día siendo participante

Para visualizar los objetivos de un día concreto siendo participante lo primero es haber accedido a la vista de objetivos como se ha señalado en los casos anteriores. Una vez ahí, en la zona de objetivos personalizados se ha implementado un calendario dinámico, de manera que puede navegarse por el mismo y al pulsar sobre un día concreto, a la derecha del calendario se muestran los

objetivos que hay marcados para ese día. Para llevar a cabo esta implementación en primer lugar, se ha pintado el calendario en el objetivos.html utilizando distintas clases a las que se ha aplicado css. A partir de ahí, es en el archivo objetivos.js donde se ha desarrollado la funcionalidad.

En cuanto a la navegación por el calendario, así como la actualización de los objetivos sobre el mismo, se ha realizado la función actualizarCalendario. En ella, se han propuesto varios modos que hacen referencia a si no se ha navegado aún por el calendario o si se ha ido hacia atrás o hacia delante sobre el mismo, de manera que se pueda obtener el número del mes en que se encuentre llamando a la función obtenerNúmeroMes con el mes adecuado. Además, previamente se ha convertido la fecha utilizada de formato javascript a string para poder utilizar de forma separada la parte correspondiente al número del mes. Con esto, se establecen cuáles son el mes y año en que se encuentra situado el calendario. Después, se hace una solicitud fetch para obtener los objetivos que aparecen en ese mes y año concreto a través de una url a la que se le pasan esas variables. Luego se utiliza la respuesta del fetch (que se pasa a formato json) para guardar los datos en la variable objetivos. También se calcula el número de días que se tienen que agregar (y se agregan) al principio del mes para que el primer día sea el lunes. Tanto estos días vacíos como los del mes se agregan al contenedor de días (díasContainer) mediante el uso de varios for. Para terminar, se hace otro bucle for que itera sobre cada celda del calendario estableciendo la variable clickable a false. Además, se obtiene el número del día de la celda actual. También, para diferenciar las celdas que tienen asociados objetivos, se llama a la función diaTieneObjetivos (implementada en el mismo archivo de javascript) que si es true, agrega la clase objetivo a dicha celda para darle un css distinto. Luego se comprueba si la celda marcada se corresponde con el día actual, y en caso de que así sea se le añade la clase today y se llama a seleccionarObjetivosDelDía para ver cuáles son los objetivos marcados para ese día. En caso contrario, se elimina la clase today de la celda. Después se añade un event listener a la celda para que al hacer click se llame a la función seleccionarObjetivosDelDia. Finalmente, se actualiza el encabezado del calendario (nombre del mes y año adecuados) y se llama a la función renderDatos, que, como se ha comentado, es la que muestra el filtro y tabla de los objetivos personalizados.

Respecto a la función seleccionarObjetivosDelDía, también implementada en el mismo javascript, es la que va a permitir que al pulsar sobre un día concreto se muestren a la derecha del calendario los objetivos asociados. En ella, se utiliza una variable auxiliar objetivosAux donde, tras recorrer la lista de objetivos, se van guardando aquellos que se deben devolver en función de una serie de comprobaciones en las que además se diferencia por tipo de repetición de cada objetivo (diariamente, semanalmente, mensualmente, anualmente o ninguna). De esta manera, antes de terminar la función se renderizan los objetivos a la derecha. Para ello, en caso de que no haya objetivos que mostrar en el día, se muestra un mensaje que lo indica, y en caso de que sí los haya, se recorre la lista y se muestra el título del objetivo.

12. Añadir un vaso de agua consumido al objetivo diario de agua siendo participante

Para implementar este caso de uso primero debe realizarse lo indicado en el caso de uso número 3 de este apartado. Así, una vez accedida a la sección del objetivo recomendado de agua, en el objetivos.html se ha implementado un botón Añadir un vaso que llama a la función agregarVaso implementada en el ParticipanteController.java. En esta función, a la que se pasa el id del participante, se hacen distintas comprobaciones. Entre ellas, en caso de que el objetivo sea null, es decir, aún no se haya añadido ningún vaso, se crea el objetivo estableciendo los parámetros iniciales como la fecha, el estado como pendiente o el entero vasos inicializado a 1, entre otros. En caso de que el objetivo ya se haya creado, se compara el número de vasos bebidos con la cantidad objetivo (8 vasos = 2 litros), de manera que si es menor, se suma un vaso. Después, si el número de vasos es igual a la cantidad objetivo, se establece el estado del objetivo como cumplido. Finalmente, mediante servicios y la llamada a la función guardarAgua, se modifica la base de datos y después se redirige a la vista de objetivos. Mientras se van añadiendo estos vasos, mediante javascript se cambia el color de los mismos, pasando a marcar en azul los consumidos y manteniendo en negro los que aún falta beber para completar los 2 litros. Además, junto a los botones, en el html se hacen diversas comparaciones para indicar al usuario si aún no ha bebido ningún vaso, la cantidad (tanto en litros como en vasos) en caso de que ya haya registrado algún vaso y el estado como cumplido al alcanzar los 8 vasos.

13. Eliminar un vaso de agua consumido del objetivo diario de agua siendo participante

A la hora de eliminar un vaso de agua, se realiza lo mismo que en el caso anterior pero de manera inversa. De este modo, al clickar sobre el botón Eliminar vaso se llama a la función quitarVaso de ParticipanteController.java donde en caso de que el objetivo exista y el número de vasos bebido sea mayor a 0 se resta 1 al número de vasos. Además, se comprueba si el número de vasos es igual al objetivo para actualizar el estado a completado o pendiente según corresponda. Finalmente se llama a la función guardarAgua ya comentada. Cabe destacar, que se regula que no puedan consumirse menos de 0 vasos. Al igual que en el caso de añadir un vaso de agua, se muestra la cantidad de agua bebida y demás elementos ya explicados mediante el html.

14. Añadir un ejercicio realizado al objetivo diario de ejercicio siendo participante

Para añadir ejercicio a la lista de ejercicio del participante de un día concreto es necesario acceder primero a la zona correspondiente a este objetivo como se refleja en el caso de uso 4. En esta sección, lo que se muestra es un formulario implementado en el script objetivos.html que permite seleccionar el tipo de ejercicio así como la duración del mismo. Al hacer el submit del formulario se llama a la función agregarEjercicio implementada en ParticipanteController.java. En ella, en caso de que el ejercicio sea distinto de ninguno, se crea

el objetivo de ejercicio correspondiente al que se le añaden los campos adecuados como la fecha. Finalmente, se llama a la función guardarEjercicio que actualiza la base de datos y se redirige a la vista de objetivos. Además, en esa vista, en la parte derecha se recorre la listaEjercicioParticipante para mostrar el ejercicio que ya ha sido añadido por el participante. En caso de que la lista esté vacía, se muestra un mensaje informando al participante.

15. Eliminar un ejercicio realizado del objetivo diario de ejercicio siendo participante

Para eliminar un ejercicio añadido mediante el caso de uso anterior, es necesario acceder a la zona del objetivo de ejercicio como se detalla en el caso de uso 4. Así, desde esta parte de la vista, concretamente en la parte derecha donde se encuentra la tabla con el listado de ejercicio realizado por el participante, se ha añadido un ícono de eliminar asociado a cada elemento de la lista. De este modo, al pulsar sobre el ícono de eliminar se llama a la función eliminarEjercicio implementada en el ParticipanteController.java donde, tras comprobar que el usuario tiene rol de participante, y el objetivo que se desea eliminar es suyo, se actualiza la base de datos mediante servicios iniciando con la llamada a la función borrarObjetivoEjercicio. Finalmente, se redirige a la vista de objetivos actualizada.

16. Registrar el tiempo de sueño en el objetivo diario de descanso siendo participante

El registro del tiempo descansado por el participante se lleva a cabo desde la vista objetivos.html en la zona de objetivo de descanso cuya acceso se ha implementado como se detalla en el caso de uso 5 de este apartado. En concreto, se ha implementado un formulario que solicita al participante introducir el número de horas y minutos que ha descansado. El submit de dicho formulario lleva a la función agregarDescanso implementada en el ParticipanteController.java donde tras diversas comprobaciones se crea el objetivo estableciendo los atributos adecuados y actualizando la base de datos mediante la llamada a guardarDescanso. Finalmente, se redirige a la vista de objetivos actualizada donde además, en el html se incluyen comprobaciones que informan al usuario del proceso del registro de sueño, indicando lo que ha dormido, si ha cumplido el objetivo o si aún no ha registrado el tiempo de sueño.

17. Eliminar el registro de sueño del objetivo diario de descanso siendo participante

Para eliminar un registro de sueño, se ha implementado en la zona de dicho objetivo un botón Eliminar registro que se muestra en la vista una vez ya se ha guardado un tiempo de descanso para el día concreto, pues en función de eso, se mostrará el formulario para añadir el registro o la posibilidad de eliminarlo. De este modo, si el participante pulsa sobre Eliminar registro se llama a la función eliminar descanso que borra dicho objetivo de la base de datos y finalmente redirige a la vista de objetivos.

18. Registrar el estado de ánimo en el objetivo de estado de ánimo siendo participante

Desde el script objetivos.html en la zona de estado de ánimo a la que se accede como se indica en el caso de uso 6, se ha implementado un formulario que permite al usuario seleccionar el estado de ánimo que describe cómo se ha sentido en un día concreto representado mediante una serie de iconos de caras (feliz, triste, enfadada, etc) Así, al pulsar Guardar, se llama a la función agregarEstadoAnimo de ParticipanteController.java que tras las comprobaciones correspondientes termina actualizando la base de datos y redirigiendo al participante a la vista de objetivos. Además, la zona de este objetivo se modifica, mostrando a través del html el estado de ánimo que ha registrado el participante.

19. Eliminar el registro del estado de ánimo del objetivo de estado de ánimo siendo participante

En el caso de querer eliminar el registro de sueño, se ha implementado un botón en la zona de este objetivo llamado Eliminar registro que lo que hace es llamar a la función eliminarEstadoAnimo del participanteController.java la cual actualiza la base de datos borrando dicho objetivo tras la llamada a la función borrarObjetivoEstadoAnimo y redirige a la página de objetivos.html. Así, una vez borrado el objetivo, el usuario puede volver a registrar cómo se siente repitiendo el proceso señalado en el caso anterior.

7.2.8. Cuaderno y sesiones

1. Acceder al cuaderno siendo Participante

Dentro de la clase de ParticipanteController se encuentra la función cuaderno(), este lleva al script cuaderno.html que tiene un botón llamado Información. Cuando se pulsa lleva a la función de ParticipanteController llamada info(). Esta devuelve el script info.html donde se ha añadido un iframe que muestra un pdf con las diapositivas.

2. Acceder a las fichas siendo Participante

Dentro de la clase de ParticipanteController se encuentra la función cuaderno(), este lleva al script cuaderno.html que tiene un botón llamado Fichas. Cuando se pulsa lleva a la función de ParticipanteController llamada fichas(). Esta devuelve el script fichas.html donde hay varios botones para acceder a las distintas fichas.

3. Rellenar la ficha taller siendo Participante

Desde el script fichaTaller.html el usuario puede llenar un formulario. Dentro del mismo hay dos atributos que se van actualizando con javascript, concretamente la barra de importancia y de capacidad. Cuando el usuario pulsa en Guardar se llama a la función processtaller() de ParticipanteController y se actualiza la ficha en la base de datos.

4. Rellenar la ficha objetivo siendo Participante

Cuando el participante se encuentra en el script fichas.html puede pulsar en el botón de Peso para llenar la ficha objetivo. Se redirige a la función fichaObjetivo() de ParticipanteController.java y lo primero que hace es comprobar si es el usuario es un participante y está dado de alta. Después se obtienen del formulario de exploración, los datos necesarios para realizar esta ficha.

Además, se realiza un cálculo para saber cuál es el peso saludable para ese participante para conseguir un IMC de 25.

Por último se calcula el porcentaje 5 y 10 del peso para pasarlo a través del modelo al script fichaObjetivo.html, donde se ha implementado un formulario para que el participante rellene cuánto quiere pesar y cuántos kilos quiere perder.

Cuando el participante pulse en el botón de Guardar se redirige a la función processFichaObjetivo() de ParticipanteController.java, donde se realiza la actualización de los datos de la ficha.

5. Acceder a las fichas de elección siendo Participante

Desde el script fichas.html el participante puede pulsar en el botón de Decisiones. En ese momento, se dirige a la función fichaEleccion() de ParticipanteController. Esta comprueba que la persona que esté accediendo sea un participante de ALTA y devuelve el script de fichaEleccion.html.

6. Rellenar una ficha de elección en concreto siendo Participante

Desde el script de fichaEleccion.html el participante puede pulsar en el número de la ficha que desee y este le dirigirá al script de fichaEleccionConcreta.html mediante la función llamada decisiones() de ParticipanteController. Dentro de fichaEleccionConcreta.html hay un formulario a llenar por el usuario y dos imágenes que hemos añadido usando iframes. Además, dentro del html se hace uso del th:if y th:unless de thymeleaf para poder mostrar la ficha indicada. Cuando el usuario pulsa en Guardar le lleva a una función de ParticipanteController llamada processdecisiones() que actualiza la ficha en la base de datos.

7. Acceder a las sesiones del programa siendo Participante

Dentro de la clase ParticipanteController hemos implementado la función de sesiones(), esta controla que solamente puedan acceder los participantes que estén de ALTA y devuelve el script de sesiones.html. Dentro del script hay varios botones para elegir el número de sesión indicada, una vez se pulsa, se dirige a la función de ParticipanteController llamada sesion1() que controla que solo puedan acceder participantes que estén de ALTA, luego busca en la base de datos la sesión con el número que ha indicado el usuario y la añade al modelo. Por último, devuelve el script de sesion.html.

8. Rellenar una sesión en concreto siendo Participante

Dentro del script sesion.html hay un formulario en el que el participante puede llenar su asistencia y tomar anotaciones del transcurso de su sesión. Una vez el usuario pulse en Guardar se le dirigirá a la función de ParticipanteController llamada actualizar(). Dentro de esta función, se actualizan los datos de la sesión en la base de datos y se actualiza la asistencia del participante en la base de datos.

9. Terminar una sesión en concreto siendo Participante

Dentro del script sesion.html hay un botón de terminar sesión, cuando se pulsa se abre un formulario en el que el participante puede llenar el peso que ha perdido y los centímetros de cintura que ha perdido. Una vez el usuario pulse en Terminar Sesión se le dirigirá a la función de ParticipanteController llamada actualizarTerm(). Dentro de esta función, se actualizan los datos de la sesión en la base de datos y se actualiza el número de sesiones completas del participante en la base de datos.

10. Hacer el cuestionario de una sesión en concreto siendo Participante

Dentro del script sesion.html hay un enlace para realizar el cuestionario, cuando se pulsa en él, se abre una nueva pestaña con el cuestionario de Google de esa sesión.

11. Descargar las diapositivas de una sesión en concreto siendo Participante

Dentro del script sesion.html hay un enlace para descargar las diapositivas, cuando se pulsa en él, se dirige a la función descargar() de ParticipanteController, esta utiliza la clase 'Path' para obtener la ruta del archivo en el sistema de archivos y la clase 'Files' para leer todos los bytes del archivo. A continuación, establece los encabezados HTTP apropiados, incluyendo el tipo de contenido y la longitud del contenido, y utiliza la clase 'ContentDisposition' para establecer el nombre de archivo original en el encabezado Content-Disposition. Finalmente, la función devuelve una respuesta HTTP con el cuerpo del archivo como un array de bytes, junto con los encabezados establecidos anteriormente y se descarga el archivo. Para implementar la función hemos usado varias clases de la biblioteca org.springframework.http.

12. Acceder a las sesiones de un participante en concreto siendo Coordinador

Dentro del script expediente.html el coordinador puede pulsar en el botón de Fichas, una vez ahí se abre un modal con varios botones para ver el número de sesión deseado. Cuando aprieta a uno de esos botones se dirige a la función session1() de CoordinadorController que devuelve el script de sesionPart.html.

13. Ver los datos rellenados por el participante de una sesión en concreto siendo Coordinador

Dentro del script sesionPart.html el coordinador puede ver los datos rellenados por el participante de su asistencia y sus notas. Si aprieta en Peso y cm perdidos

se abre un modal con los datos rellenados por el participante de su peso y sus centímetros de cintura.

7.2.9. Gestión de participantes

1. Recibir notificaciones siendo Participante

El participante recibe notificaciones en diferentes situaciones, como al ser dado de alta, agregado a un grupo, recibir una invitación a una actividad, recibir un material y cuando se elimina una invitación. Estas notificaciones le mantienen informado sobre eventos importantes en la aplicación.

El procedimiento a seguir en cada uno de los casos es el mismo. Para crear una notificación se realiza una instancia a la clase Notificacion a la que se le pasa como parámetros el participante al que se le dirige la notificación, el mensaje de la notificación y la fecha y hora actual en la que se crea la notificación.

Por otro lado, para recibir las notificaciones y que le aparezcan al participante en la aplicación, en todas las funciones que le lleven a una vista se realiza una consulta a la base de datos para obtener el listado de todas las notificaciones.

2. Rellenar el test de Findrisc antes de registrarse

Dentro del script findrisc.html hay un formulario que el usuario puede llenar antes de registrarse, en él se usa javascript para calcular el imc y el resultado final entre otras cosas. Cuando se calcula el resultado, en función de la puntuación aparecen distintas opciones. Esto se ha implementado usando javascript.

3. Ver las recomendaciones de hábitos saludables

Dentro del script findrisc.html, si la puntuación obtenida es baja, se devuelve un enlace con recomendaciones de hábitos saludables. Cuando pincha en el enlace se devuelve el script recomendaciones.html.

4. Acceder al listado de participantes de Baja o del coordinador siendo Coordinador

En CoordinadorController hay una función llamada participantes(). En esta función, en primer lugar, se controla que solamente pueda acceder un coordinador. Luego, se crean distintas páginas con la lista de participantes y se añade al modelo. Esto se hace usando la clase Page de la biblioteca org.springframework.data.domain. Finalmente se devuelve el script de participantes.html.

5. Filtrar los participantes siendo Coordinador

Dentro del script participantes.html hemos implementado un filtro para ayudar al coordinador a encontrar al participante que busca. Para ello hemos usado javascript y jquery. Dentro del script de tablaparticipantes.js si el coordinador selecciona el estado de BAJA se muestran solo los participantes de BAJA, e igual con el resto de opciones. También compara el texto introducido por el coordinador con el texto que hay en la clase de html llamada buscarParticipantes, con ello devuelve los resultados que coincidan.

6. Imprimir el listado de participante siendo Coordinador

Dentro del script participantes.html hay un botón Imprimir, cuando el coordinador pulsa en él se dirige a una función de javascript que abre una interfaz para imprimir la pantalla actual.

7. Acceder al expediente de un participante en concreto siendo Coordinador

Dentro de la clase ParticipanteController hemos implementado la función de expediente(). En ella, en primer lugar se controla que solo pueda acceder el coordinador de ese participante, o si el participante está de BAJA, cualquier coordinador. Luego devuelve el script de expediente.html.

8. Mandar una analítica a un participante en concreto siendo Coordinador

Dentro del script expediente.html hay un botón para mandar una analítica, cuando el coordinador pulsa en él lleva a la función analitica() de ParticipanteController. En esta función, se comprueba que el que haya accedido sea un coordinador y se llama a la función de mandarAnalitica() de UsuarioService. En esta función se usa Gmail API para poder enviar un correo electrónico desde la aplicación. En primer lugar, se crean los credenciales con la cuenta de gmail de Estepper. Luego se añade el mensaje en sí y el asunto del correo electrónico, se busca el correo del participante destinatario y se llama a la función de mandarcorreo(). Dentro de esta, se crea un servicio de Gmail y un mensaje con MimeMessage, luego se trabaja con el mensaje para pasarlo a la clase Message y se envía usando service. Para ello hemos usado las bibliotecas de javax.mail, com.google.api.client y com.google.api.services.gmail.

9. Acceder a la fase de valoración de un participante siendo Coordinador

Dentro del script expediente.html hay un botón llamado Fase de Valoración. Cuando se pulsa en él se llama a la función fasedevaloracion() de ParticipanteController. En esta función se controla que solamente pueda acceder un coordinador, se buscan los formularios de la fase de valoración de ese participante y se añaden al modelo. Luego se devuelve el script valoración.html.

10. Dar de alta a un participante siendo Coordinador

Dentro del script valoracion.html hay un botón para dar de alta la cuenta de un participante. Cuando el coordinador pulsa este botón, se comprueba que cumpla las condiciones para poder dar de alta al participante, para ello hemos usado javascript. Si no se cumplen, se abre un modal que indica el problema por el cual no se cumplen las condiciones. Si se cumplen, se dirige a la función processActCuenta() de ParticipanteController. En ella, en primer lugar, se controla que solo pueda acceder un coordinador. Luego se recogen los datos de los formularios de findrisc y de exploracion del participante y se llama a la función de activarcuenta() de FaseValoracionService en el que se actualizan los datos del participante en la base de datos con los datos existentes en los

formularios de findrisc y de exploración, por ejemplo su género o su edad. También se actualiza el estado de la cuenta del participante a ALTA. Luego, se crean las sesiones correspondientes del participante si no existen aún y las fichas.

11. Eliminar la cuenta de un participante siendo Coordinador

Dentro del script de valoracion.html hay un botón para eliminar la cuenta. Cuando el coordinador pulsa en él se abre una alerta usando javascript y sweetalert y si el coordinador confirma que quiere eliminar esa cuenta porque considera que el participante no debe entrar en el programa, se dirige a la función processElimCuenta() de ParticipanteController. En ella, se controla que esté accediendo la persona adecuada y se elimina de la base de datos todos los datos relacionados con el usuario y su cuenta en sí.

12. Imprimir el expediente de un participante siendo Coordinador

Dentro del script expediente.html hay un botón Imprimir, cuando el coordinador pulsa en él se dirige a una función de javascript que abre una interfaz para imprimir la pantalla actual.

13. Rellenar formulario de Exploración de un participante siendo Coordinador

Dentro del script exploracion.html hay un formulario que el coordinador rellena. Dentro de este formulario está el dato del IMC que se calcula automáticamente usando javascript. Si desea guardar los datos pulsa en Guardar. Cuando pulsa se muestra una alerta usando javascript para confirmar que quiera cambiar los datos de ese formulario. Si confirma la acción se dirige a processExploracion() de ParticipanteController. En esta función se controla que esté accediendo un coordinador y se actualizan los datos del formulario de Exploración. Además, se actualizan los datos del formulario de Findrisc dado que usa datos del de Exploracion.

14. Imprimir formulario de Exploración de un participante siendo Coordinador

Dentro del script exploracion.html hay un botón Imprimir, cuando el coordinador pulsa en él se dirige a una función de javascript que abre una interfaz para imprimir la pantalla actual.

15. Rellenar formulario de Findrisc de un participante siendo Coordinador

Dentro del script findrisc.html hay un formulario que el coordinador rellena. Dentro de este formulario están los datos de puntuación y de escala, que se calculan automáticamente usando javascript. Además, hay datos que se calculan con anterioridad desde ParticipanteController ya que vienen dados por las respuestas del formulario de Exploración. Si desea guardar los datos pulsa en Guardar. Cuando pulsa se muestra una alerta usando javascript para confirmar que quiera cambiar los datos de ese formulario. Si confirma la acción se dirige a processFindrisc() de ParticipanteController. En esta función

se controla que esté accediendo un coordinador y se actualizan los datos del formulario de Findrisc. Además, si se cumple una serie de condiciones para que el participante entre en el programa, como tener más de treinta y cinco años, se crean los demás formularios de la fase de valoración.

16. Imprimir formulario de Findrisc de un participante siendo Coordinador

Dentro del script findriscPart.html hay un botón Imprimir, cuando el coordinador pulsa en él se dirige a una función de javascript que abre una interfaz para imprimir la pantalla actual.

17. Rellenar formulario de Antecedentes de un participante siendo Coordinador

Dentro del script antecedentes.html hay un formulario que el coordinador rellena. Si desea guardar los datos pulsa en Guardar. Cuando pulsa se muestra una alerta usando javascript para confirmar que quiera cambiar los datos de ese formulario. Si confirma la acción se dirige a processantecedentes() de ParticipanteController. En esta función se controla que esté accediendo un coordinador y se actualizan los datos del formulario de Antecedentes.

18. Imprimir formulario de Antecedentes de un participante siendo Coordinador

Dentro del script antecedentes.html hay un botón Imprimir, cuando el coordinador pulsa en él se dirige a una función de javascript que abre una interfaz para imprimir la pantalla actual.

19. Rellenar formulario de Adherencia a la dieta mediterránea de un participante siendo Coordinador

Dentro del script alimentacionval.html hay un formulario que el coordinador rellena. Dentro de este formulario hay varios datos que se calculan automáticamente usando javascript. Si desea guardar los datos pulsa en Guardar. Cuando pulsa se muestra una alerta usando javascript para confirmar que quiera cambiar los datos de ese formulario. Si confirma la acción se dirige a processalimentacionval() de ParticipanteController. En esta función se controla que esté accediendo un coordinador y se actualizan los datos del formulario de Adherencia a la dieta mediterránea.

20. Imprimir formulario de Adherencia a la dieta mediterránea de un participante siendo Coordinador

Dentro del script alimentacionval.html hay un botón Imprimir, cuando el coordinador pulsa en él se dirige a una función de javascript que abre una interfaz para imprimir la pantalla actual.

21. Rellenar formulario de Actividad Física de un participante siendo Coordinador

Dentro del script actfisica.html hay un formulario que el coordinador rellena. Dentro de este formulario hay varios datos que se calculan automáticamente

usando javascript. Si desea guardar los datos pulsa en Guardar. Cuando pulsa se muestra una alerta usando javascript para confirmar que quiera cambiar los datos de ese formulario. Si confirma la acción se dirige a processactfisica() de ParticipanteController. En esta función se controla que esté accediendo un coordinador y se actualizan los datos del formulario de Actividad Física.

22. Imprimir formulario de Actividad Física de un participante siendo Coordinador

Dentro del script actfisica.html hay un botón Imprimir, cuando el coordinador pulsa en él se dirige a una función de javascript que abre una interfaz para imprimir la pantalla actual.

23. Rellenar formulario de Clasificación de un participante siendo Coordinador

Dentro del script clasificacion.html hay un formulario que el coordinador rellena. Dentro de este formulario está la clasificación, que se calcula automáticamente usando javascript. Además, la clasificación se calcula con datos de otros formularios que se recogen añadiéndolos al modelo desde ParticipanteController. Si desea guardar los datos pulsa en Guardar. Cuando pulsa se muestra una alerta usando javascript para confirmar que quiera cambiar los datos de ese formulario. Si confirma la acción se dirige a processClasificacion() de ParticipanteController. En esta función se controla que esté accediendo un coordinador y se actualizan los datos del formulario de Clasificación.

24. Imprimir formulario de Clasificación de un participante siendo Coordinador

Dentro del script clasificacion.html hay un botón Imprimir, cuando el coordinador pulsa en él se dirige a una función de javascript que abre una interfaz para imprimir la pantalla actual.

7.2.10. Gestión de grupos

1. Acceder a mi grupo siendo participante

Para llevar a cabo la implementación de este caso de uso, en el script GruposController.java se ha realizado la función chat(Model model). En esta función, después de comprobar que el rol del usuario es de tipo participante, se hacen consultas a la base de datos mediante servicios. Así, se pasan a la vista chat.html variables como la lista de participantes del grupo, los mensajes intercambiados que se muestran en esa vista ya comentados en casos de uso anteriores o el grupo en sí. De este modo, además de la zona correspondiente a los chats ya explicada, dentro de esta vista se encuentran otras dos secciones. En primer lugar, una que contiene a los participantes del grupo, de manera que en el html se recorre la lista listadoParticipantesGrupo mostrando sus avatares dentro del espacio correspondiente. Por otro lado, está una zona de progreso grupal en el que se refleja la pérdida de peso, asistencia y pérdida de cm de

cintura medios del grupo del participante. Para realizar estos cálculos se ha utilizado javascript. En concreto, pueden visualizarse en el archivo grupos.js.

2. Acceder al listado de grupos siendo coordinador

Dentro del script coordinador.html se ha implementado un botón GRUPOS que llama a la función grupos del GruposController.java. En esta función, tras comprobar que el usuario es un coordinador y actualizar aquellos grupos que ya hayan terminado antes de redirigir a la vista grupos.html, se realizan una serie de acciones. Entre ellas, se lleva a cabo la paginación y recuperación del listado de grupos mediante servicios y consultas a la base de datos, que posteriormente se envían a la vista. Así, en la vista se recorre la lista de grupos recibida y se muestra en forma de tabla.

3. Añadir participante a un grupo siendo coordinador

A la hora de añadir un participante a un grupo es necesario partir de la vista principal del coordinador donde se implementa el listado de participantes. En este listado, para cada participante se ha implementado un botón con un ícono que al pulsarlo llama a la función unirAGrupo realizada en el GruposController.java y a la que se pasa como parámetro el id del participante seleccionado. En esta función, después de comprobar que el usuario tiene rol de coordinador, se realizan diversas consultas a la base de datos mediante servicios y se redirige a la vista unirAgrupo.html. Cabe destacar que en el momento de accionar el botón de añadir a un grupo, se hacen comprobaciones de forma que si el usuario ya pertenece a un grupo se muestra un modal implementado en participantes.html que explica por qué no se puede realizar la acción. Una vez que el coordinador se encuentra en la vista unirAGrupo.html, se ha implementado una tabla para seleccionar a qué grupo se desea añadir al participante. Así, una vez el coordinador pulse sobre seleccionar en alguno de los grupos, se llama a la función actualizarGrupos a la que se le pasan tanto el id del participante como del grupo concretos. La función se encuentra implementada en el CoordinadorController.java . En ella, tras asegurar el control de acceso exclusivo del coordinador, se actualiza la información del grupo (un integrante más) y del participante (se le asigna identificador de grupo) en la base de datos. Finalmente se redirige a la vista con la lista de grupos.

4. Crear un nuevo grupo siendo coordinador

Para implementar este caso de uso, una vez accedido al listado de grupos como se indica en el caso 2 de este apartado, se ha creado un botón Crear un grupo nuevo que abre un modal mediante javascript. Dicho modal contiene un formulario con dos campos: uno para darle nombre al grupo y otro para añadir participantes. El segundo campo se ha implementado con un select de opción múltiple que recorre la lista de participantes existentes que aún no tienen asignado ningún grupo. Cuando el coordinador pulsa en el botón Añadir del modal, se llama a la función guardarGrupo implementada en GruposController.java. En esta función, se establecen los distintos elementos asociados al grupo y finalmente se actualiza la base de datos creando un nuevo grupo a

partir de la función save. Entre los valores que se asignan al grupo se encuentra por ejemplo la generación del código como número aleatorio para lo cual se hace uso de RandomStringUtils.randomAlphanumeric y la lista de participantes modificada. Por último, se redirige de nuevo a la vista del listado de grupos.

5. Ver un grupo concreto siendo coordinador

Para ver un grupo concreto, desde la vista del listado de grupos se ha implementado un botón con un icono de un ojo de forma que al hacer click sobre él se llama a la función unGrupo con el id del grupo implementada en el script GruposController.java. Esta función, tras comprobar que el usuario efectivamente es coordinador, obtiene mediante el uso de servicios el contenido de distintas variables como los mensajes intercambiados en el grupo, el listado de participantes o las observaciones realizadas acerca del mismo. Después redirige a la vista unGrupo.html con toda la información pertinente.

6. Crear una nota nueva de observación de un grupo siendo coordinador

Para crear una nueva observación, dentro del script unGrupo.html en la zona inferior se ha dedicado un espacio para las observaciones. De esta manera, aparece un botón Crear una nota nueva que llama a una función javascript que a su vez abre un modal para crear la nota. Este modal se compone de un formulario de un solo input que el coordinador debe llenar. De este modo, al pulsar en Añadir se dirige a la función guardarObservaciones desarrollada en gruposController.java que modifica la base de datos añadiendo la nueva nota a través de servicios y la llamada a la función guardar. Finalmente, redirige de nuevo al coordinador a la vista del grupo en la que se encontraba.

7. Editar una nota concreta de observación de un grupo siendo coordinador

Para editar una nota ya creada, desde el script unGrupo.html en la zona de notas y observaciones se ha implementado un botón asociado a cada nota que permite editarla y a los que, mediante javascript, se les ha agregado un evento. Así, al pulsar en él, se llama a una función que abre un modal, pero a diferencia de lo que sucedía con el caso anterior, el modal se encuentra relleno con el contenido de la nota para que el coordinador pueda modificar lo que desee. Este proceso realizado a través de eventos permite obtener información concreta sobre la nota que se quiere editar, particularmente, lo correspondiente a nota-cargada (el contenido). Esto se hace para que en el modal el campo aparezca con la información anterior. Finalmente, el submit del formulario del modal llama a la función guardarObservaciones2 con el id de la nota donde se actualiza la base de datos.

8. Eliminar una nota concreta de observación de un grupo siendo coordinador

Dentro del script unGrupo.html, al igual que sucede con el botón de editar nota, se ha implementado otro botón para eliminarlas asociado a cada una de las notas. De esta manera, al hacer click en él, mediante javascript se recupera

la información de la nota que se desea eliminar y se muestra una alerta para pedir la confirmación de eliminación. Así, si se pulsa eliminar, se llama a la función eliminarNota implementada en el archivo GruposController.java que, después de realizar el control de acceso, implementa la función borrar que modifica la base de datos eliminando de la misma la nota correspondiente.

9. Editar la información de un grupo concreto siendo coordinador

Para implementar este caso de uso, en el script del listado de grupos se ha añadido un botón con un icono de editar que redirige al coordinador al script de editar un grupo que contiene un formulario mediante la función mostrarFormularioDeEditarGrupo del GruposController.java. En este formulario, que ya se muestra relleno, se han distinguido los campos editables de los que no lo son a través del css. Cuando se envía el formulario se llama a una función que guarda la información del grupo actualizada en la base de datos y finalmente redirige a la vista del listado de grupos. Además, se ha implementado una forma rápida de eliminar a los participantes de modo que se recorre la lista de participantes del grupo y para cada uno se incluye un botón de eliminar que llama a la función correspondiente para borrar a dicho participante de la base de datos.

10. Eliminar un grupo siendo coordinador

En el script del listado de grupos se ha implementado un botón asociado a cada uno de los grupos que permite eliminarlo. Al pinchar sobre este botón, usando javascript se muestra un modal de alerta que solicita al usuario la confirmación para eliminar el grupo. De este modo, una vez que el coordinador lo confirme, se llama a la función eliminarGrupo desarrollada en GruposController.java a la que se le pasa el id del grupo que se quiere eliminar. En ella, tras comprobar el rol del coordinador, mediante servicios y accesos a la base de datos se llama a diversas funciones para eliminar de la base de datos no solo el grupo sino también los materiales enviados al mismo, los mensajes intercambiados y las observaciones realizadas. De hecho, antes de borrar el grupo, a los participantes del mismo, se les modifica el campo del grupo poniendo de nuevo su id como null. Por último se redirige a la vista de la lista de grupos actualizada.

11. Buscar un grupo concreto en la lista de grupos siendo coordinador

En el script grupos.html se ha implementado un buscador que, en tiempo real, devuelve los grupos que coinciden con el texto escrito por el coordinador. Para implementarlo, primero se ha diseñado la zona del input en el html, Después, mediante javascript, en el archivo grupos.js se ha desarrollado la función que utiliza jquery de forma que según se van detectando pulsaciones en el teclado se filtra mostrando u ocultando las filas de la tabla correspondientes.

12. Filtrar grupos activos y terminados en la lista de grupos siendo coordinador

Para filtrar los grupos activos y terminados, dentro del script grupos.html se ha generado el componente para el filtro. Después, el funcionamiento del mismo

se ha desarrollado a través de una función de javascript realizada en el archivo grupos.js. De este modo, se ha añadido un event listener que, en función del valor escogido en el select (identificado por un id), hace que la función muestre u oculte los grupos que corresponda.

7.3. Front-End o lado del cliente

7.3.1. Gestión de usuarios

The screenshot shows the user login interface for the Estepper application. The title 'Estepper' is at the top. Below it are two input fields: 'Código' (with a person icon) and 'Contraseña' (with a lock icon). To the right of the password field is a visibility toggle button. A blue 'Iniciar sesión' button is below the inputs. Below the button is a link: '¿Has olvidado tu código o contraseña? Pincha [aqui](#) para recibirlo en el correo.' At the bottom of the main form is a message: 'Si no sabes si unirte o no a nuestro proyecto, te recomendamos hacer el [TEST DE FINDRISC](#)'.

Figura 7.11: Vista de iniciar sesión siendo Usuario

The screenshot shows the user registration interface for the Estepper application. The title 'Registro' is at the top. It contains four input fields: 'Alias/Nickname' (with a person icon), 'Correo electrónico' (with an envelope icon), 'Contraseña' (with a lock icon), and 'Confirma tu contraseña' (with a lock icon). Below these fields is a checkbox agreement: 'Acepto los [términos y condiciones](#) y autorizo el uso de mis datos personales.' A blue 'Registrarse' button is at the bottom. Below the button is a link: '¿Ya estás registrado? Pincha [aqui](#) para iniciar sesión.'

Figura 7.12: Vista de registrarse siendo Usuario



Figura 7.13: Vista de recuperar datos siendo Usuario

A screenshot of a web page showing a user profile. The sidebar on the left has links for "Inicio", "Cuaderno", "Actividades", "Alimentación", "Materiales", "Progreso", and "Mi grupo". The main area shows a placeholder profile picture, the alias "Almudena: 52654", and contact information: "almunaha@ucm.es", "ALTA", "52", and "FEMENINO". There is also an "Editar perfil" button.

Figura 7.14: Vista de ver perfil siendo Usuario

A screenshot of a web page titled "Listado de usuarios". The sidebar on the left has links for "Usuarios" and "Mensajes". The main area shows a table with columns: Alias, Correo, Código, Rol, Estado, and Acciones. The table contains four rows with data: Patricia (plata@ucm.es, 172, Participante, ALTA), Almudena (almunaha@ucm.es, 111, Participante, ALTA), Inés (ineher02@ucm.es, 444, Participante, ALTA), and Mercedes (proyectoestepper@gmail.com, 222, Coordinador, ALTA). Each row has an "Acciones" column with three icons: a person, an envelope, and a trash can.

Figura 7.15: Vista de acceder al listado siendo Administrador

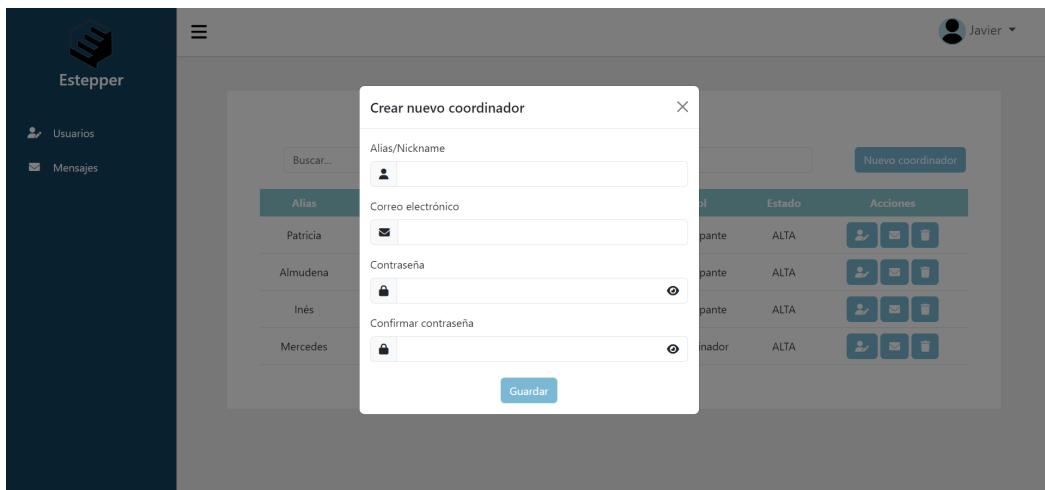


Figura 7.16: Vista de crear coordinador siendo Administrador

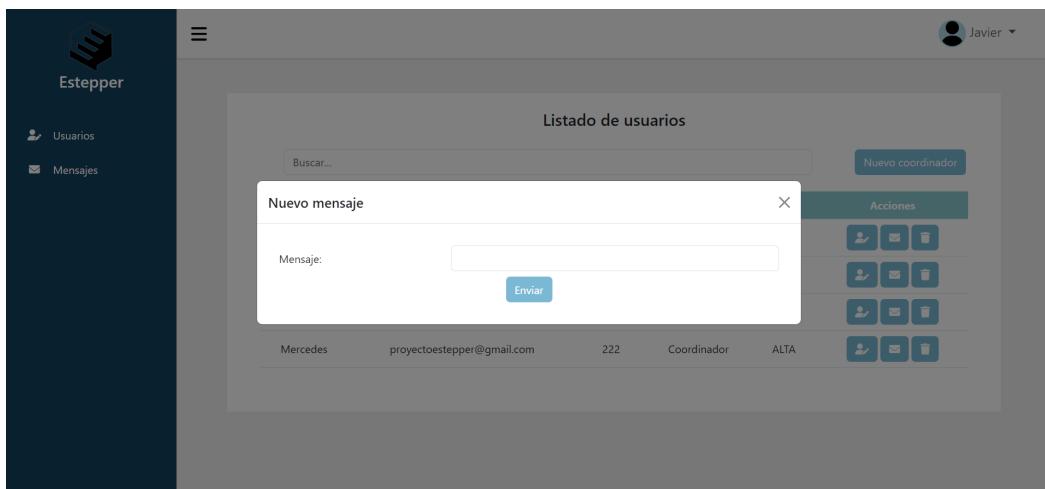


Figura 7.17: Vista de enviar mensaje de incidencia siendo Administrador

7.3.2. Actividades

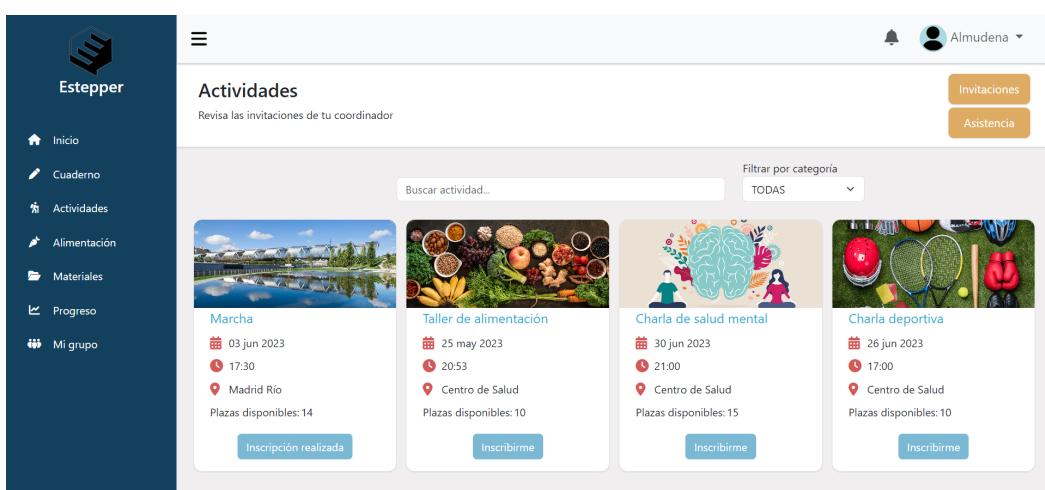


Figura 7.18: Vista de acceder las actividades siendo Participante

The screenshot shows the 'Invitaciones' (Invitations) section of the app. On the left is a dark sidebar with icons for Home, Notebook, Activities, Nutrition, Materials, Progress, and My group. At the top right is a user profile for 'Almudena'. Below the header, the title 'Invitaciones' is displayed with the sub-instruction 'Da una respuesta a tu coordinador'. Two sections are shown: 'Invitaciones pendientes' (Pending Invitations) and 'Invitaciones aceptadas' (Accepted Invitations). In the 'Pending Invitations' section, there are two entries: 'Taller de alimentación' on 25 may 2023 20:53 coordinated by Mercedes with 'Aceptar' (Accept) and 'Rechazar' (Reject) buttons; and 'Charla deportiva' on 26 jun 2023 17:00 coordinated by Mercedes with the same buttons. In the 'Accepted Invitations' section, there is one entry: 'Marcha' on 03 jun 2023 17:30 coordinated by Mercedes.

Figura 7.19: Vista de ver invitaciones siendo Participante

The screenshot shows the details for the 'Taller de alimentación' (Nutrition Workshop) activity. The top navigation bar includes the 'Estepper' logo, a sidebar with activity icons, and a user profile for 'Almudena'. The main content area has tabs for 'Descripción' (Description), 'Detalles' (Details), and 'Inscripción' (Registration). The 'Inscripción' tab is active. It contains instructions to 'Realizar la inscripción' (Register) and a note that participation will be added to the user's list of activities. A 'Confirmar asistencia' (Confirm attendance) button is at the bottom right. A 'Atrás' (Back) button is at the bottom left.

Figura 7.20: Vista de ver actividad y realizar inscripción siendo Participante

The screenshot shows the 'Actividades' (Activities) section. The sidebar includes the 'Estepper' logo and activity icons. At the top right are buttons for 'Invitaciones' (Invitations) and 'Asistencia' (Attendance). A modal window titled 'Asistencia confirmada' (Attendance confirmed) is open, showing a 'Recordatorio de actividades inscritas' (Reminder of registered activities) for a 'Marcha' (March) on 03 jun 17:30 at Madrid Río. The main area lists other activities: 'Marcha' on 03 jun 17:30 at Centro de Salud with 10 available slots, 'Centro de Salud' with 10 available slots, 'Salud mental' with 23 available slots, and 'Charla deportiva' on 26 jun 17:00 at Centro de Salud with 10 available slots. Each activity card has an 'Inscripción realizada' (Registration made) button.

Figura 7.21: Vista de ver inscripciones siendo Participante

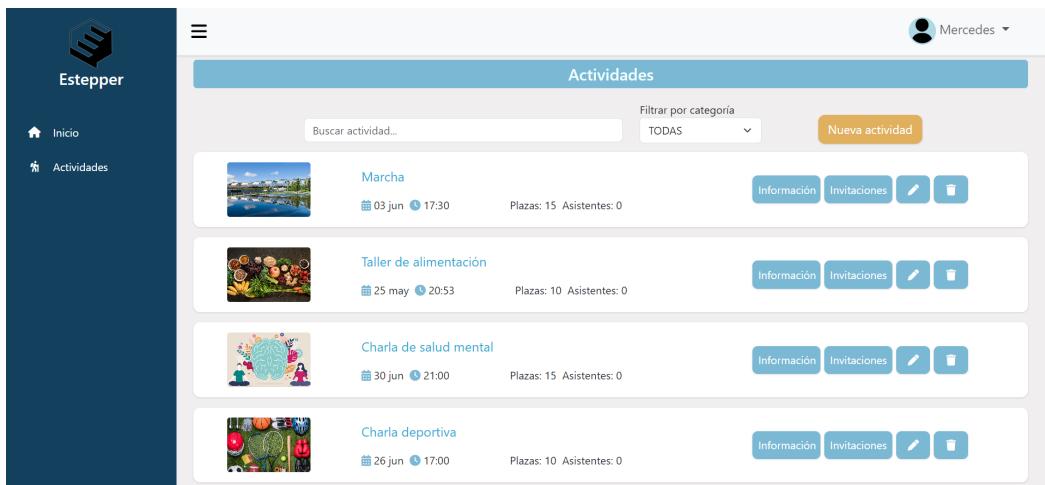


Figura 7.22: Vista de gestionar las actividades siendo Coordinador

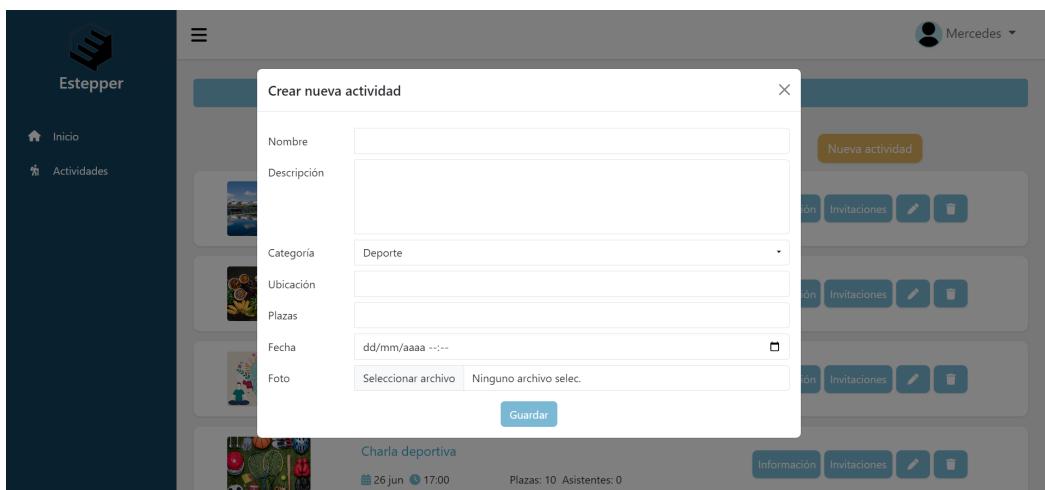


Figura 7.23: Vista de crear actividad siendo Coordinador

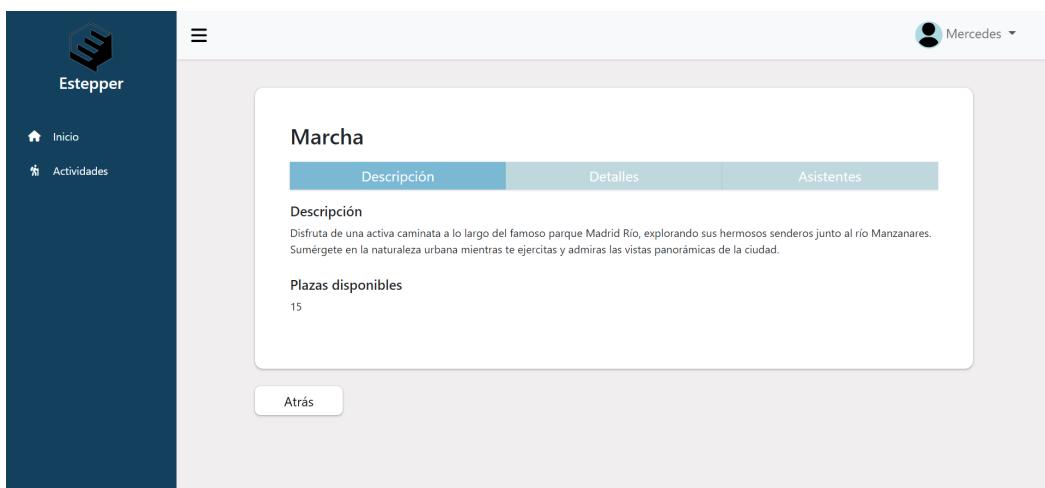


Figura 7.24: Vista de ver actividad siendo Coordinador

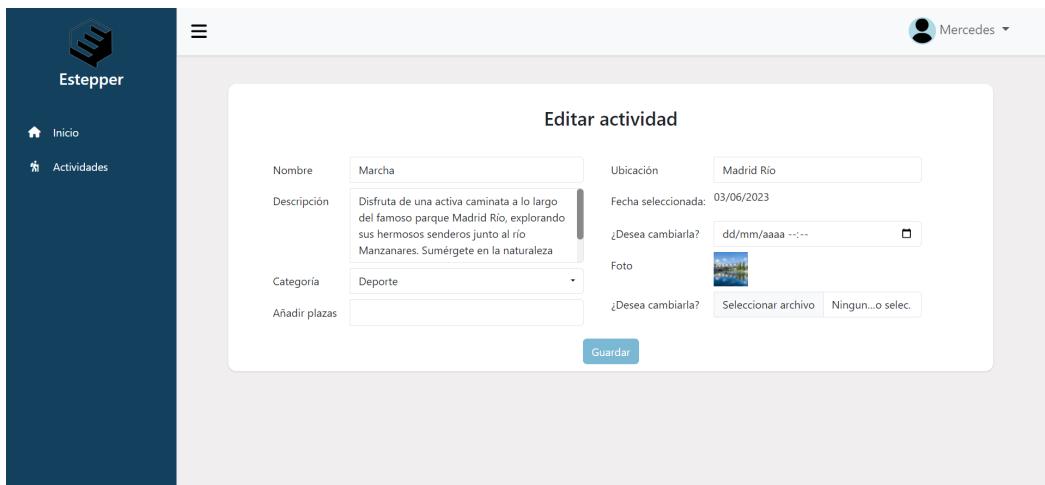


Figura 7.25: Vista de editar actividad siendo Coordinador

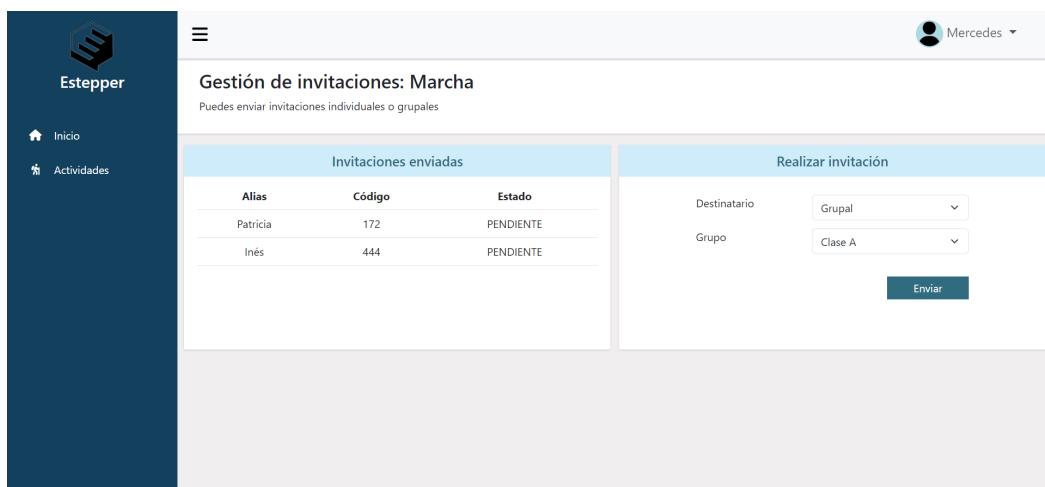


Figura 7.26: Vista de ver invitaciones siendo Coordinador

7.3.3. Materiales

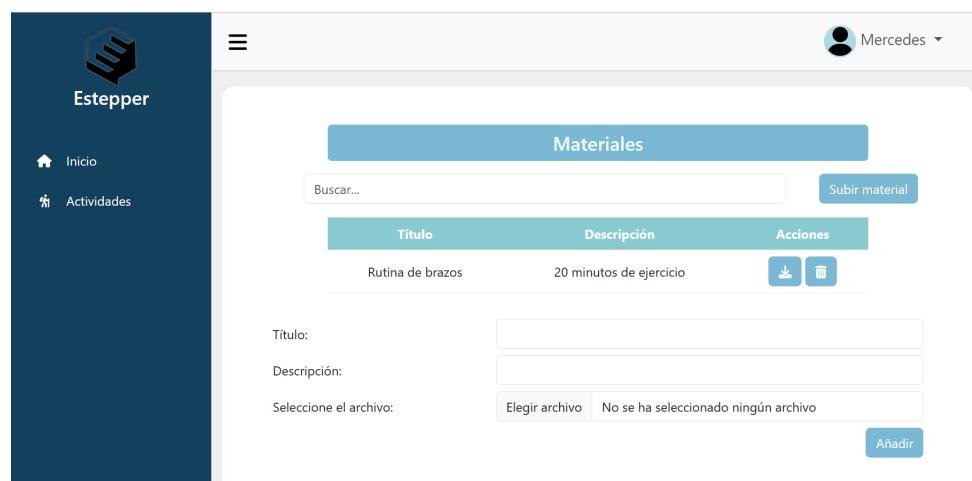


Figura 7.27: Vista de los materiales grupales e individuales siendo Coordinador

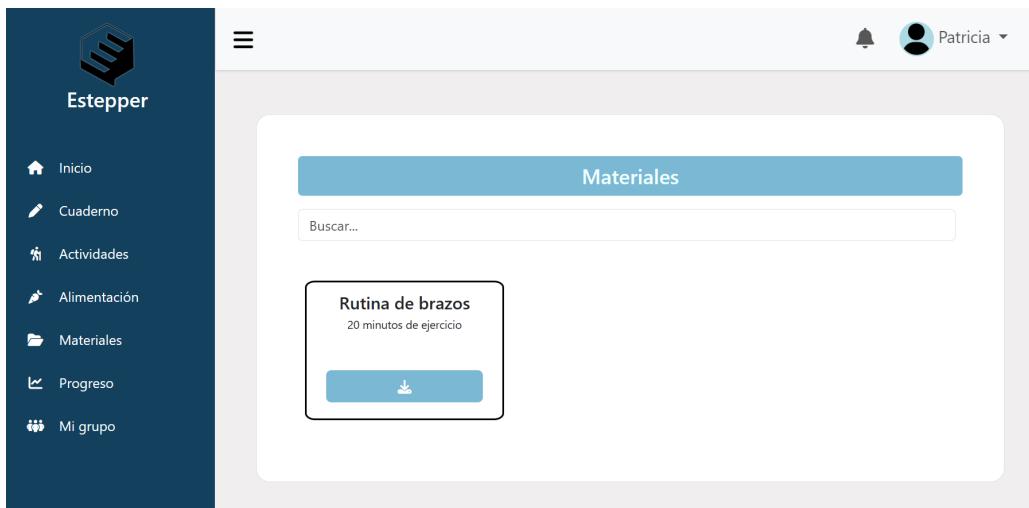


Figura 7.28: Vista de los materiales siendo Participante

7.3.4. Progreso

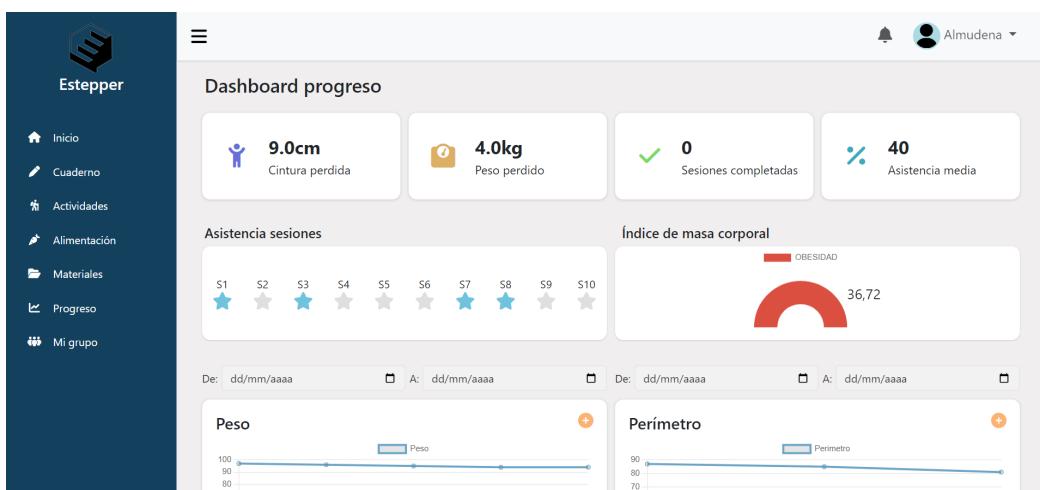


Figura 7.29: Vista de acceder al progreso propio siendo Participante

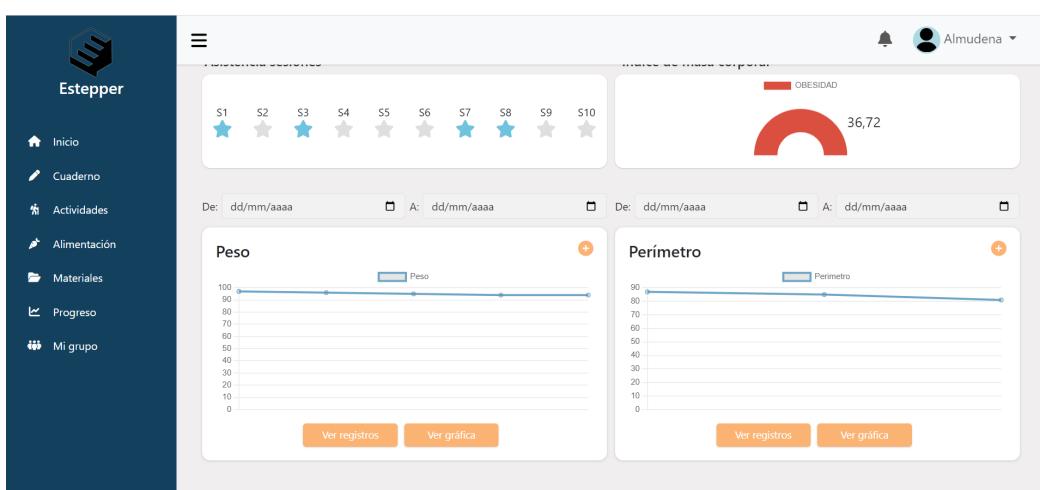


Figura 7.30: Vista de ver gráfica peso y perímetro siendo Participante

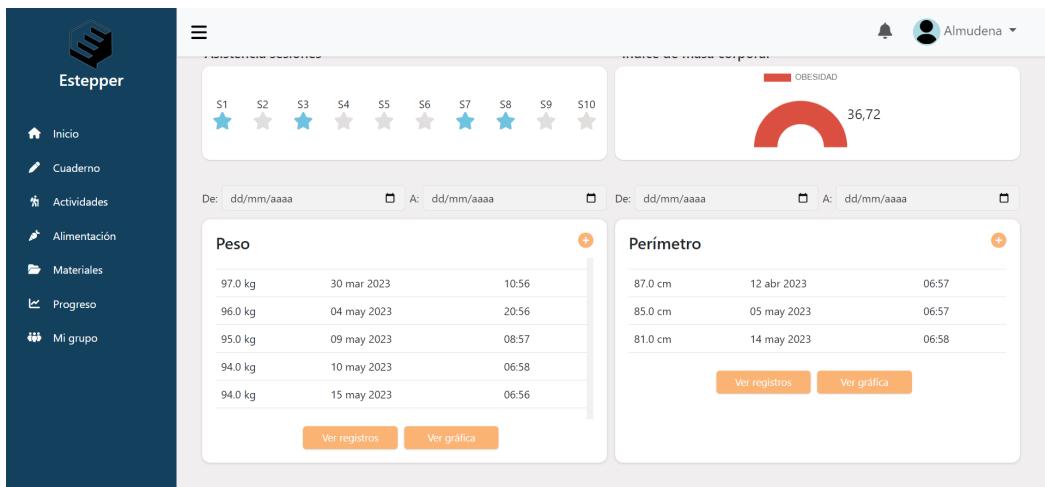


Figura 7.31: Vista de ver registros peso y perímetro siendo Participante

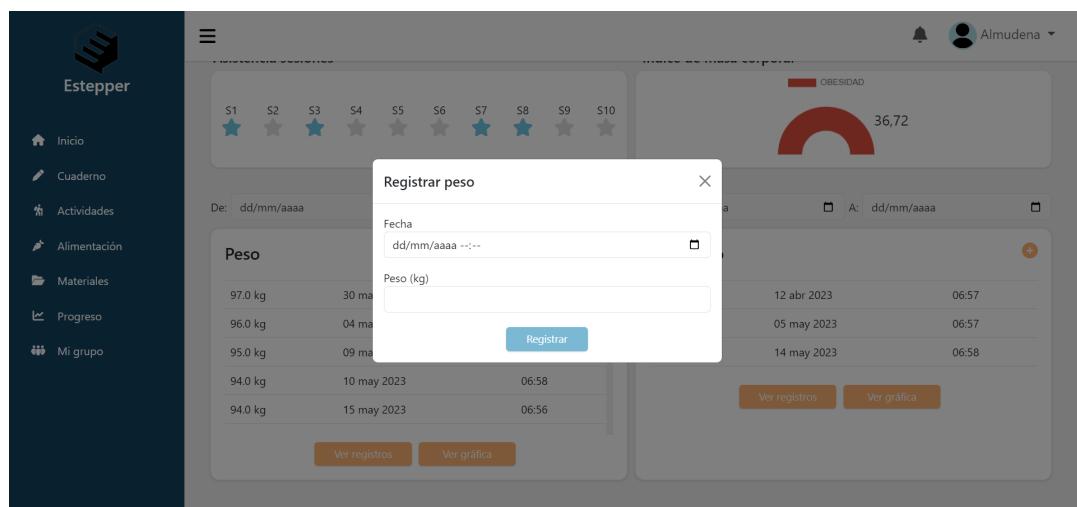


Figura 7.32: Vista de registrar peso siendo Participante

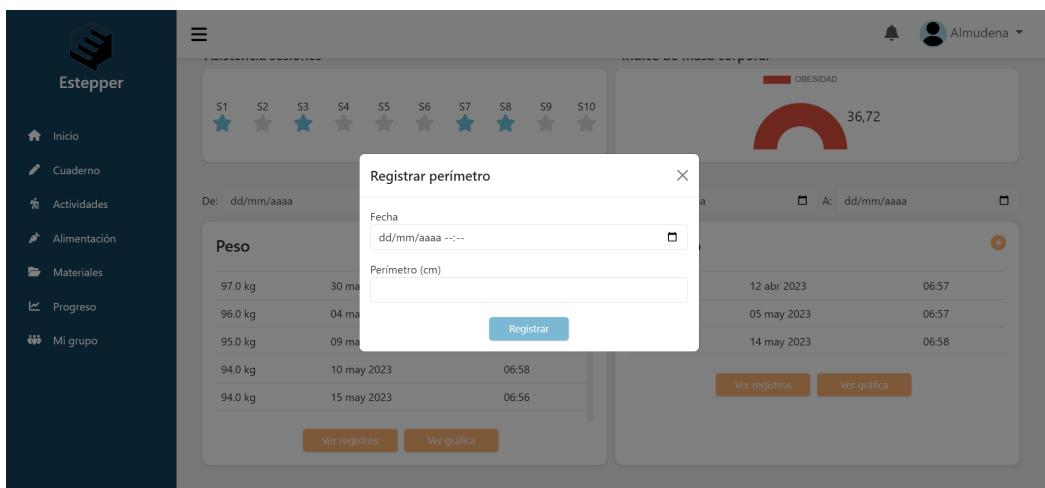


Figura 7.33: Vista de registrar perímetro siendo Participante

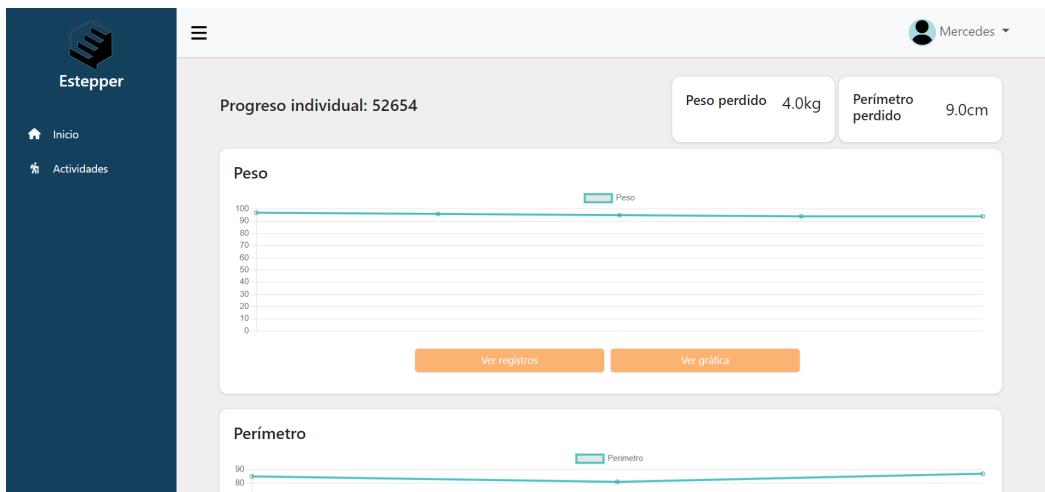


Figura 7.34: Vista de acceder al progreso de un participante siendo Coordinador

7.3.5. Chat

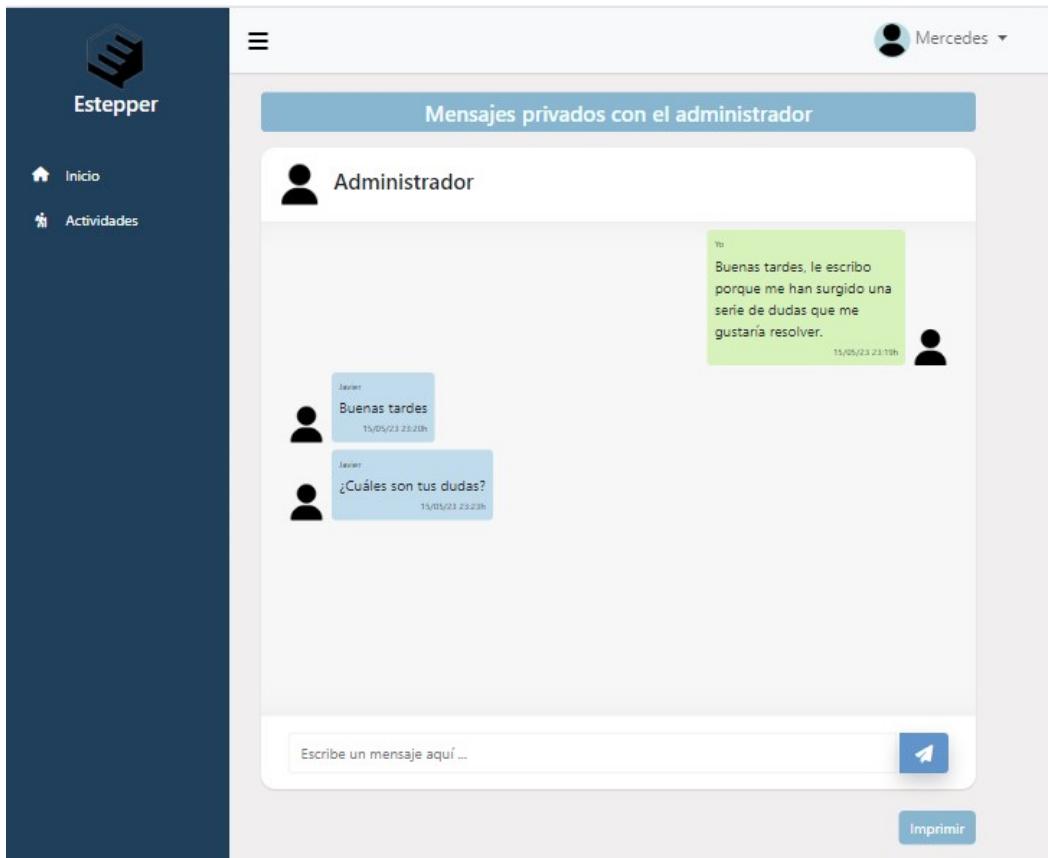


Figura 7.35: Vista de chat de ayuda siendo Coordinador

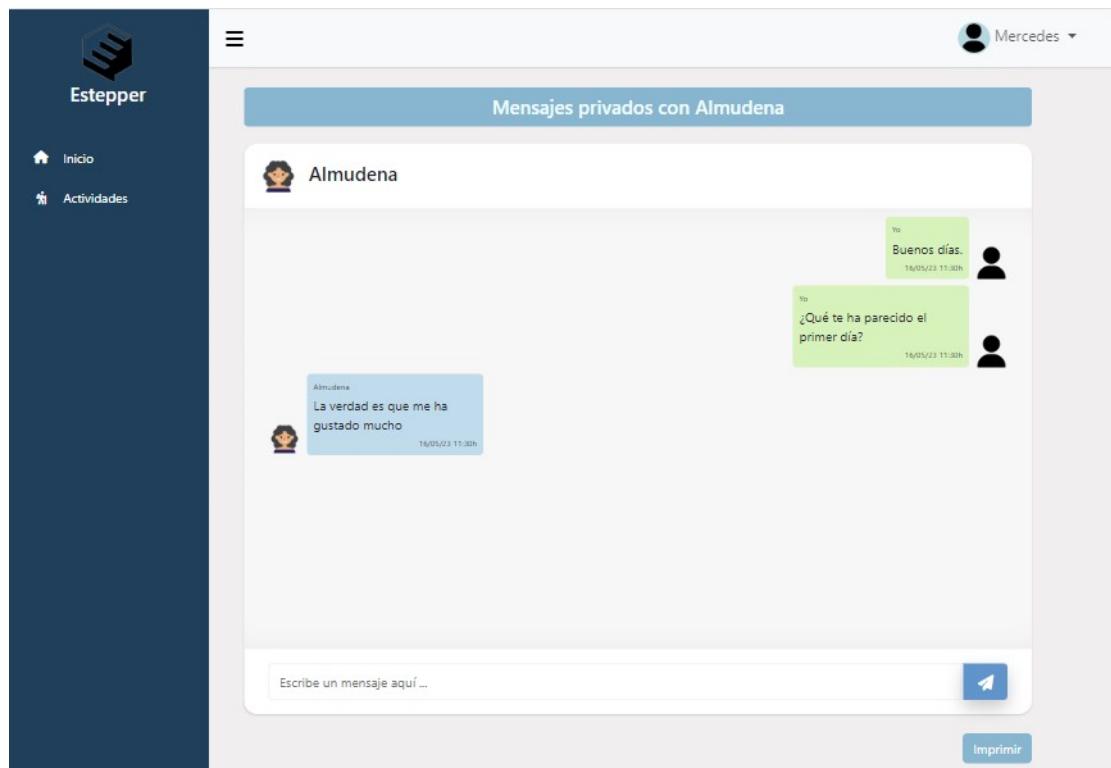


Figura 7.36: Vista de chat privado con un participante siendo Coordinador

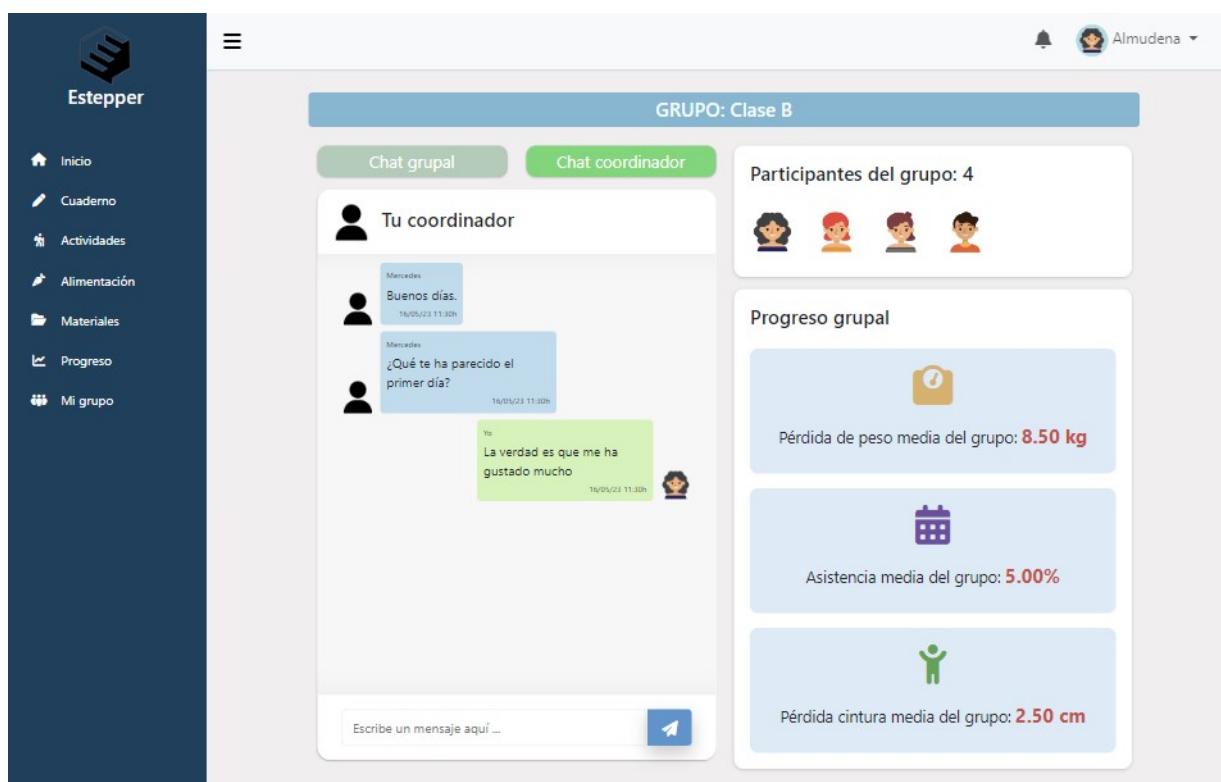


Figura 7.37: Vista del chat privado con el coordinador siendo Participante

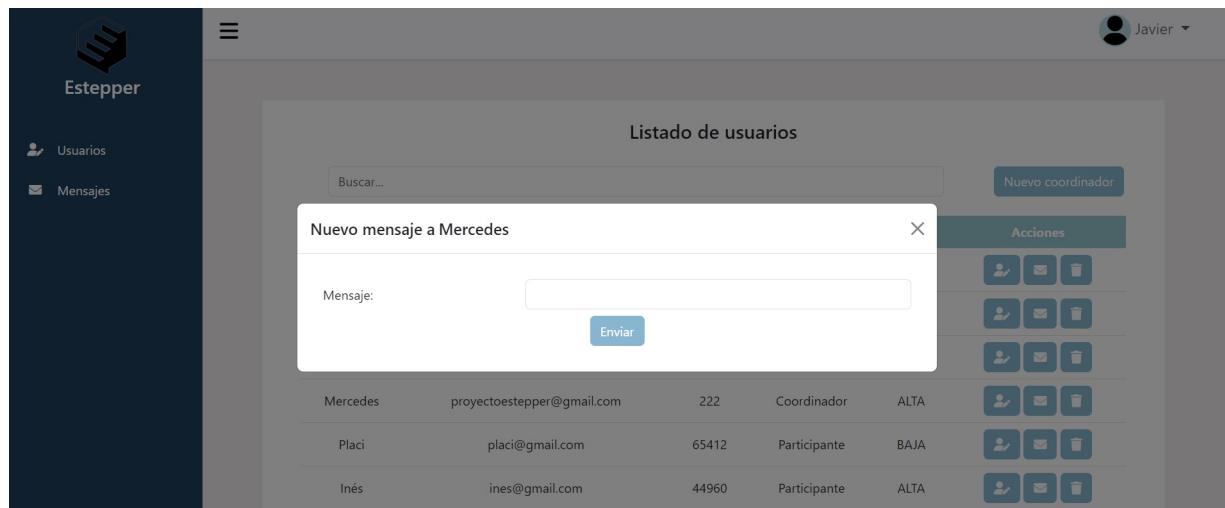


Figura 7.38: Vista de mensaje rápido siendo Administrador

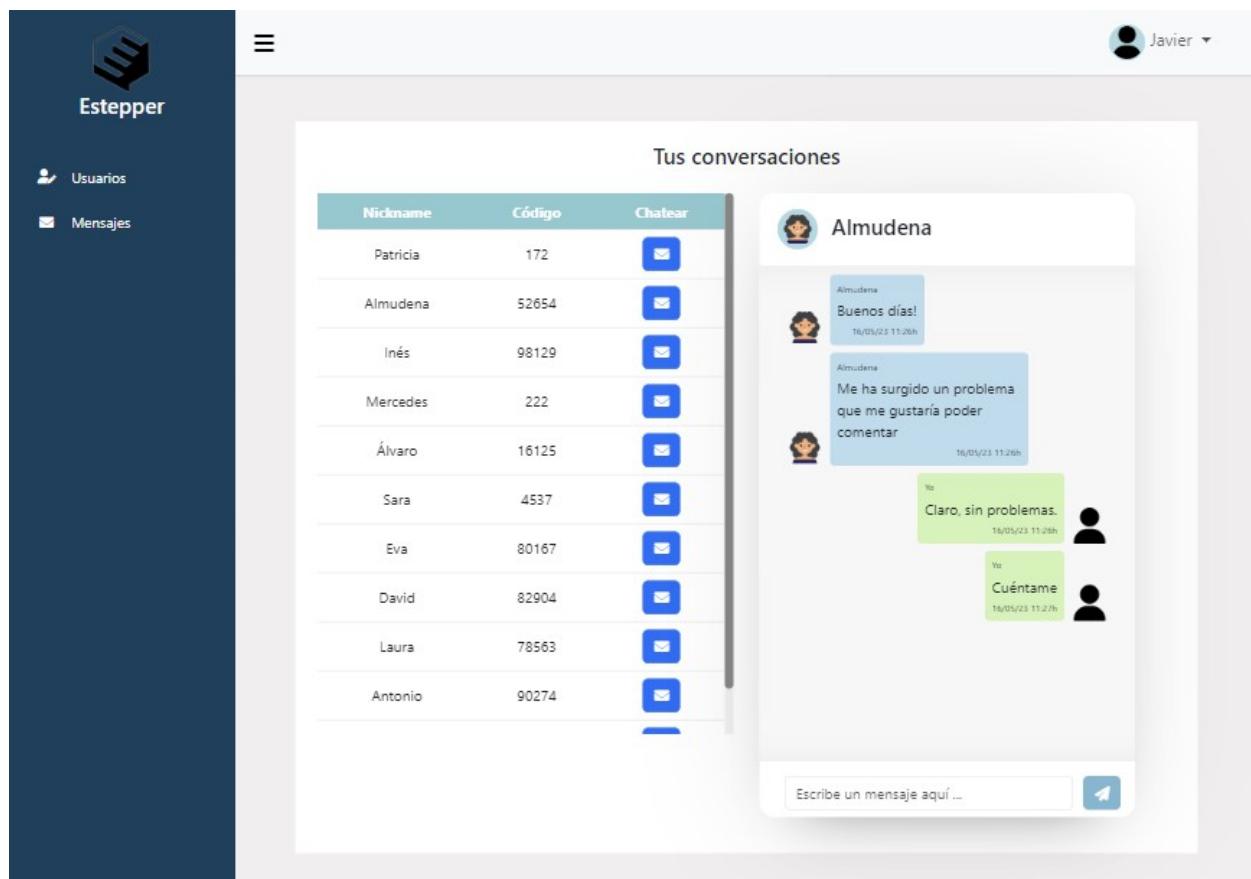


Figura 7.39: Vista de conversaciones privadas con usuarios siendo Administrador

7.3.6. Alimentación

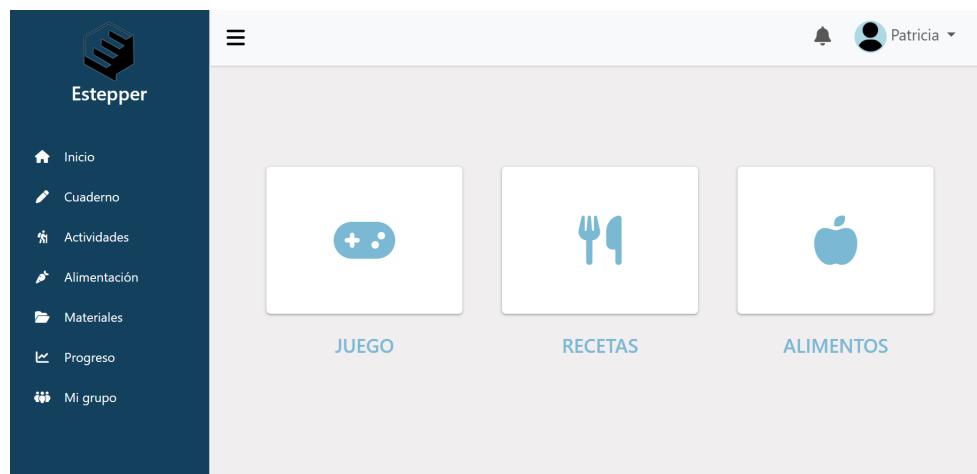


Figura 7.40: Vista del inicio de la funcionalidad siendo Participante



Figura 7.41: Vista del juego siendo Participante

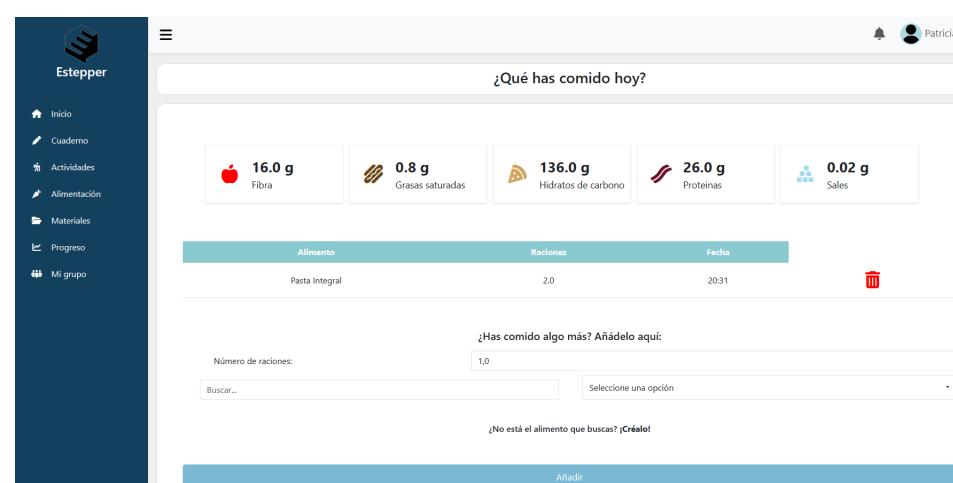


Figura 7.42: Vista de los alimentos de hoy siendo Participante

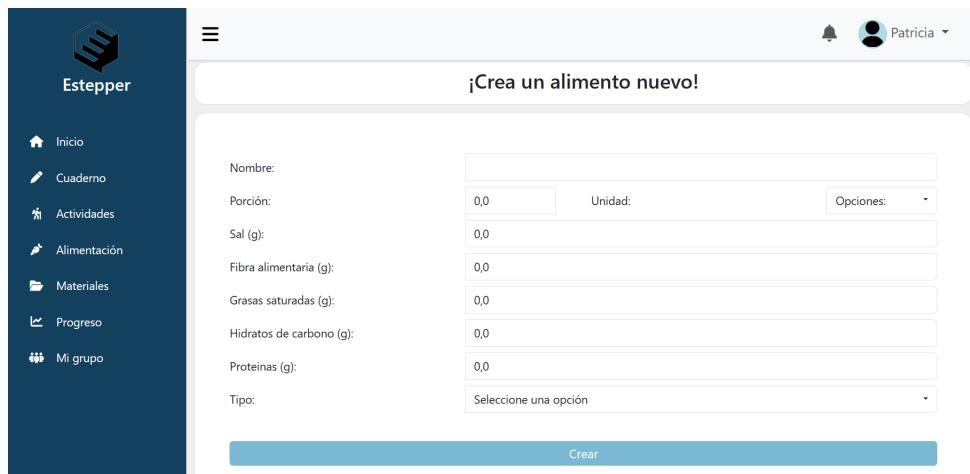


Figura 7.43: Vista de crear un nuevo alimento siendo Participante



Figura 7.44: Vista de las recetas siendo Participante

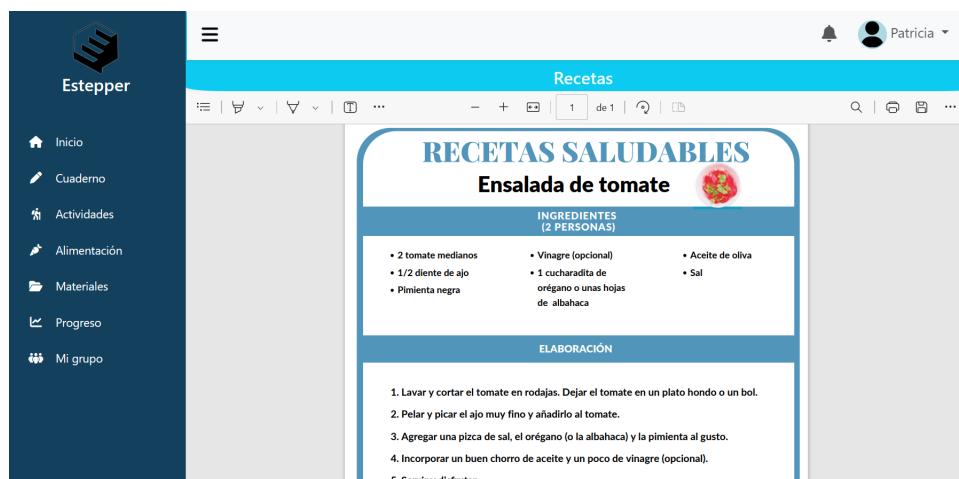


Figura 7.45: Vista de una receta siendo Participante

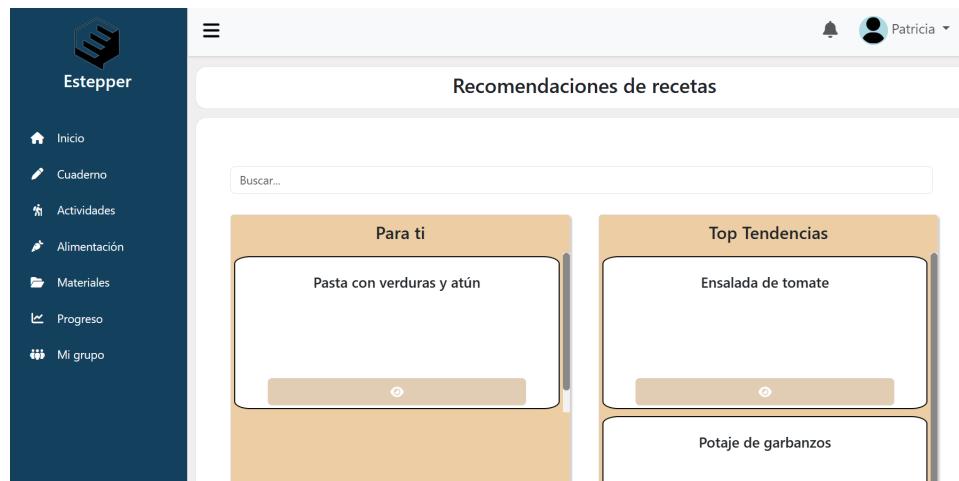


Figura 7.46: Vista de las recomendaciones de recetas siendo Participante

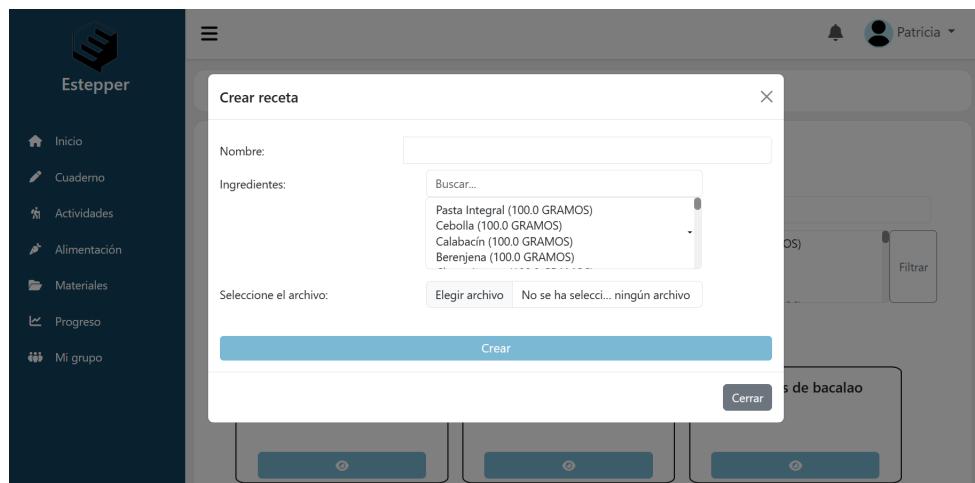


Figura 7.47: Vista de crear una receta siendo Participante

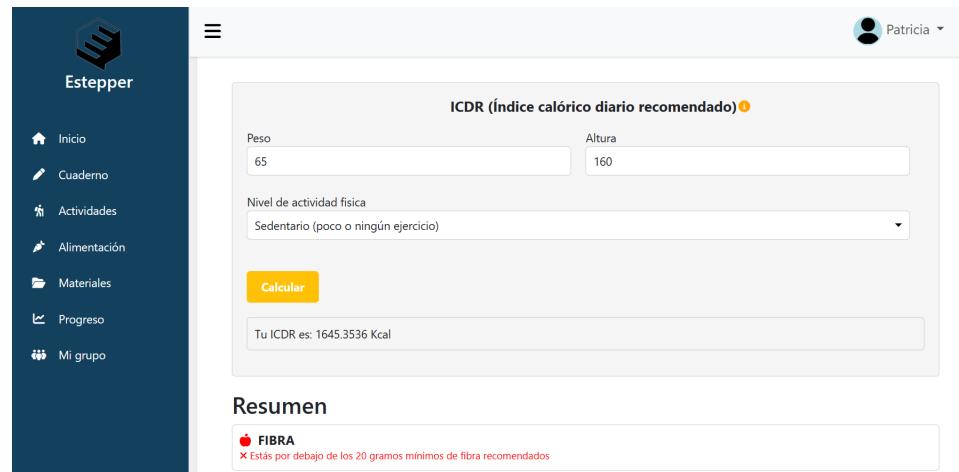


Figura 7.48: Vista de los nutrientes necesarios siendo Participante

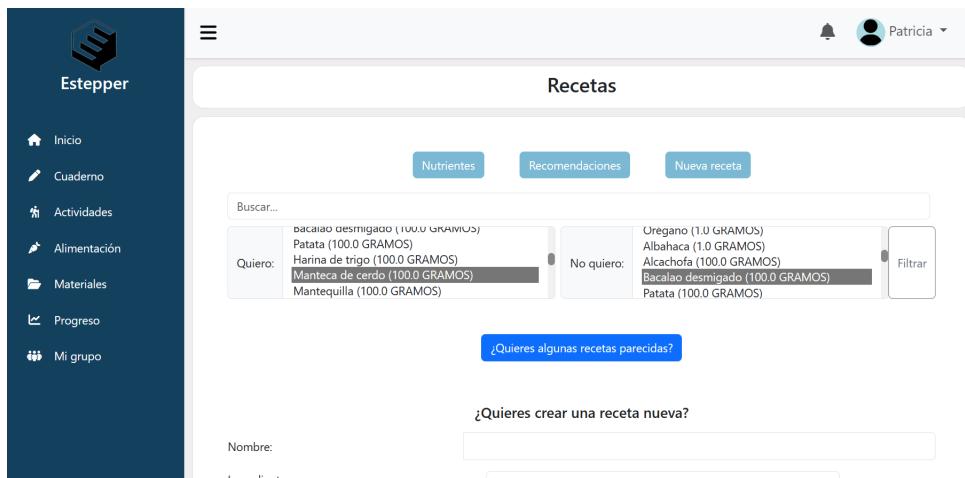


Figura 7.49: Vista del botón para ver las recetas parecidas siendo Participante

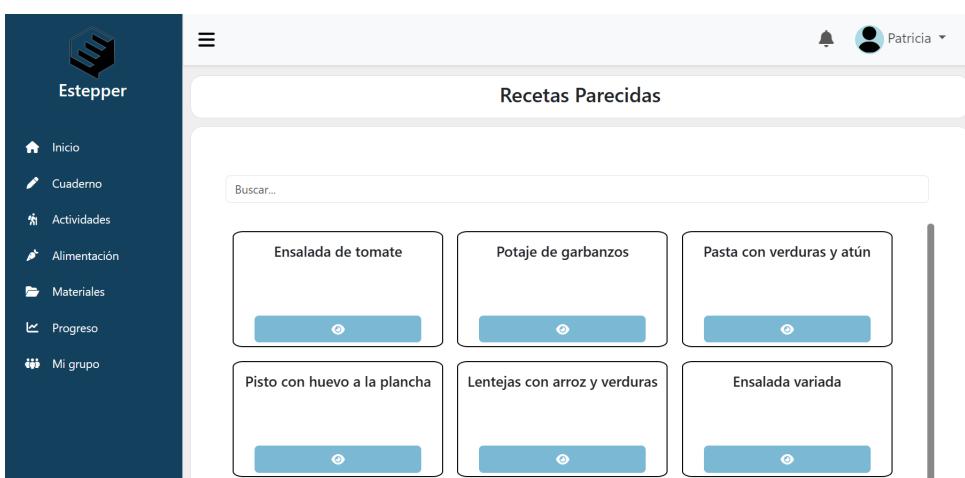


Figura 7.50: Vista de las recetas parecidas siendo Participante

7.3.7. Objetivos



Figura 7.51: Vista de objetivos recomendados (agua) siendo Participante

The screenshot shows the 'Mis Objetivos' (My Goals) section of the Estepper app. On the left is a dark sidebar with icons for Inicio, Cuaderno, Actividades, Alimentación, Materiales, Progreso, and Mi grupo. The main area has a header 'Objetivos recomendados' (Recommended Goals). Below it is a horizontal bar with four colored segments: Agua (blue), Ejercicio (green), Descanso (purple), and Test anímico (yellow). A sub-section titled 'EJERCICIO DE HOY' (Today's Exercise) shows 'BAILE: 45 minutos' (Dancing: 45 minutes) and 'NATACION: 30 minutos' (Swimming: 30 minutes). There are input fields for 'Tipo de ejercicio:' (Type of exercise:) set to 'NINGUNO' (None) and 'Duración (en minutos):' (Duration (in minutes)). A green button labeled 'Agregar ejercicio' (Add exercise) is at the bottom.

Figura 7.52: Vista de objetivos recomendados (ejercicio) siendo Participante

This screenshot shows the 'Mis Objetivos' section with a pink callout box stating 'Te recomendamos dormir 8 horas diarias' (We recommend sleeping 8 hours daily). It includes a field for 'Horas de sueño:' (Sleep hours:) with 'horas' (hours) and 'minutos' (minutes) dropdowns, and a 'Registrar' (Register) button. A message at the bottom says 'Todavía no has registrado el descanso de hoy' (You haven't registered today's rest yet).

Figura 7.53: Vista de objetivos recomendados (descanso) siendo Participante

The screenshot shows the 'Mis Objetivos' section with a yellow callout box asking '¿Cómo te sientes hoy?' (How do you feel today?). Below are nine circular icons representing different moods: happy, neutral, sad, angry, etc. A green 'Guardar' (Save) button is at the bottom. A message at the bottom says 'Todavía no has registrado cómo te sientes' (You haven't registered how you feel yet).

Figura 7.54: Vista de objetivos recomendados (estado de ánimo) siendo Participante

The screenshot shows the Estepper application interface. On the left is a dark sidebar with a logo and navigation links: Inicio, Cuaderno, Actividades, Alimentación, Materiales, Progreso, and Mi grupo. The main area has a header "Objetivos personalizados" and a date "15/5/2023". It displays two goals: "No desayunar bollería." and "Salir a correr". Below this is a calendar for May 2023. A button "Crear un objetivo nuevo" is at the bottom left, and a filter "Estado: Todos" is at the bottom right. A table lists the goals with columns for Title, Start Date, End Date, State, Repetition, and Actions.

Título	Fecha inicio	Fecha finalización	Estado	Repetición	Acciones
No desayunar bollería.	2023-05-15	2023-05-19	PENDIENTE	DIARIAMENTE	
Salir a correr	2023-05-15	2023-06-09	PENDIENTE	SEMANALMENTE	

Figura 7.55: Vista de objetivos personalizados siendo Participante

This screenshot shows the "Crear un nuevo objetivo" (Create new goal) dialog box. It includes fields for "Título del nuevo objetivo:" (Reducir el consumo de chocolate), "Fecha tope para cumplirlo:" (02/06/2023), and "Repetición:" (DIARIAMENTE). The background shows a calendar for June 2023 with the 15th highlighted. At the top of the screen, there are statistics for water intake: "Cantidad bebida (en litros): 1.5 / 2.0 L" and "Cantidad bebida (en vasos): 6 / 8 vasos", along with buttons "Añadir un vaso" and "Eliminar un vaso".

Figura 7.56: Vista de crear objetivo personalizado siendo Participante

7.3.8. Cuaderno y sesiones

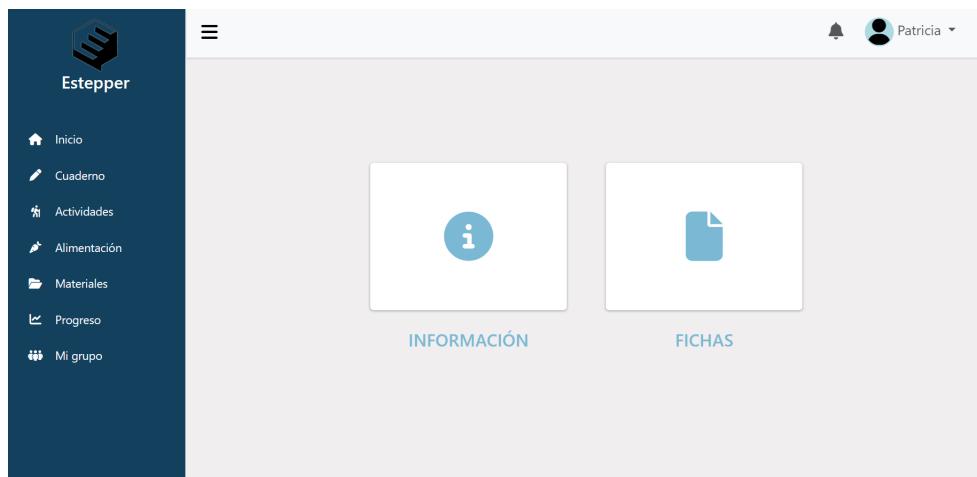


Figura 7.57: Vista de acceso al cuaderno siendo Participante

The screenshot shows the 'Diapositivas' (Slides) section of the Estepper app. The sidebar remains the same. The main area displays a presentation slide titled 'Diapositivas' with a table of nutritional information. The table has four columns: a placeholder column, a column for breakfast, a column for lunch, and a column for dinner. The breakfast row contains: 1 U de lácteos, 200 ml de leche; 2 U de fécula, 40 gr de pan; ½ U de grasa, 5 ml de aceite de oliva. The lunch row contains: Media mañana, 200 gr de pera. The dinner row contains: Comida, 300 gr de alcachofas; 4 U de fécula, 120 gr de guisantes; 2 U de proteínas, 40 gr de pan; 2 U de fruta, 100 gr de ternera; 1 ½ U de grasa, 200 gr de manzana; Cena, 15 gr de aceite. The last row is labeled 'Media tarde' and lists '1 U de lácteos' and '2 yogures'.

	1 U de lácteos 2 U de fécula ½ U de grasa	200 ml de leche 40 gr de pan 5 ml de aceite de oliva	
Media mañana	2 U de fruta	200 gr de pera	
Comida			
1 U de verdura 4 U de fécula 2 U de proteínas 2 U de fruta 1 ½ U de grasa	300 gr de alcachofas 120 gr de guisantes 40 gr de pan 100 gr de ternera 200 gr de manzana 15 gr de aceite		
Media tarde	1 U de lácteos	2 yogures	
Cena			

Figura 7.58: Vista de las diapositivas siendo Participante

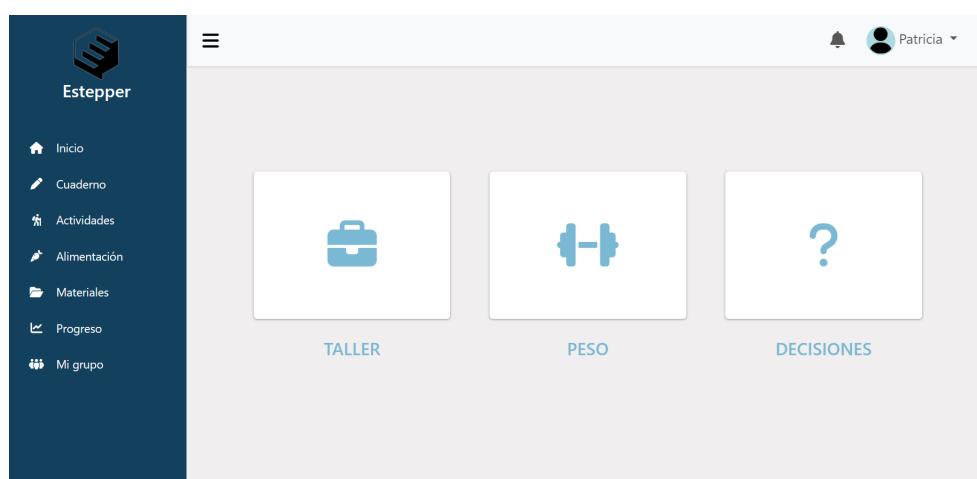


Figura 7.59: Vista de las fichas siendo Participante

Taller

Mis razones para participar en el taller:

¿Cuánto me importa? (Irrelevante - Muy importante)

¿Me siento capaz? (Nada - Totalmente capaz)

Mis temores:

Mis dificultades:

Guardar

Figura 7.60: Vista de la ficha del taller siendo Participante

Cálculo del índice de masa corporal (IMC) y del peso a perder

CÁLCULO DEL IMC

IMC = Peso en kilos / Talla en m²
IMC = 87.0 / 1.60² = 34

CÁLCULO DE PÉRDIDA DEL 5-10% DEL PESO

Peso: 87.0 kg
5% de pérdida: 4.35 kg
10% de pérdida: 8.70 kg
Peso a alcanzar:
Peso - 5% de pérdida: 82.65 kg
Peso - 10% de pérdida: 78.3 kg

PESO SALUDABLE: 64.00 kg

Figura 7.61: Vista de la ficha del peso siendo Participante

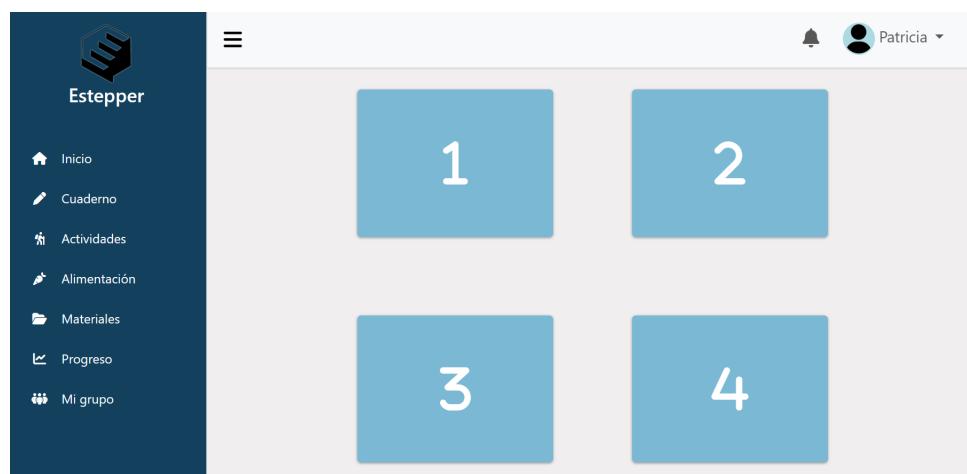


Figura 7.62: Vista de acceso a las fichas de elección siendo Participante

The screenshot shows a mobile application interface for 'Estepper'. On the left is a dark sidebar with navigation icons: Inicio, Cuaderno, Actividades, Alimentación, Materiales, Progreso, and Mi grupo. The main content area has a header 'Deciciones' with a question: '¿Cuál de estas dos galletas es más saludable? ¿Por qué?'. Below this is a section titled 'Compara calorías, grasas totales y saturadas' and 'GALLETAS DIGESTIVE'. A nutritional information table follows:

INFORMACIÓN NUTRICIONAL		
Valores Medios	Por 100 g de producto	Por ración 2 galletas (aprox. 32 g)
Valor energético	1.919 KJ / 459 Kcal	614 KJ / 147 Kcal
Proteínas	7,1 g	2,3 g
Hidratos de carbono	70,1 g	22,4 g
de los cuales Azúcares	21,2 g	6,8 g
Grasas	16,1 g	5,1 g
de las cuales Saturadas Monosaturadas	1,7 g 12,5 g	0,5 g 4,0 g
de las cuales Oleicos y Polinsaturadas	10,0 g 1,9 g	3,2 g 0,6 g
Fibra alimentaria	2,7 g	0,9 g
Sodio	0,7 g	0,2 g

Figura 7.63: Vista de la ficha de la elección siendo Participante

The screenshot shows a grid of ten session cards labeled Sesión 1 through Sesión 10. Each card is orange with white text. The sidebar on the left is identical to Figure 7.63.

Figura 7.64: Vista de acceso a las sesiones siendo Participante

The screenshot shows a detailed view of Session 2. The top bar says 'Sesión 2'. The page contains fields for 'Asistencia' (set to 'No'), 'Diapositivas' (linked to 'Taller online sesión 2.pdf'), 'Tus notas' (with a text input placeholder 'Escribe aquí...'), and 'Cuestionario' (linked to 'Cuestionario sesión 2'). At the bottom are 'Atrás' and 'Guardar' buttons, and a blue button that says '¿Has terminado la sesión?'

Figura 7.65: Vista de una sesión siendo Participante

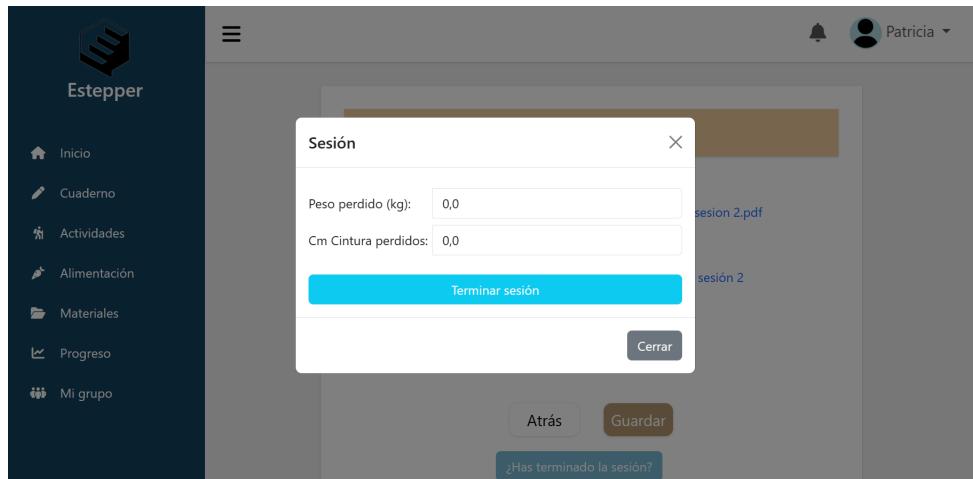


Figura 7.66: Vista de terminar una sesión siendo Participante

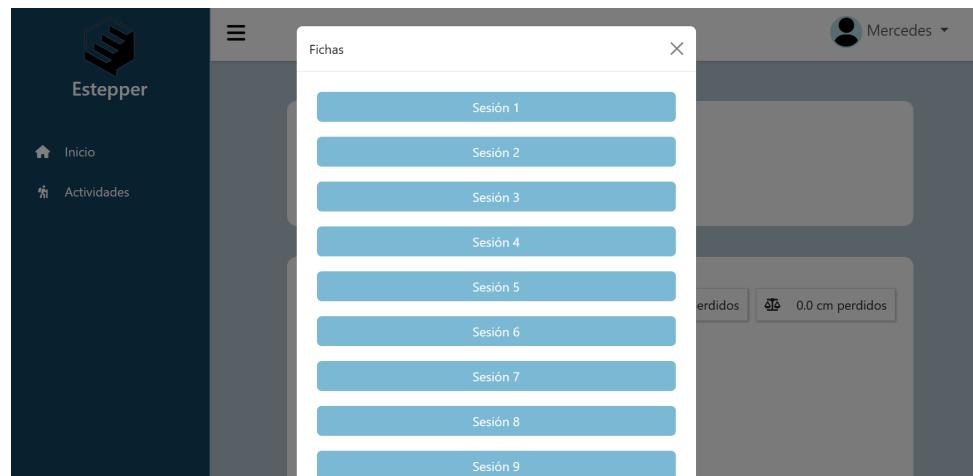


Figura 7.67: Vista de acceso a las sesiones siendo Coordinador

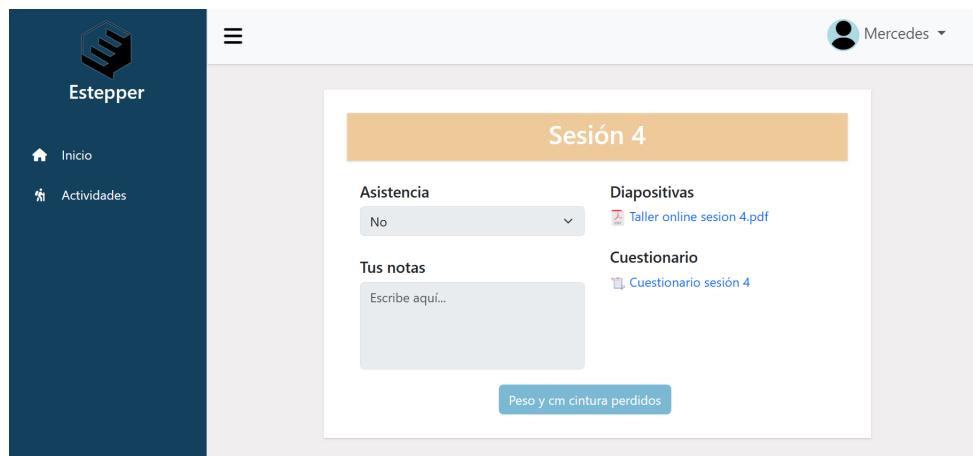


Figura 7.68: Vista de una sesión siendo Coordinador

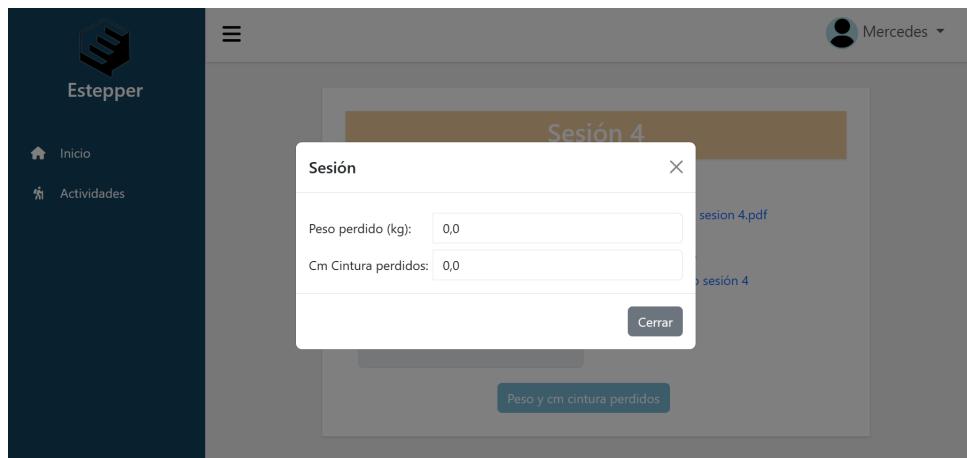


Figura 7.69: Vista de datos de peso y cintura siendo Coordinador

7.3.9. Gestión de participantes

The screenshot shows a 'TEST DE FINDRISC' form. It includes sections for age (1. Edad), BMI (2. Índice de masa corporal (IMC)), waist circumference (3. Perímetro de la cintura), and physical activity (4. ¿Normalmente practica usted 30 minutos cada día de actividad física en el trabajo y/o en su tiempo libre (incluida la actividad diaria normal)?). The BMI section has input fields for weight (Peso) and height (Altura), a 'Calcular' button, and a result field (Resultado IMC).

Figura 7.70: Vista de test de Finsdrisc siendo Usuario



Figura 7.71: Vista de recomendaciones siendo Usuario



Figura 7.72: Vista de las notificaciones siendo Participante

Alias	Código	Email	Estado	Acciones
Patricia	172	pplata@ucm.es	ALTA	
Almudena	52654	almunaha@ucm.es	ALTA	
Inés	98129	ineher02@ucm.es	BAJA	

Figura 7.73: Vista de la lista de participantes siendo Coordinador

Figura 7.74: Vista del expediente de un participante siendo Coordinador

Formulario	Editar
Exploración	
Findrisc	
Clasificación	
Antecedentes	
Alimentación	
Actividad Física	

Activar cuenta Eliminar cuenta Imprimir

Figura 7.75: Vista de la fase de valoración de un participante siendo Coordinador

Exploración

¿Es la primera vez que acude al programa? Sí

Género: FEMENINO

Peso (kg): 87,0

Perímetro de cintura (cm): 98,0

Talla/Altura (cm): 160

Edad: 40

IMC: 34

Guardar Imprimir

Figura 7.76: Vista del test de exploración de un participante siendo Coordinador

Findrisc

Edad: Menos de 45 años (0 puntos)

Perímetro de cintura (cm): Menos de 94cm (MASCULINO)/80cm (FEMENINO)

IMC: Menos de 25 kg/m²

¿Realiza habitualmente al menos 30 minutos de actividad física en el trabajo y/o en el tiempo libre? NO

¿Con qué frecuencia consume verduras o fruta? No a diario

¿Toma medicación para la presión regularmente? Sí

¿Le han encontrado alguna vez valores de glucosa (azúcar) altos? Sí

¿Alguno de sus familiares allegados y otros parientes han sido diagnosticados de diabetes (tipo 1 o tipo 2)? NO

Puntuación de Findrisc 17

Figura 7.77: Vista del test de findrisc de un participante siendo Coordinador

Figura 7.78: Vista del test de clasificación de un participante siendo Coordinador

Figura 7.79: Vista del test de antecedentes de un participante siendo Coordinador

Figura 7.80: Vista del test de adherencia a la dieta mediterránea de un participante siendo Coordinador

The screenshot shows the 'Actividad física' (Physical Activity) section of the application. At the top, there's a header with the title 'Actividad física'. Below it, under 'ACTIVIDAD FÍSICA VIGOROSA (A.F.V.)', there's a note about vigorous activities. It asks how many days per week and how much time per day/minute were spent on such activities. There are input fields for 'Nº días/semana' (0), 'Nº horas/día' (0), 'Nº minutos/día' (0), and 'Vigorosa METS-minuto/semana' (0). Under 'ACTIVIDAD FÍSICA MODERADA (A.F.M.)', there's another note about moderate activities. It asks how many days per week were spent on them. There is an input field for 'Nº días/semana' (0).

Figura 7.81: Vista del test de actividad física de un participante siendo Coordinador

7.3.10. Gestión de grupos

The screenshot shows the 'Listado de grupos' (List of Groups) section. At the top, there's a header with the title 'Listado de grupos'. Below it, there's a search bar and a dropdown for 'Estado' (Status). The main area is a table with columns: Nombre (Name), Código (Code), Número de participantes (Number of Participants), Fecha de creación (Creation Date), Fecha de finalización (End Date), Estado (Status), and Acciones (Actions). Three groups are listed: Clase A (Code CM3SEOCFTBD02G2, 3 participants, creation date 2023-04-07, status ACTIVO), Clase B (Code CFH7QMBDW2SCI66, 11 participants, creation date 2023-04-13, status ACTIVO), and Clase C (Code UFKVGX1QBAXPSAA, 2 participants, creation date 2023-05-15, end date 2023-05-08, status TERMINADO). At the bottom of the table, there are buttons for 'Crear un grupo nuevo' (Create new group) and 'Imprimir' (Print).

Figura 7.82: Vista del listado de grupos dirigidos por un Coordinador

The screenshot shows the 'Notas y observaciones' (Notes and Observations) section for a group. At the top, there's a header with the title 'Notas y observaciones'. Below it, there's a button for 'Crear una nota nueva' (Create new note). The main area is a table with columns: Fecha (Date), Nota (Note), and Acciones (Actions). Two notes are listed: '2023-05-15 Las chicas trabajaron mucho en la sesión del viernes.' and '2023-05-15 Realizar actividad 7 del cuaderno la semana del 25 de mayo'. At the bottom of the table, there are buttons for 'Imprimir' (Print).

Figura 7.83: Vista de las notas y observaciones de un grupo siendo Coordinador

Figura 7.84: Vista de un grupo siendo Coordinador

Figura 7.85: Vista de editar un grupo siendo Coordinador

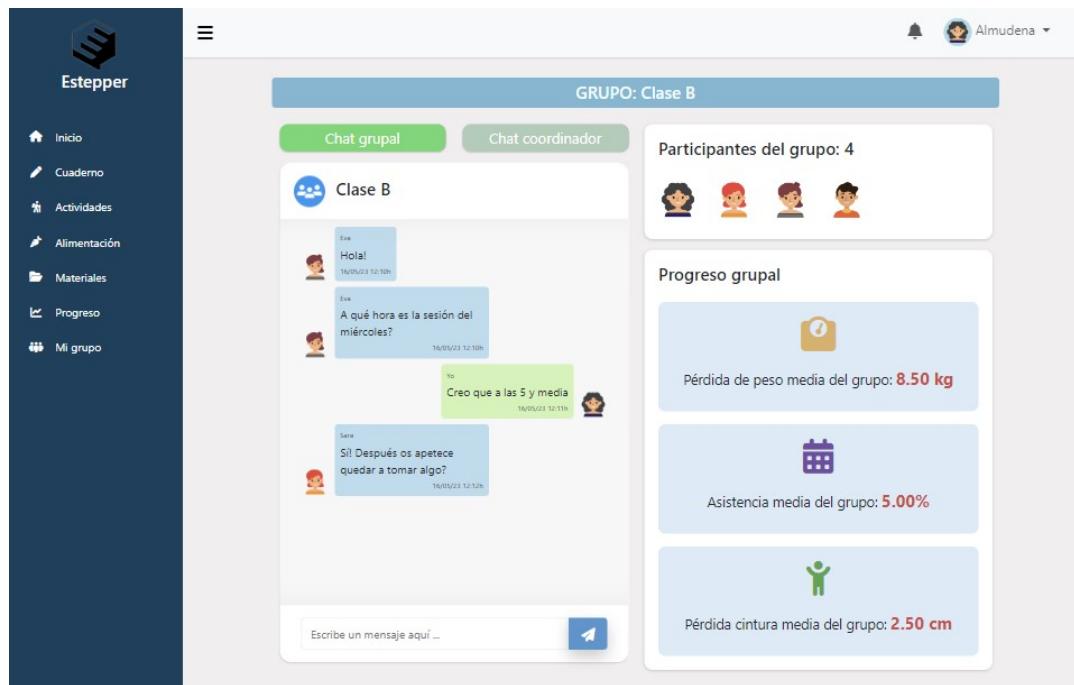


Figura 7.86: Vista de mi grupo siendo Participante

Capítulo 8

Resultados y discusión crítica

Con el objetivo de comprobar si los resultados obtenidos al desarrollar el proyecto se correspondían con las expectativas que se esperaban alcanzar, se ha contado con la ayuda de 7 usuarios a los que se ha dejado libertad para navegar por la aplicación. Dos de las personas que han podido testear la aplicación tenían una edad comprendida entre los 20 y los 35 años, cuatro tenían entre 40 y 60 años, y uno era mayor de 65.

De este modo, en un primer momento se les comentó en qué consistía el proyecto y la finalidad del mismo. En esta etapa, a modo de conversación, se les contestó a las preguntas que les iban surgiendo.

En una segunda fase, los usuarios tuvieron 10 minutos para hacer las pruebas que ellos consideraran oportunas. Pero, durante este período, no se podían resolver sus dudas, de forma que los miembros del equipo que ha desarrollado la aplicación se limitaban a observar. No obstante, sí se les facilitaron credenciales de acceso para los 3 tipos de usuarios (coordinador, participante y administrador) y se les pidió realizar pruebas accediendo a la aplicación con todos ellos.

A continuación, se inició una fase en la que en entrevistador le pedía realizar algunas acciones concretas al entrevistado para evaluar distintos aspectos. Entre estas acciones, se les pidió registrarse, eliminar participantes de un grupo, subir un material a un grupo, invitar a un usuario a una actividad, dar de alta a un participante, registrar los alimentos consumidos en un día, añadir un objetivo, registrar su peso, enviar un mensaje privado al coordinador y otro solicitando ayuda al administrador o jugar al juego interactivo, entre otras. Para terminar, se les planteó a modo de conversación una serie de preguntas sobre cómo le había resultado la aplicación. Algunas de estas preguntas fueron: ¿te ha parecido fácil navegar por la aplicación? ¿Has encontrado alguna dificultad a la hora de realizar alguna acción? ¿Consideras que es intuitiva? ¿Qué te parece la interfaz, es atractiva? ¿Se te ocurre alguna mejora a tener en cuenta para posibles desarrollos futuros? ¿Recomendarías la aplicación? Para terminar, se les agradeció su participación y se les mostró lo valiosa que resultaba su contribución en el desarrollo del proyecto.

Finalmente, el equipo de desarrollo de la aplicación se reunió para analizar tanto de manera individual como conjunta las sesiones que se habían llevado a cabo con las siete personas y se llegó a una serie de conclusiones. De la primera etapa, se

extrajo la idea de que todos los usuarios se mostraron interesados en el proyecto, les parecía una herramienta muy útil y práctica que incluso, preguntaron si podrían en un futuro hacer un uso personal de la misma. Tras la segunda fase, se observó una diferencia considerable en el uso entre los más jóvenes y la persona mayor de 65 años. Así, mientras que los dos primeros navegaron sin ningún tipo de problema y con una soltura destacable, la persona de mayor edad tardaba más tiempo en completar las tareas de forma que en los 10 minutos de navegación libre, se encontraba algo más perdida que el resto. No obstante, en la tercera etapa donde se le pedía realizar acciones concretas, se notó mucho la diferencia con la anterior, pues conseguía resolver las distintas cuestiones con relativa facilidad. En cuanto a los 4 individuos del grupo de edad intermedia, se observó que salvo momentos muy puntuales, en los que les surgieron pequeñas dudas como dónde se podía dar de alta a un participante, el feedback era muy positivo.

Por otro lado, respecto a la ronda de preguntas, todos los participantes consideraron que se trataba una aplicación con un diseño simple y atractivo. Con carácter general, era intuitiva y no se presentaban problemas en la realización de las tareas. En cuanto a las mejoras, se llegó a la conclusión de que sería una buena práctica añadir más botones del tipo "cancelar" o "atrás" para recordar al usuario dónde se encuentra en todo momento. También se sugirió la incorporación de funcionalidades como poder ver los pasos que da un usuario mediante algún dispositivo wearable. Otra de las propuestas que llamó la atención era realizar una especie de sistema de recompensas, de manera que si un usuario iba cumpliendo metas, como por ejemplo mantener una buena alimentación durante toda una semana o conseguir perder el peso deseado, obtuviera una bonificación que les permitiera canjear puntos por algo que les aumentara la motivación.

Gracias a esta pequeña evaluación, se realizaron diversas mejoras y se tuvieron en cuenta sugerencias para posibles desarrollos futuros. De esta manera, se consideró que el proyecto cumplía las expectativas adecuadamente y se terminó por hacer disponible la versión definitiva de la aplicación desarrollada. Para acceder a ella, puede hacerse mediante el repositorio de github (<https://github.com/almunaha/estepper>) o haciendo uso del servidor de ngrok en el que se encuentra desplegada entrando en <https://b960-147-96-81-61.ngrok-free.app>.

Además, hemos implementado una guía de instalación en el anexo B y una guía de uso en el anexo C con el fin de facilitar su acceso y uso.

Capítulo 9

Conclusiones y Trabajo Futuro

Conclusiones

Se ha creado una aplicación web destinada a la mejora del programa de alimentación, actividad física y salud (ALAS) que reúne las funcionalidades y herramientas necesarias para facilitar su trabajo en el seguimiento de los participantes. Ofrece a los coordinadores una plataforma para gestionar los participantes, grupos y sus progresos, permitiendo un seguimiento individualizado y en grupo. Además, la aplicación les permite enviar mensajes personalizados y orientación en tiempo real a través de un chat. Por otro lado, los participantes pueden acceder a herramientas y recursos que les permiten mejorar sus hábitos alimentarios y de actividad física, tales como ideas de recetas, alimentos recomendados, fichas y sesiones del programa, y un registro de alimentos consumidos durante el día para controlar fácilmente los nutrientes obtenidos. También pueden proponerse sus propios objetivos y hacer seguimiento de su progreso en términos de actividad física y hábitos alimentarios a través de un dashboard personalizado, lo que les permite tener una visión clara de su evolución y hacer ajustes en su plan de acción si es necesario. Finalmente, los participantes pueden comunicarse y recibir orientación en tiempo real de parte de los coordinadores y otros participantes a través de un chat. Por último, el perfil de administrador, que tiene la responsabilidad de manejar y controlar a los distintos usuarios que utilizan la aplicación.

Trabajo futuro

Como líneas de trabajo futuro, hemos considerado:

- Integración con dispositivos wearables: Permitir que los participantes conecten sus dispositivos wearables, como relojes inteligentes o pulseras de actividad, para que se puedan recopilar datos de su actividad física y sueño de forma automatizada.
- Gamificación: Introducir más elementos de juego en la aplicación, como desafíos y recompensas, para hacer que el proceso de mejora de hábitos alimentarios y actividad física sea más divertido y motivador.

- Redes sociales: Incorporar funciones de redes sociales en la aplicación, para que los participantes puedan compartir sus logros y recibir apoyo de otros usuarios que están pasando por lo mismo.
- Deporte: Se podría considerar la inclusión de una herramienta que permita a los participantes acceder a una serie de rutinas de ejercicios que complementen su plan de alimentación y nutrición. De esta manera, se busca fomentar un estilo de vida más saludable que incluya tanto una dieta adecuada como una actividad física regular y adaptada a las necesidades y objetivos individuales de cada persona.
- Servidor de la empresa: Se plantea implementar la aplicación dentro de un servidor propio, con el fin de mejorar la velocidad de carga y la estabilidad de la aplicación. Esto también permitiría tener un mayor control sobre la seguridad de los datos de los usuarios y garantizar la privacidad de la información personal. Así como, poder aceptar cookies desde los dispositivos que sean necesarios asegurando una conexión segura.

Introduction

“Technology, when applied correctly, can be a transformative agent in healthcare”
— Eric Topol

This chapter contains the translation of Chapter 1.

This section presents the web application project for improving nutrition and promoting physical activity. It details the background, objectives, as well as the work plan and subjects that have been useful in its realization.

Background

This work arises from the need to computerize part of the Nutrition, Physical Activity and Health (ALAS) program of the Autonomous Organization of Madrid Health. The program aims to facilitate regular physical exercise and promote healthy eating habits. It is designed especially for people who are overweight or obese and those at higher risk of developing diabetes and cardiovascular diseases.

To achieve these objectives, the ALAS program offers workshops aimed at the general population in order to inform and raise awareness about how healthy habits regarding nutrition and physical activity can improve health and quality of life. Additionally, the program targets the population of the city of Madrid in different areas, such as family and community, school, business, and health care.

The program also includes an intensive intervention with people who have different degrees of risk due to inadequate nutrition and sedentary lifestyle. This intervention focuses on offering these individuals personalized follow-up and a specific action plan to help them adopt healthy habits in their day-to-day lives and improve their health status. In summary, the ALAS program aims to promote a healthy lifestyle and prevent diseases related to nutrition and lack of physical activity.

Mercedes Martínez Cortés, one of the organizers of the ALAS program, has provided us with additional information on the improvements she plans to implement in the program to achieve its objectives. According to Mercedes, one of the main improvements she proposes is the implementation of a web application to better monitor participants and facilitate personalized follow-up.

In addition, Mercedes suggests the creation of tools and resources for participants, such as healthy meal ideas, recipes, and recommended foods, to help them control their diet and improve their habits. She also suggests the organization of activities

and events that allow participants to interact with each other and foster a supportive and motivating environment. Mercedes also proposes that participants be allowed to set their own goals to achieve them gradually, which can be of great help in fostering motivation and commitment to the program.

Another of the improvements proposed by Mercedes for the ALAS program includes the implementation of a chat so that participants can communicate and receive guidance in real time. Additionally, a dashboard will be created where participants can record and track their progress in terms of physical activity and dietary habits. This would allow participants to have a clear view of their progress and make adjustments to their action plan if necessary.

On the other hand, the creation of a virtual campus has been proposed to send and receive materials, such as healthy meal ideas, recipes, and recommended exercises. This will provide participants with easy access to the information and resources they need to carry out their action plan and achieve their goals.

In summary, Mercedes Martínez Cortés, organizer of the ALAS program, has shared additional information on the improvements she plans to implement in the program. These improvements include the creation of a web application, adaptable to mobile device screens, to better monitor participants and facilitate personalized follow-up, as well as the creation of tools and resources such as healthy meal ideas and recommended recipes. The implementation of a chat to allow participants to communicate and receive guidance in real time, a dashboard to record and track their progress, and a virtual campus to send and receive materials has also been proposed. In addition, participants will be allowed to set their own goals to achieve them gradually, fostering motivation and commitment to the program.

Objectives

We have divided the requirements posed by Mercedes Martínez Cortés into different functionalities in order to improve the effectiveness of the ALAS program. Each of these functionalities is described below:

User management: Implement a web application that is adaptable to mobile devices screens, which allows users to register, create their profile, and access the program's tools and resources.

Participant management: Facilitate individualized monitoring of participants by monitoring their progress, starting with an evaluation phase.

Group management: Facilitate the monitoring of participant groups to encourage their interaction by conducting a chat.

Notebook and sessions: Provide participants with tools and resources to improve their eating habits, computerizing the program's own files and sessions. In addition to monitoring attendance at different sessions and their progress.

Materials: Create a virtual campus to send and receive materials related to the program, such as ideas for healthy meals, recipes, and recommended exercises.

Chat: Implement a chat so that participants can communicate and receive real-time guidance from coordinators and other participants.

Activities: Organize activities and events that promote a supportive and motivating environment among participants, such as yoga classes, group walks, among others.

Objectives: Allow participants to propose their own objectives to gradually achieve them, thus fostering their motivation and commitment to the program.

Nutrition: Provide tools and resources to improve participants' nutrition, such as recipe ideas and recommended foods. In addition to monitoring the foods they have consumed during the day to easily control the obtained nutrients.

Progress: Create a dashboard for participants to record and track their progress in terms of physical activity and eating habits, allowing for a clear view of their evolution and making adjustments to their action plan if necessary.

Each of these functionalities is essential to provide ALAS program participants with tools and resources that allow them to improve their eating habits and increase their physical activity, as well as creating a supportive and motivating environment to foster their commitment and achieve success.

Work plan

For the realization of this project, it has been decided to use the Scrum methodology, which is based on an agile approach for the management and development of software projects.

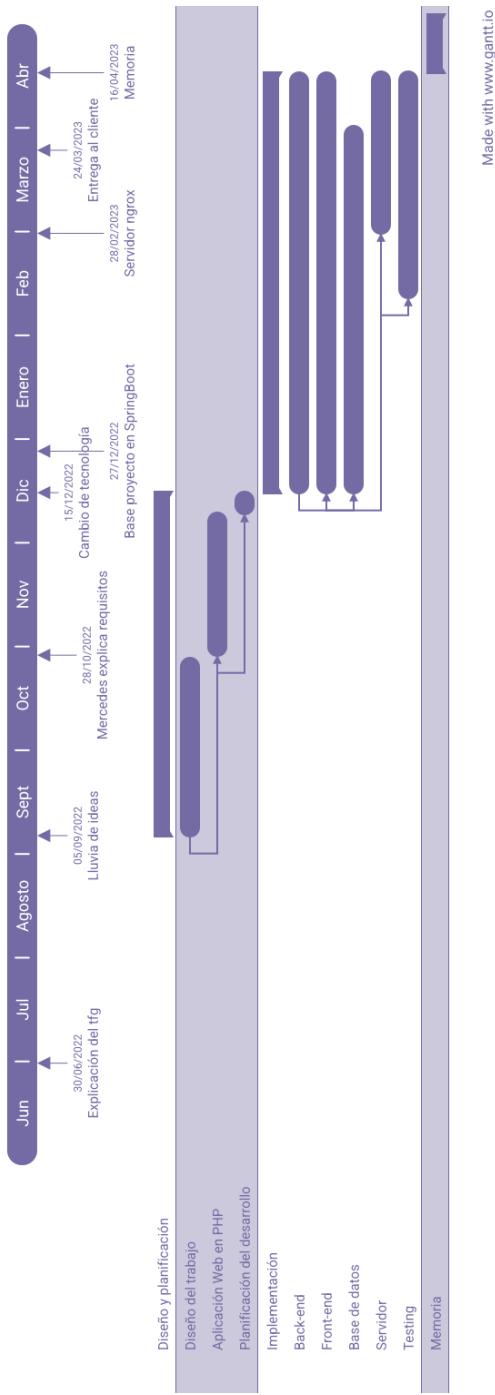
Scrum is composed of work cycles or sprints, in which the software is planned, designed, developed, tested, and presented in functional increments. In each sprint, the tasks to be performed are selected and assigned to team members. During daily meetings, progress is evaluated and adjustments are made if necessary. At the end of the sprint, a review of the work done is carried out, and the next sprint is planned. The work team is organized into well-defined roles, such as the Product Owner, the Scrum Master, and the developers. The Product Owner is responsible for defining the project's requirements and prioritizing the tasks to be performed, in this case, it has been Mercedes Martinez. The Scrum Master is the process facilitator, ensuring that Scrum principles are followed and eliminating obstacles for the team, in this case, it has been the final degree project tutor, Sonia Estévez. The developers are responsible for doing the work and ensuring that the sprint goals are met, which have been Almudena Naharro, Inés Hernández, and Patricia Plata.

To monitor the team's progress, we have created a shared document in which we have assigned tasks to be completed with a set deadline. In addition to regular meetings to ensure progress, review, and retrospective at the end of each sprint.

It is expected that the use of the Scrum methodology will allow the team to be more efficient and flexible in the development of the project, while ensuring customer satisfaction and the quality of the final product.

In addition, to carry out this work, we have based ourselves on different subjects of the curriculum, such as Software Engineering (SE), Web Applications (WA), Interactive System Design (ISD), Data Mining and Big Data Paradigm (DMBD), Programming Technologies (PT), and Databases (DB). These subjects have provided us with the necessary skills and knowledge to develop a quality project, following good practices and using the most suitable technologies.

We have designed a Gantt chart (Figure 9.1), to show graphically how we have organized ourselves and how we have been building this web application from scratch. Giving great importance to planning and design.



Made with www.gantt.io

Figure 9.1: Gantt Chart

As can be seen in the Gantt chart, during the design and planning stage of the project, the need to change technology arose. Initially, we were working on a web page using PhP. However, due to the complexity of the requirements to be implemented, the decision was made to switch to Java, as it has various tools and frameworks, such as SpringBoot, that facilitated the development of the website. Finally, we focused on documenting all the code in this report. For this purpose, we have structured it following the following headings:

1. Introduction: This section includes a summary of the purpose of this work and

the objectives to be achieved. It explains the initiative of this web application and why it arises, as well as what is intended to be achieved with the completion of this Final Degree Project. In addition, we have included a Gantt chart that describes the tasks within a time frame.

2. State of the Art: In this section, a market research has been conducted to identify companies and web applications that cover similar functionalities to those of this project.
3. Technologies Used: In this section, we have detailed all the technologies we have learned and used to implement the Final Degree Project.
4. Graphic Design of the Application: In this section, the issues involving the frontend design of the application are detailed, such as the logo, the colors used, the style, and the different sketches that have been made to carry out the application.
5. Specification of Requirements: In this section, the requirements necessary to implement the objective functionalities will be specified in detail. For this, we will use several use case diagrams.
6. Functional Design of the Application: In this section, the logic of the web application and its architecture are described in detail. More specifically, it talks about the database, the architecture, the entities, and the responsive code.
7. Implementation of the Application: In this section, the frontend and backend design of the application are explained. As well as aspects related to the framework used and the chosen model, using several graphics and images to show the structure.
8. Results and Critical Discussion: In this section, the results obtained once the code is finished are specified. In addition, it is discussed whether the established objectives have been met.
9. Conclusions and Future Work: In this section, a reflection is made on the results obtained and how it can be improved or expanded for the future.
10. Personal Contributions: In this section, the contributions of each of the developers are detailed.
11. Bibliography: In this section, the different sources and references that have served us to complete this Final Degree Project are shown.

Conclusions and Future Work

This chapter contains the translation of Chapter 9.

Conclusions

A web application aimed at improving the nutrition, physical activity, and health program (ALAS) has been created, which brings together the necessary functionalities and tools to facilitate their work in monitoring participants. It offers coordinators a platform to manage participants, groups, and their progress, allowing for individualized and group monitoring. Additionally, the application allows them to send personalized messages and real-time guidance through a chat. On the other hand, participants can access tools and resources that allow them to improve their dietary and physical activity habits, such as recipe ideas, recommended foods, program sheets and sessions, and a food log to easily monitor the nutrients obtained. They can also propose their own goals and track their progress in terms of physical activity and dietary habits through a personalized dashboard, allowing them to have a clear view of their evolution and make adjustments to their action plan if necessary. Finally, participants can communicate and receive real-time guidance from coordinators and other participants through a chat. Lastly, the administrator profile, which is responsible for managing and controlling the different users who use the application.

Future Work

As lines of future work, we have considered:

- Integration with wearable devices: Allowing participants to connect their wearable devices, such as smartwatches or activity bracelets, to collect data on their physical activity and sleep in an automated way.
- Gamification: Introducing more game elements into the application, such as challenges and rewards, to make the process of improving dietary and physical activity habits more fun and motivating.
- Social networks: Incorporating social network features into the application, so

that participants can share their achievements and receive support from other users who are going through the same process.

- Sports: Consideration could be given to the inclusion of a tool that allows participants to access a series of exercise routines that complement their dietary and nutrition plan. This way, we aim to promote a healthier lifestyle that includes both a proper diet and regular physical activity adapted to each person's individual needs and goals.
- Company Server: It is proposed to implement the application within a proprietary server in order to improve the loading speed and stability of the application. This would also allow for greater control over user data security and guarantee the privacy of personal information. In addition, it would be possible to accept cookies from necessary devices while ensuring a secure connection.

Contribuciones Personales

Almudena Naharro Muñoz

Para explicar las aportaciones que he realizado a lo largo del proyecto, se va a estructurar este apartado en base del proyecto, implementación del código y escritura de la memoria.

Durante las primeras reuniones del proyecto, al igual que mis compañeras, realicé una lluvia de ideas sobre cómo podríamos llevar a cabo la aplicación, teniendo en cuenta todos los requisitos que Mercedes nos comentaba y brindándole consejo en todo momento. Al principio de ponernos a desarrollar el código, todo el equipo nos organizamos para realizar los bocetos de la aplicación divididos por funcionalidades, para tener una idea más clara de cómo se iba a implementar.

Después de recibir la aprobación para seguir adelante, comenzamos a desarrollar el código utilizando el lenguaje PHP. En concreto, me encargué de establecer la base inicial del proyecto, que incluía la implementación del sistema de inicio de sesión y registro, la conexión a la base de datos y todas las configuraciones necesarias. Además, me enfoqué en diseñar la estructura de la aplicación, asegurándome de que estuviera bien organizada con un diseño limpio. También fui responsable de desarrollar la funcionalidad de progreso en la aplicación, realizando un dashboard en el que se muestran gráficas y otros datos. Durante todo el proceso, trabajamos en estrecha colaboración como equipo para asegurarnos de que el código estuviera bien estructurado y cumpliendo con los requisitos establecidos.

Como se ha comentado en el plan de trabajo, tuvimos que cambiar de tecnología por no ser la adecuada para este proyecto, pasando a utilizar Spring Boot. Durante un tiempo, estuve documentándome de esta tecnología, ya que era nueva para mí. Me encargué de implementar la base del proyecto inicializándolo en Spring Initializr, para posteriormente cargarlo en Visual Studio y empezar a desarrollar todos el código. Antes de que mis compañeros programaran en esta tecnología, realicé toda la parte de seguridad y acceso a la aplicación, como es el login, el registro y la implementación de los 3 tipos de entidades de usuario: participante, coordinador y administrador. Una vez la tecnología estaba más encaminada, proseguimos a volver a empezar lo que habíamos desarrollado. Además, también me dediqué a investigar y documentarme sobre los diferentes apartados que debía incluir en la memoria de este trabajo, elaborando el índice que aparece en el mismo.

En relación a la implementación del código, me enfoqué en desarrollar la funcionalidad de progreso, la cual implicó un uso intensivo de JavaScript debido a que se trataba de un dashboard de control interactivo. Una de mis contribuciones fue la implementación de la herencia de usuarios, donde creé tablas específicas para cada tipo de usuario que se extendían de la clase Usuario. Esta estructura de herencia se utilizó posteriormente en diversas entidades de la base de datos, permitiendo una mayor organización y modularidad en el sistema. Además, cree la vistas del listado del coordinador donde aparecen todos los participantes y también la vista de inicio del administrador, donde se administran todos los usuarios e implementé el css de los botones de acción para cada usuario y el funcionamiento del formulario de crear a un coordinador. Todas hemos aplicado mejoras a estas dos vistas.

Uno de los requisitos planteados por Mercedes fue la necesidad de mostrar un recordatorio en la aplicación para aquellos usuarios que no hubieran registrado su peso en el progreso durante una semana o más. Para cumplir con esta solicitud, implementé una ventana emergente que aparece al iniciar sesión, recordando al usuario la importancia de mantener su progreso actualizado.

Seguidamente, implementé la funcionalidad de actividades e invitaciones que son funcionalidades complementarias, ya que el coordinador se encarga de realizar invitaciones a las actividades. Para facilitar la experiencia de usuario, creé un buscador de actividades a cual le pudieras aplicar filtros. Luego seguí con la funcionalidad de las notificaciones de los participantes y por último implementé el juego en la parte de alimentación.

Para el juego, como trata de llenar las casillas de una tabla, tuve que crear matrices para cada una de las posibles soluciones y aplicar el método drag and drop (arrastrar y soltar) que no conocía y tuve que documentarme antes de utilizarlo. Además, para los elementos que se introducen en la tabla, realicé un archivo sql con una base de datos completa de alimentos con su respectiva cantidad y tipo de grupo. Por último, desarrollé una de las fichas del cuaderno, la ficha de marcar el objetivo del peso así como su css.

Para realizar validaciones de los campos que se introducen en los formularios del registro, el login, las actividades, el progreso, crear un coordinador, editar perfil y editar actividad, implementé un archivo de JavaScript en el que se comprueban los requisitos para cada campo, mostrando mensajes de error en el formulario en caso de haber errores. Además, cree la clase @ApiRestController que es donde se realizan las peticiones HTTP a la API REST para realizar consultas a la base de datos.

Dentro de mi contribución al proyecto, me he encargado de implementar el CSS para diversos elementos clave de la aplicación. Esto incluye el diseño del menú lateral, el encabezado que muestra la foto de perfil del usuario y el desplegable con opciones como ver perfil, ayuda y cerrar sesión. Además, he trabajado en la creación de la vista para el estado de baja de un participante, así como en la página que muestra el código generado al registrarse. En colaboración con mi compañera Patricia, hemos desarrollado la vista de sesiones. También he realizado modificaciones en el estilo de las páginas de error, como el error 400 y el error 500.

Por otro lado, he añadido el control de acceso a muchos scripts para que el usuario en caso de cambiar la url manualmente no pueda redirigirse a cualquier página sin tener permiso, por ejemplo acceder al editar perfil de otra persona.

Respecto al servidor, me encargué junto a mi compañera Inés, de descargar todos los programas necesarios en el ordenador donde desplegamos la aplicación, para poder subirla a ngrok y crear un enlace público. En respectivas ocasiones, el enlace ha sido actualizado para poder ver posibles mejoras o errores y para que Mercedes pudiera testear la aplicación.

Quiero recalcar que he ayudado a corregir errores de mis dos compañeras siempre que lo han necesitado, como realizar la descarga de materiales, resolver errores del test de findrisc, mejorar el funcionamiento del chat, la implementación de las alertas en el listado de participantes, entre otras muchas cosas, aplicando mejoras en varias funcionalidades. Además, como fui la encargada de crear el controlador API REST, expliqué el funcionamiento a Inés para que pudiera utilizarlo en los datos que aparecen en el inicio del participante.

Por último, en el desarrollo de la memoria me he encargado de realizar el resumen del trabajo en español como en inglés, he documentado cada tecnología utilizada en el proyecto así como los pasos de cómo se ha llevado a cabo el servidor con ngrok. Por otro lado, he reflejado cómo hemos adaptado la aplicación a todos los dispositivos móviles explicando el proceso y poniendo ejemplos. Cabe destacar que he participado en gran parte en el desarrollo de la adaptación responsive, implementando el encabezado y el menú fijos del móvil, así como las pantallas iniciales de cada funcionalidad y otras vistas. Además, he creado un diagrama para explicar la arquitectura que sigue la aplicación y el proceso del flujo de las peticiones HTTP.

En cuanto al resto de la memoria, he implementado los diagramas de casos de uso de las funcionalidades de gestión de usuarios, actividades y progreso, así como los casos de uso correspondientes y los diagramas de actividad. Además, para cada funcionalidad se ha realizado por cada caso de uso una explicación de cómo se ha desarrollado el código en la parte del backend. Indicar que también he implementado el backend del juego de la parte de alimentación, las notificaciones que recibe el participante y la ficha objetivo.

Inés Hernández Hurtado

Para reflejar las aportaciones individuales, hemos optado por dividir este apartado en tres secciones diferenciadas: el planteamiento y estructura del proyecto, la implementación del código y la elaboración de la memoria.

En lo que respecta al planteamiento inicial, antes de empezar a programar fue necesario reunirnos para establecer los objetivos, requisitos, tanto propios como los propuestos por Mercedes, y pautas que debíamos seguir. A partir de ese momento, participé activamente en la propuesta de ideas, ejecución y toma de decisiones sobre elementos como el nombre, logo, funcionalidades básicas, estructura de la base de datos o cuestiones de estilo y colores que se iban a utilizar, entre otros aspectos.

De este modo, una vez que las bases estaban claras, comencé a diseñar una gran cantidad de bocetos con la aplicación de GoodNotes para ipad, que posteriormente, resultaron ser una guía muy útil para la fase de implementación. Además, de manera paralela realicé un curso online de desarrollo de apps móviles pensando en cómo aprovecharlo en el proyecto. De hecho, gracias a ello se planteó la opción de utilizar Android Studio, pero después de informarnos más sobre la herramienta lo acabamos descartando. También, empecé a realizar el prototipo en Balsamiq para darle al proyecto un enfoque más realista.

El siguiente paso fue empezar a programar. De esta manera, nos dividimos funcionalidades y me tocó la parte del test de findrisc, gestión de grupos y actividades. A medida que avanzábamos con la implementación nos surgieron problemas que nos hicieron dudar sobre si el método que seguíamos era el más adecuado. Por ello, contacté con un profesor de la facultad de informática experto en aplicaciones web y, tras una reunión con él, optamos por abandonar PHP y cambiar de tecnología. Por consiguiente, se hizo un parón en la implementación y, al igual que mis compañeras, me dediqué a documentarme en profundidad sobre SpringBoot y otros recursos que íbamos a utilizar.

Al acabar el primer cuatrimestre, se comenzó a implementar de nuevo la aplicación, reorganizando el reparto de tareas. Para familiarizarme con la nueva tecnología, empecé programando lo que consideraba más sencillo y poco a poco, pasé a desarrollar funcionalidades más complejas. Así, lo primero que hice fue la parte del test de findrisc, basándome en formularios de Bootstrap para que, en función de la puntuación obtenida se motivara al usuario a registrarse o se le mostraran una serie de recomendaciones, que también implementé. A continuación, me dediqué a programar toda la parte de gestión de grupos de la aplicación a la que progresivamente fui aplicando mejoras. De este modo, por su papel transversal, opté por realizar un controlador aparte para manejar todo lo relacionado con los mismos. Así, lo primero que hice fue la posibilidad de crear un nuevo grupo desde la vista del coordinador. Para ello, controlé que al crear el grupo el coordinador pudiera añadir directamente en un solo paso a todos aquellos participantes del proyecto que no formaran parte de ningún otro grupo. Lo siguiente que realicé, también desde la perspectiva del coordinador, fue la vista del listado de grupos, donde se pueden visualizar los datos más relevantes del mismo, teniendo en cuenta aspectos como que los grupos tienen una duración determinada. En este punto, más adelante decidí incluir un buscador en tiempo real con el objetivo de hacer más eficiente la navegación del usuario en la aplicación. También, creé un filtro para poder distinguir entre grupos activos y terminados e implementé la paginación. Además, incorporé el buscador y filtro al listado de participantes. Asimismo, en este último listado incluí la posibilidad de añadir un participante concreto a un grupo, controlando y mostrando mediante una alerta modal los casos en los que no podía cumplimentarse esta acción. Después continué desarrollando la vista de un grupo. En ella, incorporé a su vez diversas funcionalidades. Además de reflejar los datos estáticos del grupo, a partir de los parámetros de progreso individual de los participantes que implementó Almudena, calculé y mostré estos parámetros a nivel grupal. También me encargué de crear el chat para los participantes y coordinador de cada grupo. Por otro lado, como

Mercedes comentó que le gustaría poder anotar comentarios sobre la evolución de los grupos, implementé la sección de notas y observaciones sobre las que se pueden realizar el conjunto de operaciones CRUD, al igual que con los grupos. También, mientras hacía el editar grupo incorporé una forma rápida de eliminar participantes de este. Cabe destacar que, para diversas acciones como eliminar las observaciones o los grupos, hice alertas modales para solicitar confirmación. De igual manera que hice con el coordinador, implementé la parte correspondiente a grupos desde la perspectiva del participante.

Otra de las funcionalidades que desarrollé por completo fue la de objetivos. Para llevarla a cabo, tuve en cuenta distintos tipos de objetivos: objetivos recomendados similares para todos los participantes y objetivos personalizados. De este modo, realicé las distintas operaciones que se ven en la aplicación para el objetivo recomendado de agua, ejercicio, descanso y test anímico. En cuanto a los objetivos personalizados, escogí realizar un calendario dinámico y una tabla que permitiera a los propios participantes llevar un control sobre sus metas. Así, al crear un nuevo objetivo, atendiendo al tipo de repetición, este se incorpora y visualiza tanto en el calendario como en la tabla. También agregué la acción de pulsar sobre el calendario de manera que, a su izquierda a modo de notas, se visualizan los objetivos del día seleccionado. Al igual que para el resto de elementos, me encargué de que estos objetivos sean editables y se puedan eliminar. Del mismo modo, implementé un filtro de objetivos pendientes y completados que se activa en caso de que éstos existan. Además, para dar feedback al usuario, implementé un control sobre el cumplimiento o no de los objetivos diarios recomendados. Esto lo añadí a la vista de inicio del participante, que también realicé, junto a gráficos para mostrar el porcentaje de asistencia y de progreso. Además de lo anterior, también he sido responsable del desarrollo del sistema de comunicación de la aplicación. Para ello, implementé varios tipos de chats, además del grupal ya mencionado. No obstante, es cierto que me surgió un problema que mi compañera Patricia me ayudó a solventar. En esta línea, desarrolté chats individuales entre coordinador y participante y también el mecanismo de solicitar ayuda que pone en contacto mediante un chat a cualquier usuario con el administrador de la aplicación. Considero importante remarcar que desde la vista de administrador esta funcionalidad la implementé con dos formatos, uno que permite enviar mensajes rápidos sin necesidad de entrar en la conversación completa con un usuario y otro accediendo a ella. Por otra parte, dentro del apartado de alimentación me encargué del punto de nutrientes donde calculé el ICDR de un participante para indicar los nutrientes, diferenciando por categorías, que debe consumir según lo que haya comido en el día y otros datos solicitados. Para desarrollar adecuadamente este aspecto, me informé sobre la geometría nutricional y realicé los cálculos a partir de la fórmula de Harris-Benedict del metabolismo basal.

Además de realizar la implementación más funcional, también me he dedicado a desarrollar todo el css de lo mencionado anteriormente, así como de hacerlo responsive para adaptar la aplicación a teléfonos móviles. De hecho, en este sentido he contribuido con otras partes de la aplicación como el css de mi perfil, las fotos de usuario que antes no tenían, de manera que al editar el perfil se puede seleccionar un avatar, la página de error 404, separar código javascript a distintos archivos o controles de acceso, entre otras cosas que se han ido necesitando. También, en el momento en

que se consideró que Mercedes ya podría empezar a testear la aplicación, junto a mi compañera Almudena investigamos y montamos un servidor en un ordenador que nos facilitó nuestra tutora Sonia, que se encuentra ubicado en la facultad de informática.

Por último, en lo que concierne a la memoria, de lo primero que me encargué fue del apartado del diseño gráfico de la aplicación. Es decir, de los aspectos nombre, logo, estilo, colores, bocetos y prototipos Balsamiq y principios de diseño. En todos estos puntos, me preocupé de poner ejemplos ilustrativos para facilitar la comprensión, sobre todo en lo relativo a los principios de diseño. De hecho, para esto último me documenté sobre distintas teorías psicológicas como los principios de Gestalt. A continuación, me puse a realizar la parte que me correspondía de la especificación de requisitos. Es decir, al igual que mis compañeras, realicé los diagramas de los casos de uso que me tocaban (gestión de grupos, objetivos y chat), así como los casos de uso asociados que sumaron un total de 49 casos de uso. Por otro lado, dentro del apartado de implementación de la aplicación me encargué de los distintos diagramas de actividad relativos a la gestión de grupos, objetivos y chat. De la misma manera, expliqué todo el back-end de los diferentes casos de uso que componen cada una de las funcionalidades mencionadas, además de lo relativo a la parte de alimentación que desarrollé, comentando con detalle el proceso que he seguido en la implementación de los mismos. También, he incluido la parte del front-end correspondiente a dichas funcionalidades. Otro de los puntos que he escrito de la memoria es el diseño funcional de la aplicación. En este punto, me he encargado de comentar la estructura del código y en particular, los apartados del modelo vista controlador, la inyección de dependencias y la seguridad que se han llevado a cabo durante todo el proyecto. Asimismo, he elaborado la sección de resultados y discusión crítica y la guía de uso de la aplicación. Además, junto a mis compañeras, nos hemos encargado de revisar el conjunto del trabajo realizado para detectar posibles mejoras y realizar las correcciones oportunas.

Patricia Plata Barroso

Se puede dividir el trabajo realizado en tres etapas: la estructuración inicial del proyecto, la implementación del código y la escritura de la memoria.

En relación a la planificación inicial del proyecto, llevamos a cabo una reflexión sobre cómo estructurarla, tomando en cuenta las solicitudes realizadas por Mercedes. Posteriormente, colaboré en la elaboración de los bocetos en un dispositivo iPad y diseñé y analicé la estructura de la base de datos. Como parte del proceso de distribución de tareas, se decidieron las áreas de responsabilidad y me correspondía la labor de desarrollar las sesiones, las fichas y la fase de valoración. En este sentido, inicié el desarrollo y la estructuración de las fichas mediante el uso de PHP, pero identifiqué que dicha tecnología no era viable para el proyecto, por lo que se buscó otra alternativa. Fue entonces que decidimos cambiar de tecnología y procedí a documentarme sobre una nueva tecnología denominada SpringBoot, APIRest y JPA.

En cuanto a la implementación del código, en primer lugar, me dediqué a la implementación de diversas funcionalidades en la aplicación. Comencé por la parte de editar perfil y ver perfil, así como a familiarizarme con el trabajo en Springboot. Posteriormente, me enfoqué en realizar la parte de la fase de valoración, donde aseguré que se mostraran los participantes correspondientes a cada coordinador, el filtro del estado, el expediente del participante con las fichas, la fase de valoración con sus formularios, la funcionalidad de mandar una analítica y el enlace que dirige a los materiales grupales, así como el dar de alta o eliminar la cuenta del usuario.

Luego, me concentré en la implementación de la funcionalidad de los materiales individuales y grupales, incluyendo su descarga, pero con la descarga encontré algunos problemas y recibí la ayuda de mi compañera Almudena. Asimismo, implementé el control de accesos en todas las funciones del controller donde faltaban.

Investigué cómo incluir python en la aplicación para poder usar librerías como la de sklearn y hacer un algoritmo de machine learning en la funcionalidad de alimentación. En primer lugar, intenté incluir la tecnología de Flask para conllevaba lanzar dos aplicaciones al mismo tiempo por lo que busqué otra forma. La otra opción que tenía era integrar jyhton, por lo que, investigando, y con el fin de seguir manteniendo la estructura del código integrando el modelo de vista controlador, agregué una factoría para que java pudiese integrar correctamente el script de python, y añadí las anotaciones bean correspondientes al inicio de la aplicación.

Después implementé toda la funcionalidad de alimentación menos la parte del juego y del botón de ver nutrientes que le correspondía a mis compañeras. Además, realicé un archivo sql para añadir a la base de datos los alimentos y recetas correspondientes al programa ALAS.

Además, empecé a cambiar el frontend de la aplicación para que fuese responsive y se adaptase a dispositivos móviles, las vistas que quedaban por adaptar se repartieron entre mis compañeras. También implementé la parte correspondiente al cuaderno, es decir, las diapositivas y las fichas, excepto la de ficha objetivo que le correspondía a Almudena. Y el formulario de terminar sesión que actualizaba el peso perdido y los centímetros de cintura perdidos en cada sesión para que los coordinadores pudiesen hacer un seguimiento de sus participantes en cada sesión. E incluí los cuestionarios en cada sesión para que pudiesen tener una idea de si habían entendido lo realizado en cada sesión.

Asimismo, integré Gmail Api con la ayuda de la guía que se encuentra en Google Developers y creando un correo electrónico para la aplicación con su token. Con ello pude realizar la funcionalidad de recuperar datos. Y, para evitar problemas por el uso del correo electrónico, añadí un formulario de consentimiento en los participantes para que, a través de cookies, se comprobase que el participante había aceptado el uso de su correo electrónico en la aplicación.

Además para asegurarme de que el trabajo de fin de grado se acabase en la fecha establecida y estuviese bien he ayudado en muchas funcionalidades como: eliminar a un participante de un grupo y ver su expediente, añadir en la base de datos de

grupos la figura del coordinador, que el participante vuelva a estar de baja si se sale de su grupo para que todo funcione bien, el filtro de palabras en los chats, ayudar a hacer el chat principal de mi grupo para que funcione, implementar bien la función para ver el porcentaje de progreso en el inicio, hice bien la parte de conversaciones del administrador y le expliqué a mi compañera Inés cómo se implementaría adecuadamente el chat del administrador sin tener que incluir en la base de datos el id del administrador ni añadirlo al modelo de todas las vistas, el eliminar un grupo, actualizar los centímetros de cintura perdidos en el participante, y el contador en progreso de los centímetros de cintura perdidos. En general, han sido mejoras muy puntuales de pequeños errores con el fin de ayudar a mis compañeras como en editar actividad que se pueda editar la foto y validar que sea una imagen, que se actualice el estado de los grupos correctamente, en los scripts que pueden usar varios actores como ver perfil y editar perfil he controlado que en función del actor le dirija a su chat de administrador correspondiente, ayudé a indexar el id en el editar las observaciones de un grupo para que funcionase correctamente y varios detalles que descubrí después de haber revisado varias veces lo implementado en toda la aplicación para asegurarme de que todo quedase bien y les diese tiempo a terminar bien su parte a mis compañeras.

Por último, hice varias funcionalidades comunes como el eliminar usuario y editar usuario desde el punto de vista del administrador, o la vista de cuando un administrador o un coordinador están con el estado de cuenta a baja. Hice un archivo sql con distintos usuarios de prueba para incluirlos como base en la base de datos. Y validé gran parte de los archivos html para que no tuviesen errores. También, quité comentarios que se habían incluido en el código y que no eran útiles, y otras mejoras para que el código fuese más eficiente y limpio. Definí los atributos enums en todas las entidades e hice todos privates y me aseguré de que no hubiesen clases de tipo repository dentro de los controllers para mantener el modelo vista controlador. La funcionalidad de imprimir y la generación de un código aleatorio en la creación de un grupo y un usuario. Además, por petición de Mercedes, quité de la aplicación todos los nombres para trabajar en todo momento con el código de cada usuario y su nickname.

En relación a la escritura de la memoria, he realizado una serie de tareas significativas para su elaboración.

En primer lugar, organicé los bocetos en Balsamiq y realicé más de la mitad de los que faltaban por hacer, cuya mayoría correspondía a las vistas que había implementado yo, y las otras eran comunes, después de la estructuración del proyecto. Asimismo, dividí el resto de los bocetos entre mis compañeras de equipo.

En segundo lugar, trabajé en la estructuración de la memoria con mis compañeras, para hacer la distribución de tareas, y comencé a redactarla. Además, expliqué en detalle cada sección de la memoria y las pautas para su elaboración, con el fin de que todos tuviéramos claro el trabajo que nos correspondía. Redacté la parte de introducción, tanto en inglés como en español, junto con el diagrama de Gantt, la dedicatoria y los agradecimientos. Además de la guía de uso y de instalación. También me encargué de la redacción de la bibliografía, el estado del arte y el anexo

de vídeos de demostración. Aunque cada compañera de equipo agregó los vídeos que le correspondían grabar, en función de cómo habíamos dividido las funcionalidades. Además, realicé la parte de conclusiones y trabajo futuro, tanto en inglés como en español.

En tercer lugar, implementé varias funciones en LaTeX para agregar imágenes y tablas de manera ordenada y en la posición que queríamos, en todas las secciones. En cuanto a la especificación de requisitos, redacté los actores necesarios y realicé los diagramas de casos de uso y las tablas de casos de uso de las funcionalidades que me correspondían. Estas incluían materiales, gestión de participantes, cuaderno y sesiones, y alimentación, sumando un total de setenta casos de uso. Además, dejé a mis compañeras una tabla de ejemplo para unificar el formato y les proporcioné ejemplos de diagramas de casos de uso, controlando que estuvieran correctamente elaborados.

En cuarto lugar, en lo que se refiere a la implementación de la aplicación, realicé los diagramas de actividad de mis funcionalidades y proporcioné ejemplos a mis compañeras de cómo hacerlos. En la parte de backend, redacté los casos de uso correspondientes a las funcionalidades que me correspondían, así como el caso de uso de la figura 6.16 de recuperar datos. También añadí una explicación sobre cómo se implementaron las cookies, para que mis compañeras lo agregaran al caso de uso de las figuras 6.11 y 6.12, es decir, registrarse e iniciar sesión. Dentro de mis funcionalidades, Inés redactó los casos de uso de las figuras 6.100 y 6.101, correspondiente a ver nutrientes y calcular ICDR, y Almudena redactó los casos de uso de las figuras 6.102, 6.105 y 6.103, correspondiente a ver juego, añadir alimentos y comprobar de la funcionalidad de alimentación; el caso de uso de la figura 6.129 de la funcionalidad de cuaderno y sesiones, es decir, llenar la ficha objetivo, y el caso de uso de la figura 6.139 de gestión de participantes, ya que ella implementó todo lo referente a las notificaciones. Y en la parte de frontend, añadí las vistas relacionadas con las funcionalidades que me correspondían.

Por último, en la parte del diseño funcional de la aplicación, redacté el epígrafe de base de datos con el esquema correspondiente y elaboré el diagrama de componentes en el epígrafe de estructura del código.

Bibliografía

*Y así, del mucho leer y del poco dormir, se
le secó el celebro de manera que vino a
perder el juicio.*

Miguel de Cervantes Saavedra

- Spring - <https://spring.io/>
- Organismo Autónomo Madrid Salud - <https://madridsalud.es/>
- GitHub - <https://github.com/>
- HTML - <https://developer.mozilla.org/es/docs/Web/HTML>
- XAMPP - <https://www.apachefriends.org/es/index.html>
- phpMyAdmin - <https://www.phpmyadmin.net/>
- Bootstrap - <https://getbootstrap.com/docs/5.1/getting-started/introduction/>
- Google Developers - <https://developers.google.com/>
- Visual Studio Code - <https://code.visualstudio.com/docs>
- JavaScript - <https://developer.mozilla.org/es/docs/Web/JavaScript>
- Python - <https://www.python.org/doc/>

Apéndice A

Vídeos de demostración

En este apéndice hemos adjuntado un enlace a la carpeta de Google Drive donde hemos subido varios vídeos e imágenes que demuestran el funcionamiento de la aplicación.

El enlace en concreto es el siguiente:

https://drive.google.com/drive/folders/1dOusv9FgMJ-6UcjrgJNmOFj_nOvSQbOk?usp=share_link

Apéndice B

Guía de instalación

A la hora de empezar a utilizar la aplicación de Estepper se presentan dos alternativas:

- Acceder sin instalación de tecnologías
- Acceder con instalación de tecnologías

Acceder sin instalación de tecnologías

En el caso de escoger la primera opción, es necesario entrar en <https://b960-147-96-81-61.ngrok-free.app> desde un dispositivo, ya sea móvil u ordenador. Después, aparecerá un mensaje sobre el que se debe hacer click en el botón de Visit Site. Después de seguir estos pasos, podrá hacer uso de la aplicación libremente o siguiendo la guía de uso del anexo C.

Acceder con instalación de tecnologías

Si se opta por el segundo método, deben seguirse los siguientes pasos para llevar a cabo una correcta instalación y puesta en marcha.

1. Instalar Xampp

Para realizar la instalación de Xampp en el equipo, el primer paso es acceder a su web oficial y descargar el ejecutable adecuado en función del sistema operativo del computador. A continuación, una vez finalizada la descarga, el siguiente paso es ejecutar dicho instalador. En la ventana de instalación de XAMPP, se te mostrará una lista de componentes disponibles para instalar. Se deben seleccionar los que están por defecto que son Apache, MySQL, PHP y phpMyAdmin. Después, elige la carpeta donde deseas instalar XAMPP. Por lo general, se recomienda mantener la ubicación predeterminada. A la hora de seleccionar el puerto escribimos el 3306. Si ya tuviese instalado Xampp o lo instalase en otro puerto, habría que cambiar el archivo application.properties. Más concretamente la línea: `spring.datasource.url=jdbc:mysql://localhost:3306/estepper`. Donde se sustituiría el número 3306 por el número de puerto del que esté

haciendo uso. Finalmente, siga los pasos para completar la instalación e inicie el programa.

2. Instalar Visual Studio Code y descargar extensiones

Visita el sitio web oficial de Visual Studio Code en <https://code.visualstudio.com/> y haz clic en el botón de descarga correspondiente a tu sistema operativo (Windows, macOS o Linux). Una vez que se haya descargado el archivo de instalación de Visual Studio Code, ejecútalo haciendo doble clic en él. Durante el proceso de instalación, siga los pasos por defecto y asegúrese de marcar la opción de ".^regar a PATH", o agregue la aplicación al entorno Path manualmente. Además, para agregar las extensiones necesarias, en primer lugar haz clic en el ícono de la barra lateral izquierda que se asemeja a una serie de cuadros superpuestos, para acceder a ellas. Luego, busque las extensiones de Extension Pack for Java (esta incluye Maven for Java) y Spring Boot Extension Pack (esta incluye Spring Boot Dashboard) e instálelas. Por último, abre la paleta de comandos en Visual Studio Code (Ctrl/Cmd + Shift + P) y busca la opción "Spring Boot: Run Dashboardz" selecciónala. Esto abrirá el Spring Boot Dashboard en la barra lateral de Visual Studio Code.

3. Instalar Git

El proceso de instalación de Git es distinto dependiendo del sistema operativo que se use. Explicaremos cómo se instala en Windows ya que es el sistema operativo más utilizado.

Paso 1: Visita el sitio web oficial de Git en <https://git-scm.com/downloads>.

Paso 2: Descarga el instalador para Windows haciendo clic en el enlace correspondiente a tu sistema (32 bits o 64 bits).

Paso 3: Ejecuta el archivo descargado y sigue las instrucciones del instalador. Asegúrate de seleccionar las opciones predeterminadas a menos que tengas un motivo específico para cambiarlas.

Paso 4: Durante la instalación, puedes elegir el editor de texto que deseas utilizar con Git. Si no tienes preferencias, puedes dejarlo en el valor predeterminado.

Paso 5: Una vez finalizada la instalación, Git estará disponible en tu sistema. Puedes abrir el "Git Bash" desde el menú de inicio para utilizar Git a través de la línea de comandos.

Una vez que hayas instalado Git, puedes configurarlo con tu nombre y dirección de correo electrónico utilizando los siguientes comandos:

```
git config --global user.name "Tu Nombre"
```

```
git config --global user.email "tu@email.com"
```

4. Instalar Java JDK

Al igual que en el caso de la instalación de Git, la instalación es distinta dependiendo del sistema operativo que se use. Explicaremos cómo se instala en Windows.

Paso 1: Visita el sitio web oficial de Oracle Java SE en <https://www.oracle.com/java/technologies/javase-jdk14-downloads.html>.

Paso 2: En la sección "Java SE Development Kit", haz clic en el enlace de descarga correspondiente a tu versión de Windows (por ejemplo, Windows x64 para sistemas de 64 bits).

Paso 3: Acepta los términos de la licencia y luego descarga el instalador del JDK para Windows.

Paso 4: Ejecuta el archivo descargado y sigue las instrucciones del instalador. Asegúrate de seleccionar las opciones predeterminadas.

Después de instalar el JDK, es necesario configurar las variables de entorno JAVA_HOME y PATH.

5. Instalar Maven

Para instalar Maven en un dispositivo con sistema operativo Windows se deben seguir los siguientes pasos:

Paso 1: Visita el sitio web oficial de Apache Maven en <https://maven.apache.org/download.cgi>.

Paso 2: En la sección "Files", busca la última versión estable de Maven y haz clic en el enlace para descargar el archivo ZIP.

Paso 3: Una vez que se haya descargado el archivo ZIP, descomprímelo en una ubicación de tu elección. Por ejemplo, puedes extraerlo en la carpeta C:\Program Files".

Paso 4: Configura la variable de entorno M2_HOME para apuntar al directorio de instalación de Maven. También agrega %M2_HOME%\bin al PATH para poder ejecutar Maven desde cualquier ubicación en la línea de comandos.

6. Instalar Jython

Para instalar Jython, siga los siguientes pasos:

Paso 1: Acceder a la página web oficial de descargas de Jython: <https://www.jython.org/download> .

Paso 2: Acceder al link “Jython Installer”. Asegurarse de que se trata de la versión 2.7.3.

Paso 3: En la terminal de Visual Studio Code navegar hasta donde se encuentre el archivo descargado y poner en la terminal: java -jar NOMBREARCHIVO.jar

Paso 4: Seguir los pasos por defecto de instalación. Asegurarse de instalar el programa en el directorio C://, en una carpeta llamada jython, de forma que se pueda acceder a la librería siguiendo la ruta de C://jython/lib. En el caso de tenerlo instalado en otra ruta, deberá cambiar el archivo de PythonServiceFactory.java sustituyendo las siguientes líneas por su ruta:

```
systemState.path.append(new PyString("C://jython//Lib"));  
systemState.path.append(new PyString("C://jython//Lib//site-packages"));
```

Además, para poder hacer uso de la librería jaydebeapi hay que añadir en el ordenador la variable path: C:\jython\bin y escribir en la consola (cmd) el siguiente comando: jython -m pip install JayDeBeApi

7. Clonar el repositorio de Github en Visual Studio Code

Para ello, se debe abrir una terminal en Visual Studio Code, navegar hasta la carpeta donde quiera introducir la aplicación y escribir el comando: git clone <https://github.com/almunaha/estepper.git>

8. Arrancar Xampp

Abrir la aplicación de Xampp y apretar a los botones de Start de Apache y MySql.

Abre phpMyAdmin en tu navegador web. Puedes acceder a él ingresando la URL correspondiente en la barra de direcciones. Por lo general, es algo como <http://localhost/phpmyadmin> si estás trabajando en tu máquina local. Y, crea una consulta sql para agregar el usuario Estepper a la aplicación, para ello, añade el siguiente código:

```
GRANT ALL PRIVILEGES ON *.* TO 'estepper'@'localhost' IDENTIFIED  
BY PASSWORD '*DBC36BBBAB11CFCE4480B619EE2F4C62DC144C6C'  
WITH GRANT OPTION;
```

9. Ejecutar la aplicación con Spring Boot Dashboard

Antes de ejecutar la aplicación, y con el fin de asegurar el funcionamiento de las funcionalidades que implican el acceso al correo electrónico de la empresa, es necesario que borre el archivo de Stored Credentials para que se genere uno nuevo, este se encuentra dentro de la carpeta tokens.

En el Spring Boot Dashboard, que aparece en el menú lateral izquierdo, haz clic en el botón "Start"(triángulo verde) junto al proyecto que deseas ejecutar. Y se empezará a ejecutar la misma. Podrás ver los registros y mensajes en la terminal integrada de Visual Studio Code.

Al haber borrado el archivo de Stored Credentials, va a ser necesario generar un archivo nuevo para hacer uso de las funcionalidades que implican enviar correos electrónicos. Para ello, simplemente acceda a una de estas funcionalidades, como a recuperar datos, cuyo flujo está explicado en el caso de uso de la siguiente figura 6.16, y pruébelo por primera vez. En este momento le aparecerá un enlace en la terminal de Visual Studio Code, acceda para permitir que la cuenta de Estepper pueda enviar correos electrónicos. Los datos de Gmail de la cuenta del correo electrónico de Estepper son: proyectoestepper@gmail.com y la contraseña es: estepperPAI2023. Una vez hecho esto, se actualizará el archivo de Stored Credentials y podrá acceder a todas las funcionalidades sin problema.

10. Añadir los datos de la base de datos por defecto

Para importar los archivos SQL necesarios para añadir los datos por defecto en phpMyAdmin, sigue estos pasos:

Paso 1: Abre phpMyAdmin en tu navegador web. Puedes acceder a él ingresando la URL correspondiente en la barra de direcciones. Por lo general, es algo como <http://localhost/phpmyadmin> si estás trabajando en tu máquina local.

Paso 2: En la interfaz principal de phpMyAdmin, selecciona la base de datos llamada Estepper. Puedes hacerlo haciendo clic en el nombre de la base de datos en el panel izquierdo.

Paso 3: Una vez que hayas seleccionado la base de datos, ve a la pestaña ‘Importar’ en la parte superior de la pantalla.

Paso 4: En la sección ‘Archivo a importar’, haz clic en el botón ‘Examinar’ y selecciona el archivo SQL que se encuentra en la aplicación dentro de la carpeta resources/static/sql llamado usuariosprueba.sql.

Paso 5: Repite la importación con los archivos datosalimentacion.sql y alimentosintercambio.sql.

11. Acceder a la aplicación

En el navegador web introduzca la siguiente url para acceder a la aplicación:
<http://localhost:7070/>

En este momento se le mostrará la vista de iniciar sesión.

Guía de uso

Desde la vista de iniciar sesión de la aplicación es necesario conocer las credenciales de acceso de los distintos tipos de usuario. Por ello, a continuación se detallan algunos ejemplos con los que se pueden hacer pruebas:

1. Administrador

- Cuenta de ejemplo: código(333) y contraseña (javierpass)

2. Coordinador

- Cuenta de ejemplo: código(222) y contraseña (mercedespass)

3. Participante

- Cuenta de ejemplo (participante de ALTA): código(172) y contraseña (patripass)
- Cuenta de ejemplo (participante de ALTA): código(52654) y contraseña (almupass)
- Cuenta de ejemplo (participante de BAJA): código(98129) y contraseña (inespass)

Introduciendo estos credenciales ya se podría hacer uso libremente de la aplicación y acceder a todas las funcionalidades que se han explicado a lo largo del documento. Si se quisiese probar el conjunto de funcionalidades siguiendo una guía, puede acceder a los vídeos de demostración del anexo A, donde se muestra cómo comprobar todas las implementaciones según cada funcionalidad.

*-¿Qué te parece desto, Sancho? – Dijo Don Quijote –
Bien podrán los encantadores quitarme la ventura,
pero el esfuerzo y el ánimo, será imposible.*

*Segunda parte del Ingenioso Caballero
Don Quijote de la Mancha
Miguel de Cervantes*

*-Buena está – dijo Sancho –; fírmela vuestra merced.
–No es menester firmarla – dijo Don Quijote–,
sino solamente poner mi rúbrica.*

*Primera parte del Ingenioso Caballero
Don Quijote de la Mancha
Miguel de Cervantes*