



Universidad de Valladolid

Escuela de Ingeniería Informática

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática
Mención Ingeniería de Software

SGEmployee: Aplicación iOS para la gestión de las vacaciones laborales.

Autor:

Pablo Carrascal Muñoz

Tutor:

D. Miguel Angel Laguna Serrano

Agradecimientos

A Miguel Ángel Laguna por sus consejos y guías para llevar este TFG a buen puerto.

Al Trío del Aulario por tantas noches de diversión en ese edificio.

A mi familia por haberme apoyado siempre en todo y haber creído en mí.

A Andrea por estar ahí siempre y apoyarme en todo.

Y para acabar, todos los miembros de la empresa Solid Gear y en especial a Gonzalo, Maci, Antonio, Toledo y al grupeto del café.

Resumen

En la actualidad, es innegable que los teléfonos móviles marcan nuestras vidas. Han creado soluciones a necesidades existentes y han creado necesidades a las que daban solución. Estos dispositivos hacen la vida más fácil a las personas, haciendo que tareas que antes requerían mucho tiempo o papeleo ahora puedan hacerse en cuestión de unas pocas pulsaciones sobre una pantalla y unos pocos segundos.

Sin embargo, en la vida laboral de las personas, la movilidad no ha ido más allá de las redes sociales como LinkedIn, para realizar contactos. A menudo para pedir vacaciones los tienen que hablar con los recursos humanos de su empresa para pedir vacaciones y esto puede hacer que la contestación se alargue, que haya que realizar gestiones innecesarias o cualquier cosa que retrase la aceptación o rechazo de sus vacaciones.

Y de esa necesidad de agilizar la gestión de las vacaciones laborales de los trabajadores nace este proyecto, dotando así a una solución ya existente de la pata que le faltaba, la aplicación para el sistema operativo iOS.

La aplicación desarrollada, *SGEmployee*, no solo permite pedir vacaciones a los empleados, también permite ver su historial de vacaciones, ver cuantos días les quedan y cuantos llevan gastados, gestionar su perfil corporativo, recibir notificaciones y avisos, ver una lista de los compañeros de empresa e interactuar con ellos y sus redes sociales, etc.

En cuanto a la arquitectura elegida para la aplicación, se ha elegido una variante de la arquitectura MVC (Modelo - Vista - Controlador) optimizada para iOS aprovechando las bondades del sistema operativo para ofrecer la mejor experiencia al usuario.

En cuanto a la metodología utilizada para el desarrollo del proyecto, se ha seguido la metodología ágil más famosa, Scrum, y gracias a su flexibilidad ha permitido que este proyecto sea finalizado en un tiempo razonable y minimizando los costes y riesgos.

En esta memoria se encuentra toda la documentación necesaria para comprender el proyecto desarrollado, desde los requisitos, análisis, diseño, implementación y pruebas, hasta el manual de usuario y otros aspectos relevantes.

Índice de contenidos

1. Introducción	1
1.1. Contexto y motivación	1
1.2. Objetivos	1
1.3. Metodología utilizada	2
1.4. Resumen del contenido de la memoria	2
2. Entorno tecnológico	4
2.1. Herramientas Utilizadas	4
2.2. Entorno de desarrollo	4
3. Plan de Desarrollo de Software	6
3.1. Introducción	6
3.1.1. Propósito	6
3.1.2. Alcance	6
3.1.3. Resumen del plan de desarrollo	6
3.2. Visión general del proyecto	7
3.2.1. Objetivos	7
3.2.2. Suposiciones y/o restricciones	7
3.2.3. Características del proyecto	8
3.2.4. Elección de la metodología de desarrollo de software	8
3.2.5. Elección del tipo de desarrollo a utilizar	9
3.2.6. Entregables del proyecto	9
3.3. Organización del proyecto	10
3.3.1. Interfaces Externas	10
3.3.2. Estructura Interna	10
3.3.3. Roles del proyecto	10
3.3.4. Reuniones, eventos y planes operativos	11
3.4. Planificación del proyecto	12
3.4.1. Estimaciones del proyecto	12
3.4.1.1. Estimación temporal	12
3.4.1.2. Estimación de costes	13
3.4.1.3. Desviación real del proyecto	13
3.5. Seguimiento del proyecto	14
4. Plan de Gestión de Riesgos	15
4.1. Introducción	15
4.1.1. Alcance	15

4.2. Gestión del riesgo	16
4.3. Control de riesgos	20
5. Requisitos	21
5.1. Los requisitos en el desarrollo ágil	21
5.2. Requisitos funcionales	22
5.3. Requisitos no funcionales	23
5.4. Requisitos de información	23
6. Análisis	24
6.1. Modelo de Casos de Uso	24
6.1.1. Actores	24
6.1.2. Diagrama de casos de uso	25
6.1.3. Especificación de casos de uso	26
6.1.4. Realización de los casos de uso en análisis	36
6.1.4.1. Modelo de dominio	36
6.1.5. Descripción de las clases del modelo de dominio.	36
6.1.5.1. Empleado	36
6.1.5.2. Compañero	37
6.1.5.3. EmpleadoPrincipal	37
6.1.5.4. MaxHolidays	37
6.1.5.5. Notice	37
6.1.5.6. MessageNotice	37
6.1.5.7. MaxHolidaysNotice	38
6.1.5.8. HolidayChangeNotice	38
6.1.5.9. Holiday	38
6.1.6. Diagramas de actividad	39
6.1.6.1. CU01: Identificarse	39
6.1.6.2. CU02: Ver Perfil	40
6.1.6.3. CU03: Modificar Perfil	41
6.1.6.4. CU04: Ver vacaciones restantes	42
6.1.6.5. CU05: Pedir vacaciones	43
6.1.6.6. CU06: Modificar estado vacaciones	44
6.1.6.7. CU07: Ver total avisos	45
6.1.6.8. CU08: Recibir notificaciones	46
6.1.6.9. CU09: Ver <i>Twitter</i> corporativo.	47
6.1.6.10. CU10: Ver compañeros de trabajo.	48
6.1.6.11. CU11: Buscar compañeros de trabajo.	49
7. Diseño	50
7.1. Diseño de la arquitectura	50
7.1.1. Patrón arquitectónico utilizado	50
7.1.2. Arquitectura general	51
7.1.3. Descripción modular	53
7.1.3.1. Descripción de la descomposición modular	53
7.1.4. Diseño de los módulos o subsistemas	53
7.1.4.1. Diseño detallado del módulo <i>Controller</i>	54

7.1.4.2.	Diseño detallado del módulo <i>Sync</i>	55
7.1.4.3.	Diseño detallado del módulo <i>Network</i>	55
7.1.4.4.	Diseño detallado del módulo <i>Data</i>	56
7.1.5.	Relación entre módulos	57
7.1.5.1.	CU01: Identificarse	57
7.1.5.2.	CU02: Ver perfil	58
7.1.5.3.	CU03: Modificar perfil	59
7.1.5.4.	CU04: Ver vacaciones restantes	60
7.1.5.5.	CU05: Pedir Vacaciones	61
7.1.5.6.	CU06: Modificar estado vacaciones	62
7.1.5.7.	CU07: Ver total avisos	63
7.1.5.8.	CU10: Ver compañeros de trabajo	64
7.1.6.	Realización en diseño de los Casos de Uso	65
7.1.6.1.	Detalle de SaveUserInDataBase	65
7.1.6.2.	Detalle de GetUserFromDataBase	65
7.1.6.3.	CU01: Identificarse	66
7.1.6.4.	CU02: Ver perfil	67
7.1.6.5.	Detalle de updateUserInfoFromServer	68
7.1.6.6.	CU03: Editar perfil	69
7.1.6.7.	CU04: Ver vacaciones	70
7.1.6.8.	CU05: Pedir vacaciones	71
7.1.6.9.	CU05: Pedir vacaciones	72
7.1.6.10.	CU06: Modificar vacaciones	73
7.1.6.11.	CU010: Ver compañeros de trabajo	74
7.1.6.12.	Detalle de getTeamMatesFromServer	75
7.2.	Modelo de Datos	76
7.2.1.	Esquema relacional de la base de datos	76
7.3.	Diseño de la interfaz de usuario	76
8.	Implementación y Pruebas	77
8.1.	Implementación	77
8.2.	Pruebas	78
8.2.1.	Introducción	78
8.2.2.	Metodología	78
8.2.3.	Errores obtenidos en el Cross QA	78
9.	Conclusiones	81
9.1.	Objetivos alcanzados	81
9.2.	Valoración personal	82
9.3.	Lineas Futuras	82
	Bibliografía	82
	Apéndices	84
A.	Glosario de Términos	85

B. Plan de Seguimiento	86
B.1. Introducción	86
B.1.1. Propósito	87
B.1.2. Alcance	87
B.2. Sprint 0: 20 de febrero - 1 de marzo	88
B.2.1. <i>User Stories</i> a desarrollar	88
B.2.2. US00 - Como Equipo quiero tener disponibles tanto los entornos de desarrollo de software como de documentación listos para empezar el proyecto.	88
B.2.2.1. Descripción	88
B.2.2.2. Puntos	88
B.2.2.3. Tareas	88
B.2.2.4. Comentarios	89
B.2.3. US01 - Como <i>Product Owner</i> quiero disponer de la documentación inicial del proyecto para tener una visión global del mismo.	89
B.2.3.1. Descripción	89
B.2.3.2. Puntos	89
B.2.3.3. Tareas	89
B.2.3.4. Comentarios	89
B.2.4. Riesgos	89
B.3. Sprint 1: 2 de marzo - 8 de marzo	90
B.3.1. <i>User Stories</i> a desarrollar	90
B.3.2. US02 - Como <i>Product Owner</i> quiero tener un modelo de dominio de la aplicación.	90
B.3.2.1. Descripción	90
B.3.2.2. Puntos	90
B.3.2.3. Tareas	90
B.3.2.4. Comentarios	90
B.3.3. US03 - Como Usuario quiero poder identificarme en la aplicación con mis credenciales corporativas.	91
B.3.3.1. Descripción	91
B.3.3.2. Puntos	91
B.3.3.3. Tareas	91
B.3.3.4. Comentarios	91
B.3.4. Riesgos	91
B.4. Sprint 2: 9 de marzo - 20 de marzo	92
B.4.1. <i>User Stories</i> a desarrollar	92
B.4.2. US04 - Como Usuario quiero poder ver una lista de mis compañeros de trabajo, poder ver su foto de perfil, ver su <i>Twitter</i> o <i>LinkedIn</i> y llamar a su teléfono.	92
B.4.2.1. Descripción	92
B.4.2.2. Puntos	92
B.4.2.3. Tareas	92
B.4.2.4. Comentarios	93
B.4.3. Riesgos	93
B.5. Sprint 3: 21 de marzo - 31 de marzo	94
B.5.1. <i>User Stories</i> a desarrollar	94

B.5.2.	US05 - Como Usuario quiero poder ver y editar mi perfil corporativo desde la aplicación	94
B.5.2.1.	Descripción	94
B.5.2.2.	Puntos	94
B.5.2.3.	Tareas	94
B.5.2.4.	Comentarios	95
B.5.3.	Riesgos	95
B.6.	Sprint 4: 1 de abril - 6 de abril	96
B.6.1.	<i>User Stories</i> a desarrollar	96
B.6.2.	US06 - Como <i>Product Owner</i> quiero tener una primera versión del modelo de la arquitectura del sistema y de su diseño, así como de los patrones que son utilizados.	96
B.6.2.1.	Descripción	96
B.6.2.2.	Puntos	96
B.6.2.3.	Tareas	96
B.6.2.4.	Comentarios	96
B.6.3.	Riesgos	97
B.7.	Sprint 5: 7 de abril - 19 de abril	98
B.7.1.	<i>User Stories</i> a desarrollar	98
B.7.2.	US09 - Como Usuario quiero poder recibir notificaciones cuando alguna de mis vacaciones cambie de estado	98
B.7.2.1.	Descripción	98
B.7.2.2.	Puntos	98
B.7.2.3.	Tareas	98
B.7.2.4.	Comentarios	99
B.7.3.	US10 - Como Usuario quiero poder recibir notificaciones que el administrador decida mandarme	99
B.7.3.1.	Descripción	99
B.7.3.2.	Puntos	99
B.7.3.3.	Tareas	99
B.7.3.4.	Comentarios	99
B.7.4.	US11 - Como Usuario quiero poder ver una lista con las últimas notificaciones que he recibido	99
B.7.4.1.	Descripción	99
B.7.4.2.	Puntos	100
B.7.4.3.	Tareas	100
B.7.5.	Riesgos	100
B.8.	Sprint 6: 20 de abril - 5 de mayo	101
B.8.1.	<i>User Stories</i> a desarrollar	101
B.8.2.	US08 - Como Usuario quiero poder pedir vacaciones para poder disfrutar de mis derechos laborales.	101
B.8.2.1.	Descripción	101
B.8.2.2.	Puntos	101
B.8.2.3.	Tareas	101
B.8.2.4.	Comentarios	102
B.8.3.	Riesgos	102
B.9.	Sprint 7: 08 de mayo - 15 de mayo	103

B.9.1. <i>User Stories</i> a desarrollar	103
B.9.2. US07 - Como Usuario quiero poder ver los días de vacaciones de los que dispongo y cuantos me quedan por usar	103
B.9.2.1. Descripción	103
B.9.2.2. Puntos	103
B.9.2.3. Tareas	103
B.9.2.4. Comentarios	103
B.9.3. Riesgos	104
B.10.Sprint 8: 16 de mayo - 24 de mayo	105
B.10.1. <i>User Stories</i> a desarrollar	105
B.10.2. US12 - Como Usuario quiero poder ver información sobre la empresa, como por ejemplo su dirección, CIF, etc.	105
B.10.2.1. Descripción	105
B.10.2.2. Puntos	105
B.10.2.3. Tareas	105
B.10.2.4. Comentarios	105
B.10.3. Riesgos	105
B.11.Sprint 9: 25 de mayo - 31 de mayo	106
B.11.1. <i>User Stories</i> a desarrollar	106
B.11.2. US13 - Como Usuario quiero tener una versión de la aplicación instalada en mi dispositivo	106
B.11.2.1. Descripción	106
B.11.2.2. Puntos	106
B.11.2.3. Tareas	106
B.11.2.4. Comentarios	106
B.11.3. Riesgos	107
B.12.Sprint 10: 01 de junio - 7 de junio	108
B.12.1. <i>User Stories</i> a desarrollar	108
B.12.2. US14 - Como <i>Product Owner</i> quiero tener el plan de desarrollo de software completo.	108
B.12.2.1. Descripción	108
B.12.2.2. Puntos	108
B.12.2.3. Tareas	108
B.12.3. Riesgos	108
B.13.Sprint 11: 08 de junio - 15 de junio	109
B.13.1. <i>User Stories</i> a desarrollar	109
B.13.2. US15 - Como <i>Product Owner</i> quiero tener el modelo de dominio casi completo.	109
B.13.2.1. Descripción	109
B.13.2.2. Puntos	109
B.13.2.3. Tareas	109
B.13.2.4. Comentarios	109
B.13.3. US16 - Como <i>Product Owner</i> quiero tener el diseño de software del sistema casi completo.	110
B.13.3.1. Descripción	110
B.13.3.2. Puntos	110
B.13.3.3. Tareas	110

B.13.3.4. Comentarios	110
B.13.4. Riesgos	110
B.14.Sprint 12: 19 de junio - 05 de julio	111
B.14.1. <i>User Stories</i> a desarrollar	111
B.14.2. US17 - Como <i>Product Owner</i> tener la memoria del proyecto acabada	111
B.14.2.1. Descripción	111
B.14.2.2. Puntos	111
B.14.2.3. Tareas	111
B.14.2.4. Comentarios	111
C. Manual de Usuario	112
C.1. Introducción	112
C.2. Funcionalidades	112
C.2.1. Identificarse	112
C.2.1.1. Paso 1. Introducir las credenciales y pulsar en el botón de identificarse.	112
C.2.1.2. Paso 2. Dependiendo de si las credenciales son correctas, se procede a una pantalla u otra.	112
C.2.2. Abrir Y cerrar el menú	113
C.2.2.1. Paso 1. Pulsar sobre el botón menú para abrir o cerrar el menú	113
C.2.3. Ver Perfil	114
C.2.3.1. Paso 1. Pulsar sobre el botón perfil en el menú y se abrirá el perfil.	114
C.2.4. Editar Perfil	114
C.2.4.1. Paso 1. Se parte de la pantalla perfil y se pulsa en el botón Editar Perfil	114
C.2.4.2. Paso 2. Si pulsas sobre el botón cancelar, los cambios no se guardan y se vuelve a la pantalla de perfil.	115
C.2.4.3. Paso 3. Si realizas algún cambio y pulsas sobre el botón hecho, los cambios se guardan y se vuelve a la pantalla perfil.	115
C.2.5. Cambiar ajustes	116
C.2.5.1. Paso 1. Pulsar sobre el botón ajustes en el menú y se abrirán los ajustes	116
C.2.6. Ver Compañeros de trabajo	117
C.2.6.1. Paso 1. Pulsar sobre el botón equipo en el menú y se abrirá la pantalla de equipo	117
C.2.7. Ver datos de la empresa	118
C.2.7.1. Paso 1. Pulsar sobre el botón compañía en el menú y se abrirá la pantalla de datos sobre la compañía.	118
C.2.8. Ver vacaciones	119
C.2.8.1. Paso 1. Pulsar sobre el botón vacaciones en el menú y se abrirá la pantalla de ver vacaciones.	119
C.2.9. Pedir vacaciones	120
C.2.9.1. Paso 1. Pulsar sobre el botón (+) pedir vacaciones.	120
C.2.9.2. Paso 2. Introducir fechas para las vacaciones y pulsar hecho.	120
C.2.10. Cancelar vacaciones	121
C.2.10.1. Paso 1. Deslizar a la derecha unas vacaciones. Estas han de estar en el estado pendiente de aprobación o aprobadas.	121

C.2.10.2. Paso 2. Las vacaciones habrán cambiado de estado.	121
C.2.11. Ver avisos	122
C.2.11.1. Paso 1. Pulsar sobre el botón avisos en el menú y se abrirá la pantalla de avisos.	122
D. Manual de instalación	123

Índice de figuras

3.1. Estructura interna del proyecto	10
6.1. Diagrama de casos de uso de la aplicación.	25
6.2. Modelo de dominio de la aplicación.	36
6.3. Diagrama de actividad del CU01.	39
6.4. Diagrama de actividad del CU02.	40
6.5. Diagrama de actividad del CU03.	41
6.6. Diagrama de actividad del CU04.	42
6.7. Diagrama de actividad del CU05.	43
6.8. Diagrama de actividad del CU6.	44
6.9. Diagrama de actividad del CU07.	45
6.10. Diagrama de actividad del CU08.	46
6.11. Diagrama de actividad del CU09.	47
6.12. Diagrama de actividad del CU10.	48
6.13. Diagrama de actividad del CU11.	49
7.1. Descripción del patrón MVC	50
7.2. Arquitectura general de la aplicación	51
7.3. Descomposición modular de la aplicación	53
7.4. Diseño detallado del módulo <i>Controller</i>	54
7.5. Diseño detallado del módulo <i>Sync</i>	55
7.6. Diseño detallado del módulo <i>Network</i>	55
7.7. Diseño detallado del módulo <i>Data</i>	56
7.8. Relación entre módulos para el CU01	57
7.9. Relación entre módulos para el CU02	58
7.10. Relación entre módulos para el CU03	59
7.11. Relación entre módulos para el CU04	60
7.12. Relación entre módulos para el CU05	61
7.13. Relación entre módulos para el CU06	62
7.14. Relación entre módulos para el CU07	63
7.15. Relación entre módulos para el CU10	64
7.16. Detalla en diseño de SaveUserInDataBase	65
7.17. Detalla en diseño de GetUserFromDataBase	65
7.18. Relación en diseño del CU01	66
7.19. Relación en diseño del CU02	67
7.20. Detalla en diseño de updateUserInfoFromServer	68
7.21. Relación en diseño del CU03	69

7.22. Relación en diseño del CU04	70
7.23. Relación en diseño del CU05	71
7.24. Relación en diseño del CU05	72
7.25. Relación en diseño del CU06	73
7.26. Relación en diseño del CU10	74
7.27. Detalla en diseño de getTeamMatesFromServer	75
7.28. Esquema relacional de la base de datos	76
 B.1. Explicación gráfica del flujo de Scrum	 86
C.1. Pantalla identificación.	112
C.2. Identificándose.	112
C.3. Identificación fallida.	113
C.4. Pantalla principal.	113
C.5. Identificación fallida.	113
C.6. Pantalla del perfil.	113
C.7. Menú abierto.	114
C.8. Pantalla del perfil.	114
C.9. Pantalla de perfil.	114
C.10. Pantalla de editar perfil.	114
C.11. Cancelar editar el perfil.	115
C.12. Pantalla de perfil.	115
C.13. Editar el perfil	115
C.14. Pantalla de perfil.	115
C.15. Pantalla de ajustes.	116
C.16. Lista de compañeros.	117
C.17. Compañeros filtrados.	117
C.18. Datos de la empresa	118
C.19. Dirección de la empresa	118
C.20. Pantalla de ver vacaciones.	119
C.21. Botón pedir vacaciones.	120
C.22. Pantalla de pedir vacaciones.	120
C.23. Pantalla de pedir vacaciones.	120
C.24. Enviando datos al servidor.	120
C.25. Vacaciones.	121
C.26. Eliminar vacaciones.	121
C.27. Las vacaciones habrán cambiado de estado.	121
C.28. Pantalla de avisos.	122

Lista de Tablas

2.1. Descripción del portatil utilizado	4
2.2. Descripción del equipo de sobremesa utilizado	5
3.1. Roles del proyecto y personas que desempeñará cada uno	11
3.2. Estimación en coste y tiempo de los <i>sprints</i> del proyecto.	12
3.3. Desviación en coste y tiempo de los <i>sprints</i> del proyecto.	14
4.1. Riesgo 01	16
4.2. Riesgo 02	16
4.3. Riesgo 03	17
4.4. Riesgo 04	17
4.5. Riesgo 05	18
4.6. Riesgo 06	18
4.7. Riesgo 07	19
4.8. Riesgo 08	19
4.9. Riesgo 09	20
5.1. Requisitos funcionales del proyecto.	22
5.2. Requisitos no funcionales del proyecto.	23
5.3. Requisitos de información del proyecto.	23
6.1. Descripción del CU01	26
6.2. Descripción del CU02	27
6.3. Descripción del CU03	28
6.4. Descripción del CU04	29
6.5. Descripción del CU05	30
6.6. Descripción del CU06	31
6.7. Descripción del CU07	32
6.8. Descripción del CU08	33
6.9. Descripción del CU09	33
6.10. Descripción del CU10	34
6.11. Descripción del CU11	35
8.1. Descripción de un error encontrado durante el X-QA	78
8.2. Descripción de un error encontrado durante el X-QA	79
8.3. Descripción de un error encontrado durante el X-QA	79
8.4. Descripción de un error encontrado durante el X-QA	79

8.5. Descripción de un error encontrado durante el X-QA	79
8.6. Descripción de un error encontrado durante el X-QA	80
8.7. Descripción de un error encontrado durante el X-QA	80
8.8. Descripción de un error encontrado durante el X-QA	80
8.9. Descripción de un error encontrado durante el X-QA	80
B.1. <i>User Stories</i> para el Sprint 0	88
B.2. <i>User Stories</i> para el Sprint 1	90
B.3. <i>User Stories</i> para el Sprint 2	92
B.4. <i>User Stories</i> para el Sprint 3	94
B.5. <i>User Stories</i> para el Sprint 4	96
B.6. <i>User Stories</i> para el Sprint 5	98
B.7. <i>User Stories</i> para el Sprint 6	101
B.8. <i>User Stories</i> para el Sprint 7	103
B.9. <i>User Stories</i> para el Sprint 8	105
B.10. <i>User Stories</i> para el Sprint 9	106
B.11. <i>User Stories</i> para el Sprint 10	108
B.12. <i>User Stories</i> para el Sprint 11	109
B.13. <i>User Stories</i> para el Sprint 12	111

Capítulo 1

Introducción

1.1. Contexto y motivación

Uno de los momentos más dulces para un trabajador en su vida laboral son las vacaciones, ese periodo de tiempo en el que se despreocupa de sus quehaceres laborales y se dedica a viajar, a sus aficiones, etc.

Dependiendo de la empresa, el proceso para pedir vacaciones a los superiores puede llevar a confusiones o mal entendidos lo cual no beneficia a ninguna de las dos partes.

Con la aparición y popularidad de los teléfonos móviles inteligentes o *smartphones* se abren un mundo de posibilidades para desarrollar aplicaciones que faciliten la vida a sus usuarios.

Es aquí donde se detecta la necesidad de crear un sistema moderno para la gestión de las vacaciones, lejos de las hojas de cálculo en las que se apuntan fechas de vacaciones de los empleados o apuntarlas directamente sobre un calendario. Por todo esto y teniendo en cuenta que ya existían soluciones en la parte de servidor y para el sistema operativo Android, se propuso este TFG para hacer la versión para el sistema iOS para así completar la oferta de aplicaciones para los sistemas más conocidos actualmente en el sector de la movilidad.

1.2. Objetivos

El objetivo principal de este TFG (Trabajo de Fin de Grado) consiste en el desarrollo de una aplicación móvil para el sistema operativo iOS la cual permita gestionar tanto las vacaciones de los empleados de una empresa como avisos, timeline de Twitter, perfil en la compañía, etc. Este TFG ha sido realizado en el ámbito de la empresa.

A partir de este objetivo principal, surgen los objetivos secundarios que se detallan a continuación:

- Aprender el lenguaje de programación Swift[1] en su versión más actual (En la fecha de la redacción de esta memoria esa versión corresponde a Swift 3.2)
- Aprender a utilizar el framework Cocoa Touch así como los demás frameworks que proporciona el sistema operativo iOS
- Aprender a utilizar con soltura el entorno de desarrollo que proporciona Apple, vease Xcode, Simulator...
- Aprender a utilizar como cliente una API de tipo REST[2] y a manejar sus diferentes endpoints.
- Familiarizarse con la metodología de desarrollo ágil SCRUM[3].

- Mejorar la capacidad de desarrollo de un proyecto software en base a una serie de requisitos dados por un cliente.
- Planificar un proyecto software y aprender a manejar los posibles riesgos, impedimentos e imprevistos que pudieran surgir.
- Ser capaz de elaborar una memoria detallada de un proyecto software.

1.3. Metodología utilizada

Ya que las metodologías ágiles están al alza como metodologías de desarrollo de proyectos software se ha decidido utilizar Scrum como metodología ágil a seguir.

Este modelo en concreto se caracteriza por, entre otras, seguir las siguientes características[4]:

- Permite gran flexibilidad ante cambios
- Adopta un modelo de desarrollo incremental en lugar de la planificación y ejecución completa del producto.
- Cada iteración tiene que proporcionar un resultado completo, un incremento de producto final
- Implícitamente hace que la gestión del riesgo forme parte del ciclo de vida del proyecto.
- Permite al mismo tiempo ejecutar varias fases o etapas del desarrollo

1.4. Resumen del contenido de la memoria

En este apartado se describen brevemente cada uno de los capítulos y secciones de los que se compone esta memoria para así ofrecer un pequeño resumen de cada uno de los mismos:

- **Capítulo 1. Introducción:** se exponen los objetivos, metodología utilizada y se presenta un pequeño resumen.
- **Capítulo 2. Entorno tecnológico:** se detallan los equipos y herramientas que se utilizan a lo largo de todo el proyecto.
- **Capítulo 3. Plan de Desarrollo de Software:** se expone una visión global del proyecto, entre otras cosas se detallan la planificación, los roles que participan, etc.
- **Capítulo 4. Plan de Gestión de Riesgos:** se detallan y analizan los posibles riesgos que se pueden dar en el proyecto y como actuar en cada situación.
- **Capítulo 5. Requisitos:** se detallan los requisitos funcionales, no funcionales y de información.
- **Capítulo 6. Análisis:** se exponen los actores del sistema, los casos de uso, el modelo de dominio y los diagramas de actividad.
- **Capítulo 7. Diseño:** se presenta y se detalla la arquitectura general de la aplicación, la descomposición modular del sistema, el modelo de datos y la realización en diseño de los casos de uso más significativos de la aplicación.

- **Capítulo 8. Implementación y Pruebas:** se detalla brevemente la implementación y las pruebas realizadas, así como los errores encontrados en dichas pruebas.
- **Capítulo 9. Conclusiones:** se exponen los objetivos alcanzados, la valoración personal del alumno y se presentan las líneas futuras del proyecto.
- **Bibliografía :** se muestra la documentación utilizada durante todo el proyecto.
- **Anexos:** glosario de términos, manual de usuario y plan de desarrollo de software.

Capítulo 2

Entorno tecnológico

2.1. Herramientas Utilizadas

Las herramientas utilizadas durante el desarrollo de este proyecto han sido:

- **Xcode**: herramienta de desarrollo integrado (IDE) utilizada para desarrollar la aplicación para iOS
- **Simulator**: simulador de iOS integrado con Xcode
- **Gitlab**[5]: plataforma de desarrollo colaborativo de software y de control de versiones
- **Git**: Sistema de control de versiones utilizado, en concreto, se utilizará el sistema de *branching* *GitFlow*[6]
- **Jira**[7]: Sistema de gestión de proyectos ágiles
- **Overleaf**[8]: editor online de Latex utilizado para la redacción de esta memoria

2.2. Entorno de desarrollo

El entorno de trabajo durante el desarrollo ha sido compartido entre un ordenador portátil Toshiba (Tabla 2.1) y un Mac mini (Tabla 2.2).

A continuación se detallan sus especificaciones:

Equipo 1	
Marca	Toshiba
Modelo	Satellite L755-1J9
Hardware	
Procesador	Intel® Core™ i5-2450M @ 2.50 GHz
Memoria	8 GB de SDRAM DDR3 1333 MHz
Disco duro	Sandisk 120 GB SSD Plus
Targeta gráfica	Nvidia Geforce 520M
Memoria gráfica	1 GB GDDR3
Pantalla	1366 x 768 15,5"
Software	
OS	OS X El Capitan 10.11.6

Tabla 2.1: Descripción del portátil utilizado

Equipo 2	
Marca	Apple
Modelo	Mac Mini (2014)
Hardware	
Procesador	Intel® Core™ i5
Memoria	8 GB de SDRAM LPDDR3 a 1.600 MHz
Disco duro	Serial ATA de 1 TB a 5.400 rpm
Targeta gráfica	Intel HD
Memoria gráfica	
Pantalla	2x Asus 1920x1080 21,5"
Software	
OS	OS X El Capitan 10.11.6

Tabla 2.2: Descripción del equipo de sobremesa utilizado

A parte de los dispositivos utilizados para desarrollar la aplicación y escribir la documentación y memoria se han utilizado varios dispositivos móviles para probar en un dispositivo real la misma.

Entre ellos se encuentran:

- **iPhone 5s:** iOS 10.3
- **iPhone 6:** iOS 10.3
- **iPhone 6 Plus:** iOS 10.3
- **iPhone 6s Plus:** iOS 10.3

Viendo la selección de dispositivos de prueba usados se puede decir que la aplicación se probó en toda la variedad de tamaños de pantalla que a fecha de escritura de esta memoria Apple tiene a la venta, lo que da cierta seguridad, sobre todo, en el área de la interfaz de usuario.

Capítulo 3

Plan de Desarrollo de Software

3.1. Introducción

En esta sección de la memoria se detallará el desarrollo del proyecto, dando así una visión global del mismo.

Como ya se ha mencionado con anterioridad, este proyecto seguirá una metodología ágil, más en concreto *Scrum*. A lo largo de esta sección se dará cuenta del porqué de esta decisión y sus consecuencias a la hora del desarrollo del proyecto.

3.1.1. Propósito

El propósito de todo plan de desarrollo de software, y de este en concreto, es proporcionar de forma detallada información para poder realizar un control y seguimiento sobre el proyecto a realizar.

Como usuarios de este documento tendremos:

- El alumno: se encargará de analizar, diseñar, planificar y desarrollar el proyecto.
- El tutor del TFG: será el encargado de revisar la documentación e implementación del proyecto así como su avance, aportando nuevas ideas o posibles soluciones a los problemas que surjan.

3.1.2. Alcance

En esta memoria se detalla una visión general del proyecto, así como sus participantes y los roles que estos desempeñan en él. También se desglosa un plan de trabajo en iteraciones (también llamadas *Sprints* en *Scrum*) y se muestra más información que pudiera resultar relevante para el desarrollo de este proyecto.

3.1.3. Resumen del plan de desarrollo

El plan de desarrollo se divide principalmente en:

- **Visión general del proyecto a desarrollar**, que a su vez se subdivide en:
 - Objetivos del proyecto
 - Suposiciones y/o restricciones
 - Características del proyecto

- Elección de la metodología de desarrollo
- Entregables del proyecto
- **Organización del proyecto:** la cual incluye tanto cómo se va a estructurar el mismo, los roles de cada miembro y sus responsabilidades.
- **Planificación del proyecto:** incluye las estimaciones sobre las fechas, entregables, desviaciones e hitos.
- **Seguimiento del proyecto:** referencia al plan de seguimiento del proyecto

3.2. Visión general del proyecto

3.2.1. Objetivos

El objetivo principal es desarrollar la aplicación SGEmployee, en su versión para iOS, la cual facilitará a los empleados de una empresa la gestión de sus vacaciones laborales a través de sus dispositivos móviles.

Además de la gestión de las vacaciones, SGEmployee ofrecerá las siguientes funcionalidades:

- **Gestión del perfil corporativo:** Gestiona tanto la foto de perfil como la fecha de cumpleaños, las redes sociales y el teléfono de cada empleado.
- **Obtener información básica de la empresa:** Así puede conocerse la localización de la empresa, su CIF, su dirección y número de teléfono
- **Obtener el *TimeLine* de la cuenta de Twitter de la empresa:** Para estar al tanto de las ultimas noticias sobre la empresa
- **Recibir notificaciones que el administrador desee comunicar:** Recibir cualquier notificación que el administrador del sistema desee comunicar a sus empleados

3.2.2. Suposiciones y/o restricciones

El proyecto debe cumplir con las siguiente restricciones:

1. Restricciones de presupuesto:

- La aplicación ha de desarrollarse con bibliotecas y *frameworks* que sean gratuitos y además su licencia no ha de establecer ni suponer ningún problema de derechos.

2. Restricciones de recursos:

- El equipo de trabajo estará formado por 2 integrantes: el alumno y el tutor de TFG de la universidad.

3. Restricciones de la aplicación:

- La aplicación debe ser desarrollada para el sistema operativo iOS. En concreto debe soportar la versión iOS 9.0 y superiores.
- La aplicación debe consumir y actualizar los datos de una API de tipo REST.

3.2.3. Características del proyecto

A continuación definiremos las características generales de este proyecto para así, una vez elicítadas, poder tomar decisiones acorde a estas características:

- Es un proyecto de duración corta/media, para ser mas exactos debería de aproximarse a 600 horas.
- El proyecto será realizado por una persona, el alumno siendo supervisado por el tutor de TFG.
- Entrega de una documentación exhaustiva y precisa que incluya tanto el análisis y diseño de la aplicación, como la planificación, gestión de riesgos y los demás entregables del proyecto.
- Es importante la gestión de riesgos ya que no se tiene familiaridad con la tecnología y además se utiliza una API externa de tipo REST que puede ser propensa a algún fallo ajeno a este proyecto.
- Los requisitos pueden ser susceptibles de cambios, ampliación de su descripción y/o variación de su prioridad

3.2.4. Elección de la metodología de desarrollo de software

Son varias las metodologías de desarrollo de software y los modelos de proceso que en la actualidad más se utilizan. Desde el modelo en cascada[9] hasta las metodologías ágiles, pasando por el proceso unificado[10].

Como ya se ha mencionado con anterioridad en esta memoria, se ha decidido que la metodología a usar para este proyecto *Scrum*. A continuación se detallará un razonamiento del por qué de esta elección, a parte de por lo ya mencionado antes:

- **Flexibilidad:** se asume que los cambios son parte natural del proyecto y así, en cada iteración, se permiten adecuar o cambiar tanto los requisitos como su funcionalidad. También permite adecuar el desarrollo a la velocidad real que este lleva, permitiendo así hacer pruebas de concepto que servirán para mejorar la siguiente iteración.
- **Rápida mitigación de riesgos:** al tratarse de un desarrollo iterativo e incremental se han de gestionar los posibles riesgos que puedan surgir a lo largo de una iteración promoviendo así su mitigación de manera anticipada.
- **Fácil de integrar:** es una metodología sencilla y fácil de integrar en los proyectos software. Su sencillez hace que el tiempo de aprendizaje sea corto haciéndolo ideal para un proyecto con un tiempo ajustado como es este.
- **Retroalimentativo:** a través de varias entregas de documentación y de software con el tutor de la universidad permitiendo así obtener críticas y sugerencias para el correcto avance del proyecto.

Scrum es una metodología ágil y como toda metodología ágil se se ve reflejada en los puntos que el manifiesto ágil recoge. Este manifiesto es un resumen de buenas practicas que toda metodología ha de tener para poder considerarse ágil. Uno de sus puntos más característicos y que más nos afecta es el que se refiere a la documentación.

Valorar más el software que funciona que la documentación exhaustiva

Esta es una característica a tener en cuenta, ya que hemos elegido una metodología ágil pero dadas las características del proyecto, y su realización como trabajo de fin de grado, es necesario realizar una exhaustiva documentación.

Aquí es donde entra una de las bondades antes mencionadas de *Scrum*, la flexibilidad, y es que si bien es cierto que la documentación adquiere un rol minoritario, se puede introducir dentro de los *Sprints* como una tarea más a realizar, realizando la documentación y los diagramas de manera incremental y adecuando la aplicación a esos modelos para, al final, revisarlos y completarlos teniendo así una documentación completa y exhaustiva que se ajusta a la realidad de la aplicación. Por lo tanto así se fortalece la idea de usar *Scrum* en lugar de una metodología más tradicional como podrían ser el proceso unificado (UP) o el modelo en cascada por ejemplo.

3.2.5. Elección del tipo de desarrollo a utilizar

En la actualidad a parte de las soluciones nativas que propone los fabricantes de sistemas operativos existen lo que se conoce como **desarrollos híbridos**. Estos desarrollos híbridos, ya sean tanto insertando un navegador Web o convirtiendo APIs, proponen una solución única de codificación válida para varios sistemas. Si bien tiene algunas ventajas, también tiene varios inconvenientes como pueden ser:

- **Perdida de rendimiento:** cuando se necesita rendimiento y capacidad de proceso, estas soluciones a menudo carecen de la granularidad necesaria para poder afrontar esas necesidades.
- **Dependientes del creador:** si el creador del *framework* o biblioteca deja de dar soporte, el desarrollo se vería en una situación muy delicada. Esto puede pasar también con las soluciones nativas, pero dado el estado de la tecnología y el mercado actual es menos probable.
- **Acceso a sensores:** A menudo las soluciones híbridas carecen de la posibilidad de acceder a algunos de los sensores del dispositivo. En las aplicaciones nativas esto no es un problema ya que las APIs de los diferentes sensores son desarrolladas por los creadores del sistema.

Viendo los inconvenientes de estas tecnologías y que en la ya existía una solución nativa para la plataforma Android y una solución de servidor adaptada para iOS, decidimos, sin dudar, que el desarrollo nativo para iOS era lo que necesitábamos.

3.2.6. Entregables del proyecto

Como ya se mencionó con anterioridad, el proyecto tiene una serie de entregables irrenunciables. Si bien es cierto que al haber elegido una metodología ágil, incremental e iterativo, los

documentos no verán su versión final hasta el final del proyecto entero. En cada iteración se verá una nueva versión actualizada y mejorada de cada entregable o documento permitiendo así acometer mejoras y minimizar el riesgo de errores.

A continuación se elicit la lista de entregables de este proyecto:

- Plan de Desarrollo de Software.
- Plan de Gestión de riesgos.
- Especificación de requisitos.
- Modelo de análisis.
- Modelo de diseño y arquitectura.
- Implementación y pruebas.
- Versión final del producto.
- Manual de usuario.
- Plan de seguimiento.

3.3. Organización del proyecto

3.3.1. Interfaces Externas

Las interfaces externas son aquellos miembros que permiten la conexión entre el equipo de desarrollo y los clientes por lo que la interfaz externa en este proyecto sera el *Product Owner*

3.3.2. Estructura Interna

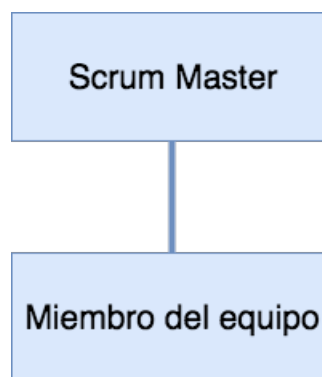


Figura 3.1: Estructura interna del proyecto

3.3.3. Roles del proyecto

A continuación se hace una descripción de los principales roles que participan en el proyecto:

1. **Product Owner**[11]:

- Define los objetivos del proyecto y producto.
- Colabora como parte del equipo para planificar y revisar cada uno de los objetivos de cada Sprint o iteración.
- Participa en las demostraciones con el cliente.

2. **Scrum Master**:

- Protege y aísla al equipo de interrupciones externas para así mantener su productividad.
- Es el valedor de que se cumplan los valores y principios ágiles
- Quitar los posibles impedimentos que le puedan surgir al equipo y que le aparten de conseguir sus objetivos en cada iteración.

3. **Team**[12]: Grupo de personas que de manera conjunta desarrollan el producto del proyecto. Comparten tanto la responsabilidad del desarrollo como el objetivo común y la calidad del mismo.

Vistos los roles de *Scrum* que intervienen en este proyecto, procedemos ahora a definir quién va a ocupar esos roles durante su desarrollo.

Rol	Nombre
Product Owner	Pablo Carrascal Muñoz y Miguel Angel Laguna
Scrum Master	Pablo Carrascal Muñoz
Team	Pablo Carrascal Muñoz

Tabla 3.1: Roles del proyecto y personas que desempeñará cada uno

3.3.4. Reuniones, eventos y planes operativos

Para seguir la metodología *Scrum* se realizaran as siguientes reuniones o eventos:

1. **Reunion inicial:** En ella se hará a grandes rasgos la planificación inicial de este proyecto.
2. **Sprint Planning** o planificación de *sprint*: En ella se debate y acuerdan los objetivos del sprint: las funcionalidades a desarrollar, posibles cambios sugeridos por el/los *product owner* o solucionar posibles fallos y errores que hayan surgido en el *sprint* anterior.
3. **Daily** o reunión diaria: Pequeña reunión de pie, no más de 5-10 minutos en la que se expone a los demás miembros del equipo el trabajo realizado el día anterior y el trabajo que se está realizando o se realizará. Si ha surgido algún impedimento o bloqueo será en esta reunión en la que se comente con el resto del equipo a modo de información pudiendo así los demás miembros dar posibles soluciones al bloqueo.
4. **Demo** o demostración: Se realizará al final de cada *sprint* y en ella se mostrarán las funcionalidades desarrolladas durante ese periodo de tiempo, para ver así, si los criterios de aceptación y los objetivos del *sprint* se han visto cumplidos o por el contrario no se ha planificado bien.

5. **Retro** o retrospectiva: Reunión para ver que ha ido bien y que no, pudiendo así observar las debilidades y puntos fuertes y mejorar en base a lo anterior. Esta reunión no tiene fecha exacta ya que puede variar dependiendo de los objetivos de cada *sprint* o del estado del proyecto.

3.4. Planificación del proyecto

3.4.1. Estimaciones del proyecto

A continuación se dará una estimación tanto temporal como de costes del proyecto y se comparará con el tiempo real para ver las posibles desviaciones y si estas afectan o han afectado al desarrollo del proyecto.

3.4.1.1. Estimación temporal

Sprint	Fecha de Inicio	Fecha de Fin
0	20/02/2017	01/03/2017
1	02/03/2017	08/03/2017
2	09/03/2017	15/03/2017
3	16/03/2017	22/03/2017
4	23/03/2017	29/03/2017
5	30/03/2017	05/04/2017
6	06/04/2017	12/04/2017
7	17/04/2017	26/04/2017
8	27/04/2017	03/05/2017
9	04/05/2017	10/05/2017
10	11/05/2017	19/05/2017
11	22/05/2017	31/05/2017
12	01/06/2017	07/06/2017
13	08/06/2017	15/06/2017

Tabla 3.2: Estimación en coste y tiempo de los *sprints* del proyecto.

En la Tabla 3.2 se muestra una estimación del número de *sprints* que se realizarán en el proyecto y sus fechas aproximadas de realización. También se detalla la complejidad de cada *sprint* a través de uno de los métodos más utilizados de las metodologías ágiles, los puntos.

Cada *User Story* se vota entre los miembros del equipo y se le asigna una valoración en puntos[13]. Se pueden usar diferentes escalas pero en este caso y dado que es una de las más conocidas y usadas elegimos la sucesión de *Fibonacci*.

Para este proyecto se estima que cada *sprint* se desarrollarán entre 8 y 10 puntos, exceptuando los tres últimos *sprints* en los que la estimación será de una media de 16 puntos. Este aumento de puntos al final viene dado por la cercanía de la fecha de entrega del proyecto y por la necesidad de corregir o ampliar tanto la aplicación como la documentación.

En sus guías, *Scrum* recomienda seguir *sprints* de aproximadamente dos semanas, sin embargo para este proyecto se decidió acortar los *sprints*, al menos de manera temporal a una semana por varias razones:

- Era la primera vez que el equipo trabajaba con esta metodología y había que acostumbrarse a la manera de trabajar de *Scrum*.
- El desconocimiento del sistema operativo iOS y su *framework* base *Cocoa Touch*, por lo que podrían surgir bloqueos o errores que en *sprints* de dos semanas sería un problema corregir (Se hablará más tarde de este tema en el apartado de gestión de riesgos)
- Era más cómodo para la supervisión del *scrum master* y el *product owner*, y así comprobar el avance en el aprendizaje de la tecnología.

3.4.1.2. Estimación de costes

Como el equipo de desarrollo es un alumno universitario y este trabajo es un trabajo de fin de grado no hay ningún coste económico en el proyecto. Por dar algún dato orientativo, a modo de calculo rápido, el proyecto esta proyectado para 600 horas por lo que, estimando el coste de un Ingeniero informático de **15€/h**, el coste total del proyecto rondaría un total de **600*15 = 9000€**

Ese dato orientativo estima el coste únicamente del coste humano del proyecto, sin embargo existen más costes a añadir como por ejemplo:

- Pago a Apple para poder ser desarrollador (**100€**)
- Coste de amortización de los equipos informáticos:
 - Ordenadores: poniendo una amortización de 4 meses de una vida media de 4 años tenemos aproximadamente **108€**
 - Teléfonos móviles: siguiendo la misma tasa de amortización tenemos aproximadamente **200€**

3.4.1.3. Desviación real del proyecto

Como se expuso con anterioridad, los datos de las anteriores subsecciones eran orientativos, una estimación ajustada a los cálculos del proyecto. A continuación veremos cual de esas estimaciones era correcta y cual ha sufrido una desviación.

Sprint	Fecha real de Inicio	Fecha real de Fin	Puntos
0	20/02/2017	01/03/2017	7
1	02/03/2017	08/03/2017	10
2	09/03/2017	20/03/2017	10
3	21/03/2017	31/03/2017	10
4	01/04/2017	06/04/2017	10
5	07/04/2017	19/04/2017	11
6	20/04/2017	05/05/2017	12
7	08/05/2017	15/05/2017	10
8	16/05/2017	24/05/2017	8
9	25/05/2017	31/05/2017	10
10	01/06/2017	07/06/2017	10
11	08/06/2017	15/06/2017	10
12	19/06/2017	05/07/2017	10

Tabla 3.3: Desviación en coste y tiempo de los *sprints* del proyecto.

Vemos como se disparan varios de los riesgos que se habían previsto, como el no llegar a tiempo a la finalización de un *sprint*, el desconocimiento de la tecnología o la mala estimación de la complejidad de las historias de usuario. Todo esto provocó que a pesar de aumentar las horas de trabajo e incluir fines de semana o festivos se produjera una variación tanto del número total de *sprints* como de la fecha e inicio de cada uno de ellos.

Por dar más detalles, se subestimaron las historias de usuario iniciales a pesar de que se tuvo en cuenta la poca familiaridad con el sistema y el entorno de desarrollo.

Al final vemos como el hecho de aumentar las horas de trabajo hace que aunque si bien el numero de *sprints* se vio reducido en número, las fechas totales del proyecto se desviarán casi un mes. Este aumento de horas y de tiempo provocó un desgaste significativo en el equipo, lo que, si el entorno de trabajo fuera real, provocaría sobrecostes en el proyecto o posibles problemas dentro del equipo.

3.5. Seguimiento del proyecto

Para ver el seguimiento total y completo del proyecto, consultar el Plan de Seguimiento en el apéndice B

Capítulo 4

Plan de Gestión de Riesgos

4.1. Introducción

Este capítulo es uno de los más importantes del proyecto, ya que para su correcta realización y para conseguir todos sus objetivos es importante tener una buena gestión de los riesgos que puedan surgir para así poder evitarlos en el caso de que aparezcan o bien, si llegan a suceder, mitigarlos o minimizarlos lo más rápido posible.

Estos riesgos pueden ser de diferente naturaleza, y dependiendo de ella y del estado en el que se encuentre el proyecto afectarán de una manera u otra.

Toda buena gestión de riesgos empieza por la identificación y análisis de todos los posibles riesgos que pueden surgir en el proyecto.

4.1.1. Alcance

Ya que la metodología utilizada *Scrum*, el análisis de riesgos es una tarea que se realizará durante todo el desarrollo del proyecto, habiendo así unos riesgos potenciales iniciales y otros que probablemente vayan surgiendo a lo largo del tiempo de desarrollo.

Este documento de riesgos por lo tanto se irá actualizando a medida que avanza el proyecto quedando así un el documento final de gestión de riesgos.

4.2. Gestión del riesgo

R01	
Nombre	Baja de alguno de los miembros del equipo
Categoría	Proyecto
Probabilidad	Baja
Contexto	Puede darse a lo largo de todo el tiempo que dure el proyecto
Análisis	Uno de los miembros del equipo no está disponible por un período de tiempo indefinido, teniendo en ese tiempo una tarea ya asignada que ha de realizar
Estrategia	Transferencia
Plan de acción	En base al tiempo que se vaya a encontrar de baja, reorganizar y replanificar el proyecto en base a las tareas que no se han podido o no se vayan a realizar para poder así mantener el control sobre el proyecto. Una posible solución es asignar más horas de trabajo posterior para así recuperar el tiempo.

Tabla 4.1: Riesgo 01

R02	
Nombre	Inestabilidad del entorno de trabajo o versiones de software
Categoría	Proceso técnico
Probabilidad	Baja-Media
Contexto	Puede darse a lo largo de todo el proyecto.
Análisis	Tanto el IDE como las versiones de sistema operativo que se usan pueden tener problemas de estabilidad o errores que deriven en retrasos u otro tipo de eventos. También pueden sufrir ese tipo de errores las herramientas utilizadas para crear y mejorar la documentación a entregar.
Estrategia	
Plan de acción	Antes de actualizar las herramientas, estudiar detenidamente las nuevas características que ofrecen las nuevas versiones para así ver si son de relevancia y decidir si actualizar o no.

Tabla 4.2: Riesgo 02

R03	
Nombre	Fallo de alguno de los equipos utilizados para desarrollar el proyecto
Categoría	Proyecto
Probabilidad	
Contexto	Puede darse durante todo el proyecto
Análisis	A lo largo del desarrollo del proyecto, alguno de los equipos que se utilizan puede fallar o romperse. En el caso de que fuera uno de los dos ordenadores el riesgo aumentaría de nivel incluso haciendo peligrar el proyecto
Estrategia	Reserva
Plan de acción	Utilizar un equipo prestado o si existe la posibilidad y compatibilidad, utilizar uno de los que la universidad proporciona

Tabla 4.3: Riesgo 03

R04	
Nombre	Poca experiencia con el software utilizado
Categoría	Producto
Probabilidad	Media
Contexto	Fase de desarrollo
Análisis	El desconocimiento de la plataforma para la que se va a desarrollar, iOS, y el lenguaje de programación Swift así como sus herramientas de desarrollo y debug pueden hacer que el proyecto se vea afectado, alargando su tiempo de desarrollo.
Estrategia	Busqueda
Plan de acción	Investigar por cuenta propia más a fondo o si se ve que no se avanza, contactar con algun experto en la materia en busca de ayuda.

Tabla 4.4: Riesgo 04

R05	
Nombre	No cumplir con la fecha de una entrega o sprint
Categoría	Proceso
Probabilidad	Media-Alta
Contexto	Puede no cumplirse la fecha de entrega de un hito o sprint en cualquiera de las fases
Análisis	Puede darse que llegue la fecha de entrega de un hito o la fecha de finalización de un sprint y que los objetivos de ese sprint no se vean cumplidos, por lo que esto retrasará el resto de la realización retrasando así también su fecha de entrega o llegando incluso a ponerlo en peligro.
Estrategia	Transferencia
Plan de acción	Replanificar el siguiente sprint para no retrasar el proyecto

Tabla 4.5: Riesgo 05

R06	
Nombre	Pobre captación de requisitos y/o criterios de aceptación
Categoría	Proceso
Probabilidad	Media
Contexto	Realización incorrecta del trabajo
Análisis	Si un requisito o varios no estan bien definidos y no se tienen claros cuales son sus criterios de aceptación el cliente puede quedar insatisfecho con el producto
Estrategia	Búsqueda
Plan de acción	Hablar con la interfaz externa para comprobar que se entendieron correctamente los requisitos.

Tabla 4.6: Riesgo 06

R07	
Nombre	Errores software en el servidor
Categoría	Proceso
Probabilidad	Baja-Media
Contexto	Bugs en la parte del servidor dada por el cliente
Análisis	Al ser una parte heredada en el proyecto y que hay que utilizar, puede que este software no sea robusto del todo provocando el mal funcionamiento del sistema en conjunto llevando a la aplicación a estados inesperados
Estrategia	Búsqueda
Plan de acción	Hablar con el cliente para que especifiquen el error y en caso de que no se pueda solucionar, especificar la manera de gestionarlo

Tabla 4.7: Riesgo 07

R08	
Nombre	Mala estimación de los puntos de historia de usuario
Categoría	Proceso
Probabilidad	Alta-Muy Alta
Contexto	Puede no cumplirse una fecha de entrega de un hito
Análisis	Puede darse una mala estimación del esfuerzo necesario para una tarea que luego resulte ser excesivamente pesada y requerir de un esfuerzo superior al estimado, retrasando así el resto del proyecto
Estrategia	Transferencia
Plan de acción	Aumentar el tiempo del <i>Sprint</i> actual y replanificar el siguiente para que el proyecto en conjunto no se vea afectado

Tabla 4.8: Riesgo 08

R09	
Nombre	Aparición de errores en el software desarrollado
Categoría	Proceso
Probabilidad	Media
Contexto	Puede no cumplirse una fecha de entrega de un hito
Análisis	Puede ser que durante el proceso de revisión y pruebas del software después del desarrollo de una historia de usuario aparezcan errores software que lleven más tiempo del previsto para corregirlos. Esto haría que la iteración no se acabara a tiempo.
Estrategia	Transferencia
Plan de acción	Aumentar el tiempo de la iteración actual y re-planificar el siguiente para que el proyecto en conjunto no se vea afectado

Tabla 4.9: Riesgo 09

4.3. Control de riesgos

Como era de esperar, a lo largo del proyecto se han disparado numerosos riesgos. Los más comunes tienen como consecuencia no llegar a tiempo al final de alguna iteración, teniendo así que, primero ampliar el tiempo de desarrollo de la actual iteración y segundo replanificar la siguiente para que el proyecto en su conjunto no se viera afectado.

Las principales causas han sido:

- Poca experiencia con el software y sus herramientas.
- Dada esa poca experiencia, se ha calculado mal el tiempo necesario para llevar a cabo ciertas tareas, subestimando así el tiempo que era necesario para llevarlas a cabo.
- Aparición de errores software que, al tener poca experiencia, ha llevado más tiempo de lo estimado acabar una tarea o iteración.

Las consecuencias finales de todo esto ha implicado dedicar a ciertas tareas más recursos de tiempo de lo que se había estimado, aumentando la duración en horas trabajadas del proyecto. Sin embargo y gracias a la gestión del riesgo y a la flexibilidad de la metodología *Scrum* el proyecto no se vio alargado en el tiempo más de lo que se había estimado.

Capítulo 5

Requisitos

5.1. Los requisitos en el desarrollo ágil

La especificación de requisitos en los desarrollos ágiles difieren bastante del modelo tradicional. La parte común es que sigue siendo el cliente el que se encarga de definir el problema y la funcionalidad que desea. Aunque bien es cierto que en las metodologías ágiles se intenta guiar al cliente para orientarle mejor sobre su producto en lugar de tener que acatar un pliego de condiciones ya establecido.

Los requisitos en el mundo del desarrollo ágil se suelen separar en historias de usuario y estas son agrupadas en una lista donde se les asignan diferentes prioridades. Esta lista evoluciona a lo largo de todo el desarrollo, por lo que no se verá completa hasta que el desarrollo termine.

Si bien en un desarrollo tradicional los requisitos son tomados por un grupo de expertos en la materia y analistas, en las metodologías ágiles el equipo entero conoce la visión del cliente y la lista de historias de usuario se actualiza de forma continua con sus integrantes.

Teniendo en cuenta la explicación dada se listan a continuación todos los requisitos del proyecto.

5.2. Requisitos funcionales

ID	Nombre	Descripción
RF01	Identificarse en la aplicación	El sistema deberá permitir al usuario entrar en la aplicación con su usuario y contraseña corporativos.
RF02	Editar perfil corporativo	El sistema deberá permitir al usuario cambiar tanto su foto de perfil como sus redes sociales, posición, fecha de nacimiento y teléfono en la aplicación.
RF03	Ver compañeros de trabajo	El sistema deberá permitir al usuario ver una lista con sus compañeros de trabajo y realizar acciones tales como llamar, enviar correo o ver sus redes sociales.
RF04	Pedir vacaciones	El sistema deberá permitir al usuario solicitar vacaciones para un periodo de tiempo si se le han concedido vacaciones para ese año.
RF05	Consultar vacaciones	El sistema deberá permitir al usuario ver si dispone de días para pedir vacaciones para el año actual, al anterior y el siguiente.
RF06	Notificación de cambio de vacaciones	El sistema deberá permitir al usuario activar la recepción de notificaciones sobre alguna novedad en sus vacaciones.
RF07	Notificación de avisos	El sistema deberá permitir al usuario activar la recepción de avisos que el administrador de su sistema desee mandarle.
RF08	Modificación de vacaciones	El sistema deberá permitir al usuario pedir la modificación del estado de sus vacaciones si estas han sido concedidas o se encuentran pendientes de revisión.
RF09	Consulta de avisos	El sistema deberá permitir al usuario ver una lista con los avisos que se le ha mandado el administrador del sistema.
RF10	Cacheo de la información	El sistema deberá permitir al usuario ver información sobre sus vacaciones aunque no disponga de red.
RF11	Avisos por falta de conexión	El sistema deberá avisar al usuario cuando no disponga de red o cuando la red vuelva a estar disponible.
RF12	Obtención de datos de un servidor	El sistema deberá poder consultar y actualizar un servidor de tipo REST
RF13	<i>Timeline</i> de Twitter	El sistema deberá mostrar el <i>timeline</i> de Twitter de la compañía.

Tabla 5.1: Requisitos funcionales del proyecto.

5.3. Requisitos no funcionales

ID	Nombre	Descripción
RNF01	Solicitud de permisos	La aplicación debe solicitar los permisos necesarios al usuario para realizar las tareas correspondientes para su correcto funcionamiento.
RNF02	Facilidad de uso	La aplicación debe ser fácil de usar.
RNF03	Uso de CoreData para la persistencia	La aplicación deberá usar CoreData para la persistencia de los datos tanto online como offline.
RNF04	Versiones compatibles	La aplicación deberá ser compatible con las versiones de iOS 9.* y 10.*
RNF05	Internacionalización	La aplicación deberá estar traducida tanto en castellano como en inglés.
RNF06	Tiempo de respuesta	El tiempo de respuesta de cada petición a red deberá ser de menos de 15 segundos. Es decir, el <i>timeout</i> de cada llamada es 15 segundos.

Tabla 5.2: Requisitos no funcionales del proyecto.

5.4. Requisitos de información

ID	Nombre	Descripción
RIN01	Contenido de un usuario	nombre, apellidos, email, fecha de nacimiento, telefono, cuenta de twitter, cuenta de linkedin, id, fecha de actualización.
RIN02	Contenido de un aviso	tipo, mensaje, fecha de actualización, creador del aviso, vacaciones máximas, estado de vacaciones, año de vacaciones, id de las vacaciones a modificar.
RIN03	Contenido de unas vacaciones concretas	fecha de inicio, fecha de fin, estado, fecha de actualización, notas.
RIN04	Contenido de vacaciones anuales	número de días de vacaciones máximos, número de días de vacaciones pendientes, numero de días de vacaciones usados, año, fecha de actualización

Tabla 5.3: Requisitos de información del proyecto.

Capítulo 6

Análisis

En este capítulo se describirán los actores que participan en el sistema y se especificarán los casos de uso de la aplicación. También se incluirá el modelo de dominio, la descripción de sus clases y los diagramas de actividad de la aplicación.

Toda esta información tiene como objetivo proporcionar una descripción completa de la funcionalidad que tendrá la aplicación.

Nota aclaratoria: Como ya se ha venido diciendo a lo largo de esta memoria, el desarrollo del proyecto se hace de manera ágil, por ello tanto los requisitos de el apartado anterior como los casos de uso que se detallen en este, serán traducidos en historias de usuario (Cada caso de uso será por lo tanto una historia de usuario). Puede obtenerse más información en el Plan de Seguimiento que se encuentra en el apéndice B

6.1. Modelo de Casos de Uso

6.1.1. Actores

- **Empleado:** será el único actor del sistema y representa al usuario que utilizará la aplicación.

Nota: La aplicación *SGEmployee* cuenta con un actor que seria el empleado de la empresa. Sin embargo, en el sistema completo, incluyendo la parte servidor, existen más actores:

- **Administrador:** Es el encargado de dar de alta a los empleados y de gestionar quien puede y quien no revisar y aceptar vacaciones, escribir mensajes, etc.
- **Revisor:** Puede cambiar el estado de las vacaciones de los empleados y puede a su vez enviar mensajes a los mismos.

En esta memoria solo aparece lo referente a la aplicación desarrollada para iOS, por lo que, el resto de actores, no se verán reflejados de manera detallada en los diagramas venideros.

6.1.2. Diagrama de casos de uso

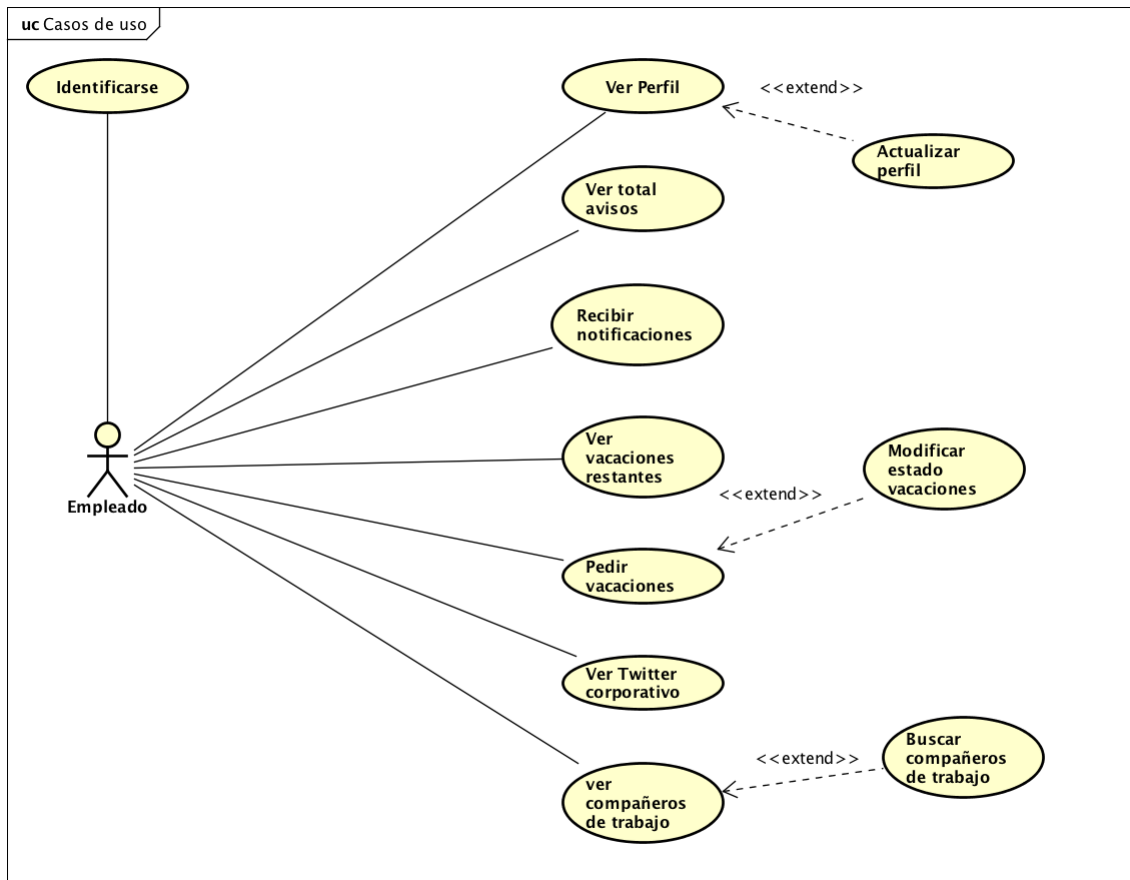


Figura 6.1: Diagrama de casos de uso de la aplicación.

6.1.3. Especificación de casos de uso

CU01	Identificarse
Actores	Empleado
Descripción	El usuario desea identificarse en la aplicación con sus credenciales corporativas
Secuencia Normal	<ol style="list-style-type: none">1. El caso de uso se inicia cuando el actor desea identificarse en la aplicación.2. El <i>Sistema</i> muestra la pantalla de identificación.3. El <i>Empleado</i> introduce su usuario y contraseña corporativos.4. El <i>Sistema</i> comprueba que las credenciales sean correctas de manera remota.5. El <i>Sistema</i> muestra el <i>timeline</i> de <i>Twitter</i>.
Postcondición	El Actor <i>Empleado</i> queda identificado en el sistema.
Excepciones	<ol style="list-style-type: none">5 Si el dispositivo no tiene conexión a Internet, el <i>Sistema</i> avisa al <i>Empleado</i> y el caso de uso queda sin efecto.5 Si las credenciales introducidas por el <i>Empleado</i> no son correctas, el <i>Sistema</i> avisa al <i>Empleado</i> y el caso de uso continua en el paso 2.

Tabla 6.1: Descripción del CU01

CU02	Ver perfil
Actores	Empleado
Descripción	El usuario desea ver su perfil corporativo en la aplicación.
Precondición	El usuario debe estar identificado en la aplicación.
Secuencia Normal	<ol style="list-style-type: none"> 1. El caso de uso se inicia cuando el actor <i>Empleado</i> solicita ver su perfil. 2. El <i>Sistema</i> carga la información de perfil (local o remota). 3. El <i>Sistema</i> muestra los datos de perfil actualizados al usuario.
Postcondición	Ninguna
Flujo alternativo	No existe
Excepciones	<ol style="list-style-type: none"> 2 Si no hay información en local, el <i>Sistema</i> carga información por defecto. 2 Si se produce un error tratando de obtener información en remoto, el <i>Sistema</i> avisa al <i>Empleado</i> y el caso de uso termina aquí.

Tabla 6.2: Descripción del CU02

CU03	Modificar perfil
Actores	Empleado
Descripción	El usuario desea modificar su perfil corporativo en la aplicación.
Precondición	El usuario debe estar identificado en la aplicación.
Secuencia Normal	<ol style="list-style-type: none"> 1. El caso de uso se inicia cuando el actor <i>Empleado</i> desea editar su perfil. 2. El <i>Sistema</i> carga la información de perfil (local o remota). 3. El <i>Empleado</i> modifica los datos del perfil. 4. El <i>Sistema</i> actualiza los datos del perfil con el servidor remoto.
Postcondición	Ninguna
Flujo Alternativo	Si el <i>Empleado</i> cancela en cualquier momento el caso de uso queda sin efecto.
Excepciones	<ol style="list-style-type: none"> 2 Si no hay información en local, el <i>Sistema</i> carga información por defecto. 2,4 Si se produce un error tratando de obtener o actualizar información en remoto, el <i>Sistema</i> avisa al <i>Empleado</i> y el caso de uso queda sin efecto.

Tabla 6.3: Descripción del CU03

CU04	Ver vacaciones restantes
Actores	Empleado
Descripción	El <i>Empleado</i> desea ver las vacaciones que tiene pedidas y los días que le quedan por pedir.
Precondición	El usuario debe estar identificado en la aplicación.
Secuencia Normal	<ol style="list-style-type: none"> 1. El caso de uso se inicia cuando el actor <i>Empleado</i> desea ver los días de vacaciones que tiene en total y cuántos ha gastado. 2. El <i>Sistema</i> carga la información de vacaciones (local o remota). 3. El <i>Sistema</i> muestra los datos actualizados.
Postcondición	Ninguna
Flujo alternativo	No existe
Excepciones	<ol style="list-style-type: none"> 2 Si no hay información en local, muestra información por defecto. 2,4 Si se produce un error tratando de obtener o actualizar información en remoto, el <i>Sistema</i> avisa al <i>Empleado</i> y el caso de uso queda sin efecto.

Tabla 6.4: Descripción del CU04

CU05	Pedir vacaciones
Actores	Empleado
Descripción	El <i>Empleado</i> desea pedir vacaciones
Precondición	<ul style="list-style-type: none"> ▪ El usuario debe estar identificado en la aplicación. ▪ El usuario tiene que tener días disponibles de vacaciones.
Secuencia Normal	<ol style="list-style-type: none"> 1. El caso de uso se inicia cuando el actor <i>Empleado</i> desea pedir nuevas vacaciones. 2. El <i>Sistema</i> muestra la pantalla de solicitud de vacaciones. 3. El <i>Empleado</i> introduce las fechas para las que quiere vacaciones. 4. El <i>Sistema</i> actualiza los datos con el servidor.
Postcondición	El <i>Empleado</i> tiene que tener una solicitud de vacaciones para la fecha dada.
Flujo alternativo	Si el <i>Empleado</i> cancela en cualquier momento el caso de uso queda sin efecto.
Excepciones	<ol style="list-style-type: none"> 4 Si se produce un error tratando de obtener o actualizar información en remoto, el <i>Sistema</i> avisa al <i>Empleado</i> y el caso de uso queda sin efecto.

Tabla 6.5: Descripción del CU05

CU06	Modificar estado vacaciones
Actores	Empleado
Descripción	El <i>Empleado</i> desea pedir modificar el estado de unas vacaciones.
Precondición	<ul style="list-style-type: none"> ▪ El usuario debe estar identificado en la aplicación. ▪ El usuario debe tener una petición de vacaciones en estado <i>Pendiente de revisión</i> o <i>Aceptada</i>
Secuencia Normal	<ol style="list-style-type: none"> 1. El caso de uso se inicia cuando el actor <i>Empleado</i> desea modificar el estado de unas vacaciones. 2. El <i>Usuario</i> selecciona unas vacaciones para actualizar su estado. 3. El <i>Sistema</i> actualiza el estado de las vacaciones de manera remota. 4. El <i>Sistema</i> muestra los datos actualizador en pantalla.
Postcondición	Las vacaciones habrán cambiado de estado.
Flujo alternativo	<ol style="list-style-type: none"> 3.1 Si el estado es <i>Pendiente de revisión</i> las vacaciones se cancelan automáticamente. 3.2 Si el estado es <i>Aceptada</i> las vacaciones pasan al estado <i>Pendientes de cancelación</i>.
Excepciones	<ol style="list-style-type: none"> 3 Si se produce un error tratando de obtener o actualizar información en remoto, el <i>Sistema</i> avisa al <i>Empleado</i> y el caso de uso queda sin efecto.

Tabla 6.6: Descripción del CU06

CU07	Ver total avisos
Actores	Empleado
Descripción	El <i>Empleado</i> desea ver todos los avisos y notificaciones que le han llegado.
Precondición	El usuario debe estar identificado en la aplicación.
Secuencia Normal	<ol style="list-style-type: none"> 1. El caso de uso se inicia cuando el actor <i>Empleado</i> desea ver los avisos que le han llegado. 2. El <i>Sistema</i> carga la información de avisos (local o remota). 3. El <i>Sistema</i> comprueba que el dispositivo tenga conexión a Internet. 4. El <i>Sistema</i> muestra los datos actualizados en pantalla.
Postcondición	Ninguna
Flujo alternativo	No existe
Excepciones	<ol style="list-style-type: none"> 2 Si se produce un error tratando de obtener o actualizar información en remoto, el <i>Sistema</i> avisa al <i>Empleado</i> y el caso de uso queda sin efecto.

Tabla 6.7: Descripción del CU07

CU08	Recibir notificaciones
Actores	Empleado
Descripción	El <i>Empleado</i> desea recibir notificaciones <i>push</i>
Precondición	El usuario debe estar identificado en la aplicación.
Secuencia Normal	<ol style="list-style-type: none"> 1. El caso de uso se inicia cuando el actor <i>Empleado</i> desea activar o desactivar las notificaciones. 2. El <i>Sistema</i> le presenta los ajustes de notificaciones. 3. El <i>Empleado</i> activa o desactiva las notificaciones. 4. El <i>Sistema</i> activa las notificaciones y se registra en el servidor remoto.
Postcondición	Las notificaciones se habrán activado.
Flujo alternativo	<ol style="list-style-type: none"> 3.1 Si las notificaciones están activadas, se desactivarán, y el caso de uso continua.
Excepciones	<ol style="list-style-type: none"> 4 Si se produce un error tratando de obtener o actualizar información en remoto, el <i>Sistema</i> avisa al <i>Empleado</i> y el caso de uso queda sin efecto.

Tabla 6.8: Descripción del CU08

CU09	Ver Twitter Corporativo
Actores	Empleado
Descripción	El <i>Empleado</i> desea ver el <i>timeline</i> del Twitter de la empresa.
Precondición	El usuario debe estar identificado en la aplicación.
Secuencia Normal	<ol style="list-style-type: none"> 1. El caso de uso se inicia cuando el actor <i>Empleado</i> desea ver el contenido de la cuenta de Twitter de la compañía. 2. El <i>Sistema</i> muestra el <i>timeline</i> de Twitter de la empresa.
Postcondición	Ninguna
Flujo alternativo	No existe
Excepciones	<ol style="list-style-type: none"> 2 Si se produce un error tratando de obtener la información en remoto, el <i>Sistema</i> avisa al <i>Empleado</i> y el caso de uso queda sin efecto.

Tabla 6.9: Descripción del CU09

CU010	Ver compañeros de trabajo
Actores	Empleado
Descripción	El <i>Empleado</i> desea ver el un resumen del perfil de sus compañeros de trabajo.
Precondición	El usuario debe estar identificado en la aplicación.
Secuencia Normal	<ol style="list-style-type: none"> 1. El caso de uso se inicia cuando el actor <i>Empleado</i> pulsa en el menú el botón de Equipo. 2. El <i>Sistema</i> carga la información de los compañeros (local o remota). 3. El <i>Sistema</i> muestra en pantalla la información actualizada.
Postcondición	Ninguna
Flujo alternativo	No existe
Excepciones	<ol style="list-style-type: none"> 2 Si se produce un error tratando de obtener o actualizar información en remoto, el <i>Sistema</i> avisa al <i>Empleado</i> y el caso de uso queda sin efecto.

Tabla 6.10: Descripción del CU10

CU011	Buscar compañeros de trabajo
Actores	Empleado
Descripción	El <i>Empleado</i> desea ver el un resumen del perfil de sus compañeros de trabajo.
Precondición	El usuario debe estar identificado en la aplicación.
Secuencia Normal	<ol style="list-style-type: none"> 1. El caso de uso se inicia cuando el actor <i>Empleado</i> desea buscar un compañero de trabajo. 2. El <i>Sistema</i> ejecuta el Caso de uso 10. 3. El <i>Empleado</i> introduce el nombre del compañero. 4. El <i>Sistema</i> busca entre los empleados el que encaje con el nombre introducido por el <i>Empleado</i>. y muestra su perfil en pantalla.
Postcondición	Ninguna
Flujo alternativo	Si el <i>Empleado</i> cancela en cualquier momento el caso de uso queda sin efecto.
Excepciones	<ol style="list-style-type: none"> 4 Si no hay ninguna coincidencia, se mostrará una lista vacía y el caso de uso queda sin efecto.

Tabla 6.11: Descripción del CU11

6.1.4. Realización de los casos de uso en análisis

6.1.4.1. Modelo de dominio

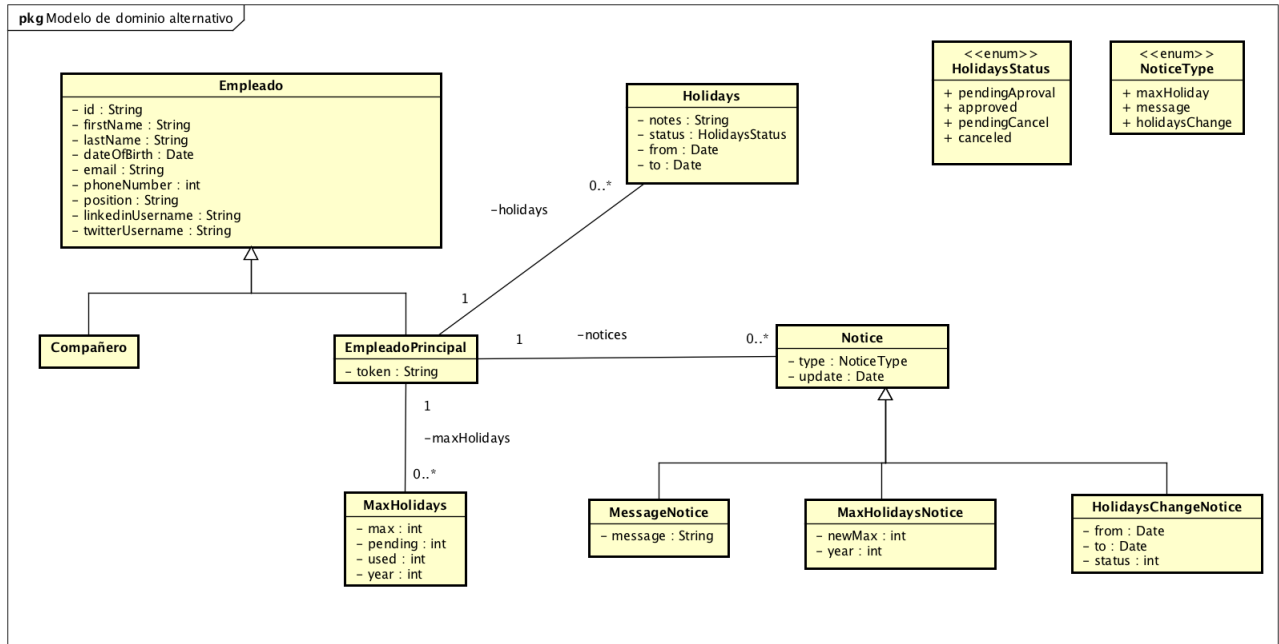


Figura 6.2: Modelo de dominio de la aplicación.

6.1.5. Descripción de las clases del modelo de dominio.

6.1.5.1. Empleado

- **Descripción:** clase que modela un empleado.
- **Responsabilidades:** referenciar un empleado de la compañía en la aplicación.
- **Atributos:**
 - *id*: identificador único del empleado.
 - *firstName*: nombre del empleado.
 - *lastName*: apellidos del empleado.
 - *dateOfBirth*: fecha de nacimiento del empleado.
 - *email*: correo electrónico del empleado.
 - *phoneNumber*: número de teléfono del empleado.
 - *position*: cargo del empleado en la empresa.
 - *linkedinUsername*: nombre usuario de *LinkedIn* del empleado.
 - *twitterUsername*: nombre de usuario de *Twitter* del empleado.

6.1.5.2. Compañero

- **Descripción:** clase que modela un compañero en la empresa. Clase hija de *Empleado*
- **Responsabilidades:** referenciar un compañero de empresa en la aplicación.

6.1.5.3. EmpleadoPrincipal

- **Descripción:** clase que modela el usuario principal de la aplicación. Clase hija de *Empleado*
- **Responsabilidades:** referenciar un el usuario principal de la aplicación.
- **Atributos:**
 - *token*: token que recibe del servidor para identificar la sesión actual a la hora de realizar peticiones contra él.

6.1.5.4. MaxHolidays

- **Descripción:** clase que modela los días de vacaciones de los que se disponen para un año y los días usados del total.
- **Responsabilidades:** referenciar un el la vacaciones asignadas para un año en la aplicación.
- **Atributos:**
 - *max*: días totales que se disponen de vacaciones.
 - *pending*: número de días restantes.
 - *used*: número de días utilizados.
 - *year*: año para el que se ha establecido el número de días disponibles para vacaciones.

6.1.5.5. Notice

- **Descripción:** clase que modela un aviso o notificación.
- **Responsabilidades:** referenciar un aviso o notificación en la aplicación.
- **Atributos:**
 - *type*: tipo de la notificación, si es un mensaje, un cambio de estado de vacaciones o un aumento de los días disponibles.
 - *update*: fecha del aviso o notificación.

6.1.5.6. MessageNotice

- **Descripción:** clase que modela un aviso o notificación que muestra un mensaje. Clase hija de *Notice*.

- **Responsabilidades:** referenciar un aviso o notificación del tipo mensaje en la aplicación.
- **Atributos:**
 - *message*: mensaje a mostrar.

6.1.5.7. MaxHolidaysNotice

- **Descripción:** clase que modela un aviso o notificación que cambia los días totales de los que se dispone para vacaciones. Clase hija de *Notice*.
- **Responsabilidades:** referenciar un aviso o notificación del tipo aumento o decremento de días totales de vacaciones.
- **Atributos:**
 - *newMax*: nuevo número de días disponibles.
 - *year*: año para el que se modifica.

6.1.5.8. HolidayChangeNotice

- **Descripción:** clase que modela una notificación de cambio de estado para una solicitud de vacaciones. Clase hija de *Notice*.
- **Responsabilidades:** referenciar un aviso o notificación del tipo cambio de estado en vacaciones.
- **Atributos:**
 - *from*: fecha de inicio de las vacaciones.
 - *to*: fecha de fin de las vacaciones.
 - *status*: nuevo estado para esas vacaciones.

6.1.5.9. Holiday

- **Descripción:** clase que modela unas vacaciones.
- **Responsabilidades:** referenciar unas vacaciones en la aplicación.
- **Atributos:**
 - *notes*: mensaje opcional que el administrador escriba.
 - *status*: estado de las vacaciones.
 - *from*: fecha de inicio de las vacaciones.
 - *to*: fecha de fin de las vacaciones.

6.1.6. Diagramas de actividad

6.1.6.1. CU01: Identificarse

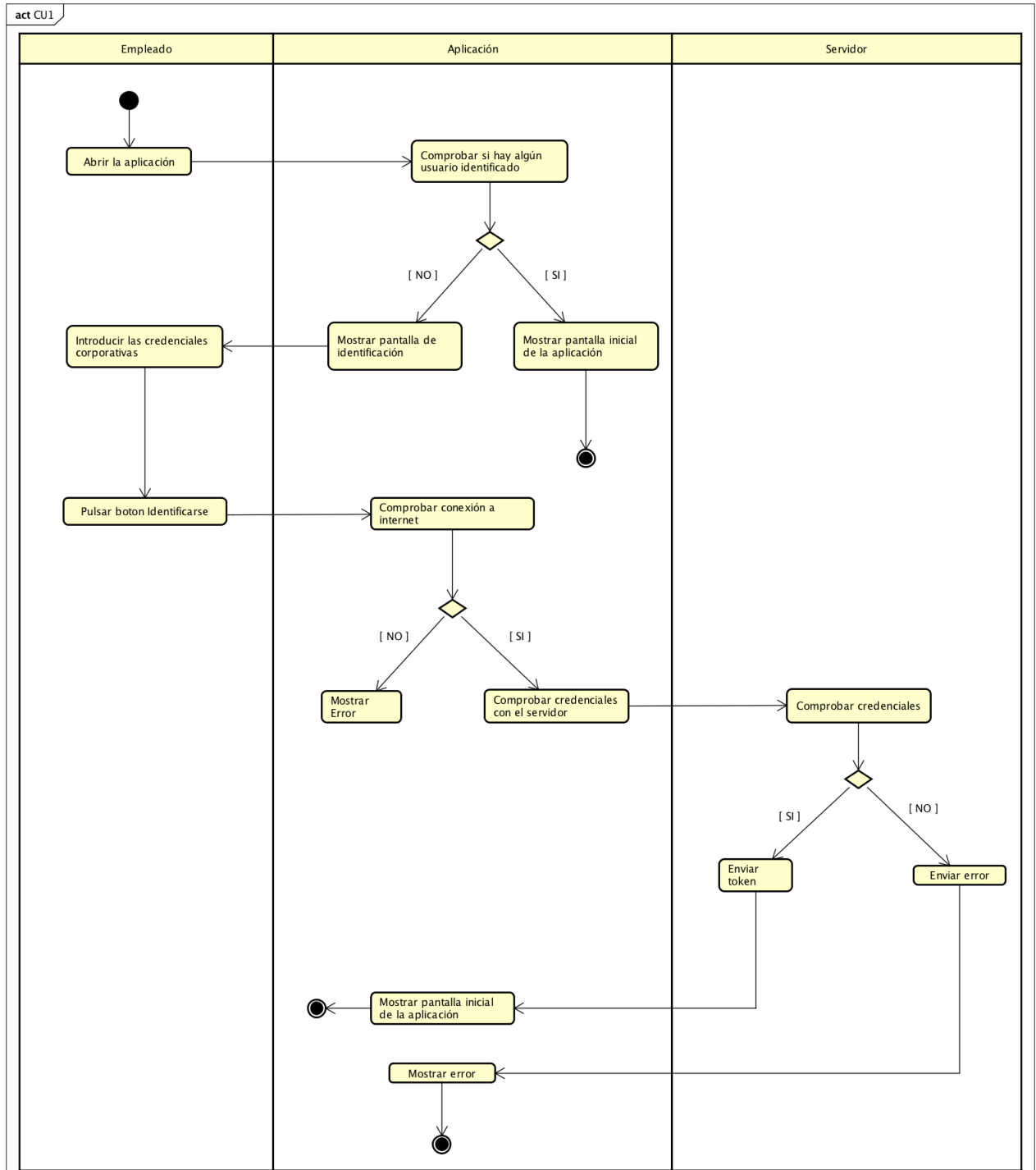


Figura 6.3: Diagrama de actividad del CU01.

6.1.6.2. CU02: Ver Perfil

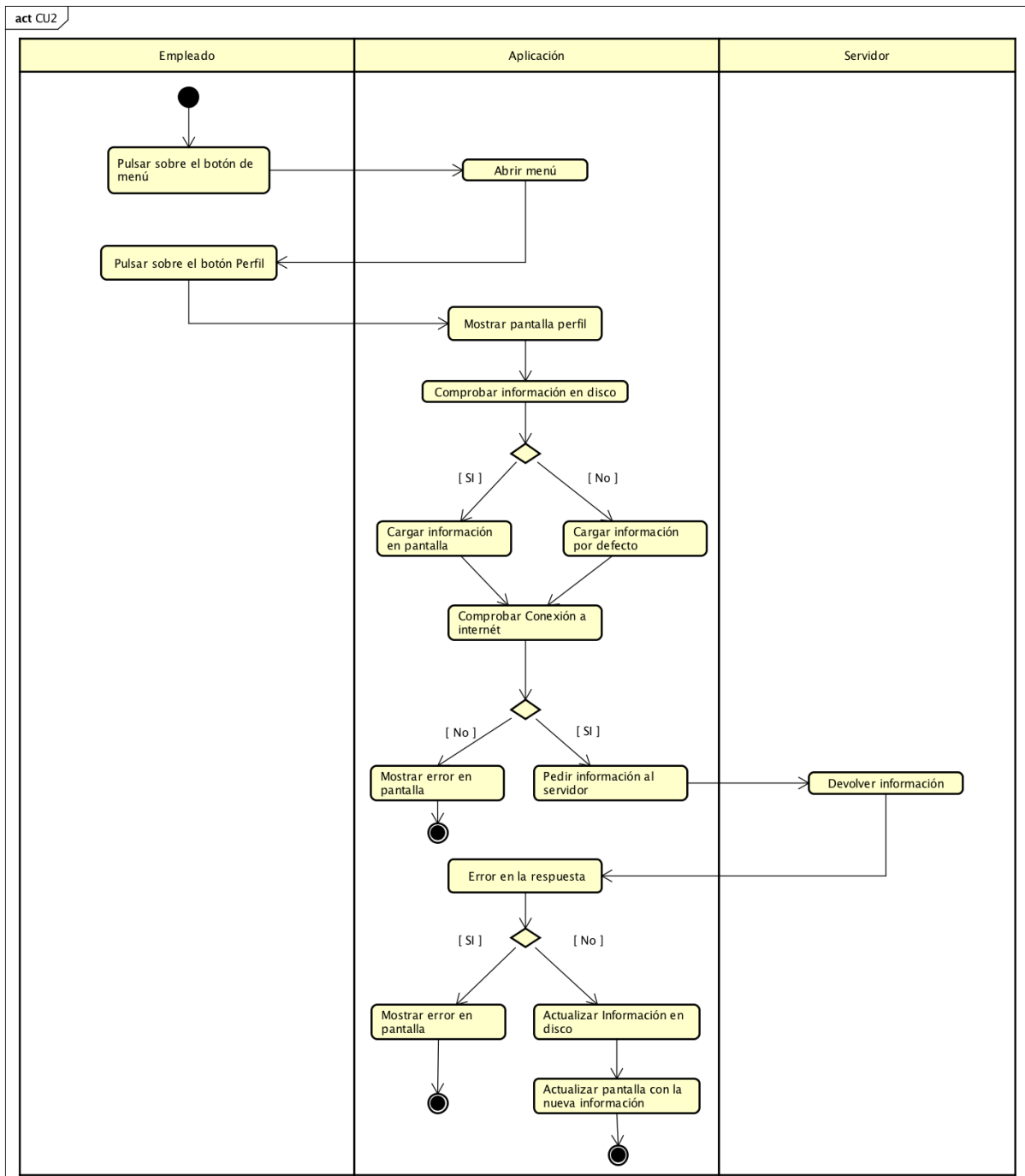


Figura 6.4: Diagrama de actividad del CU02.

6.1.6.3. CU03: Modificar Perfil

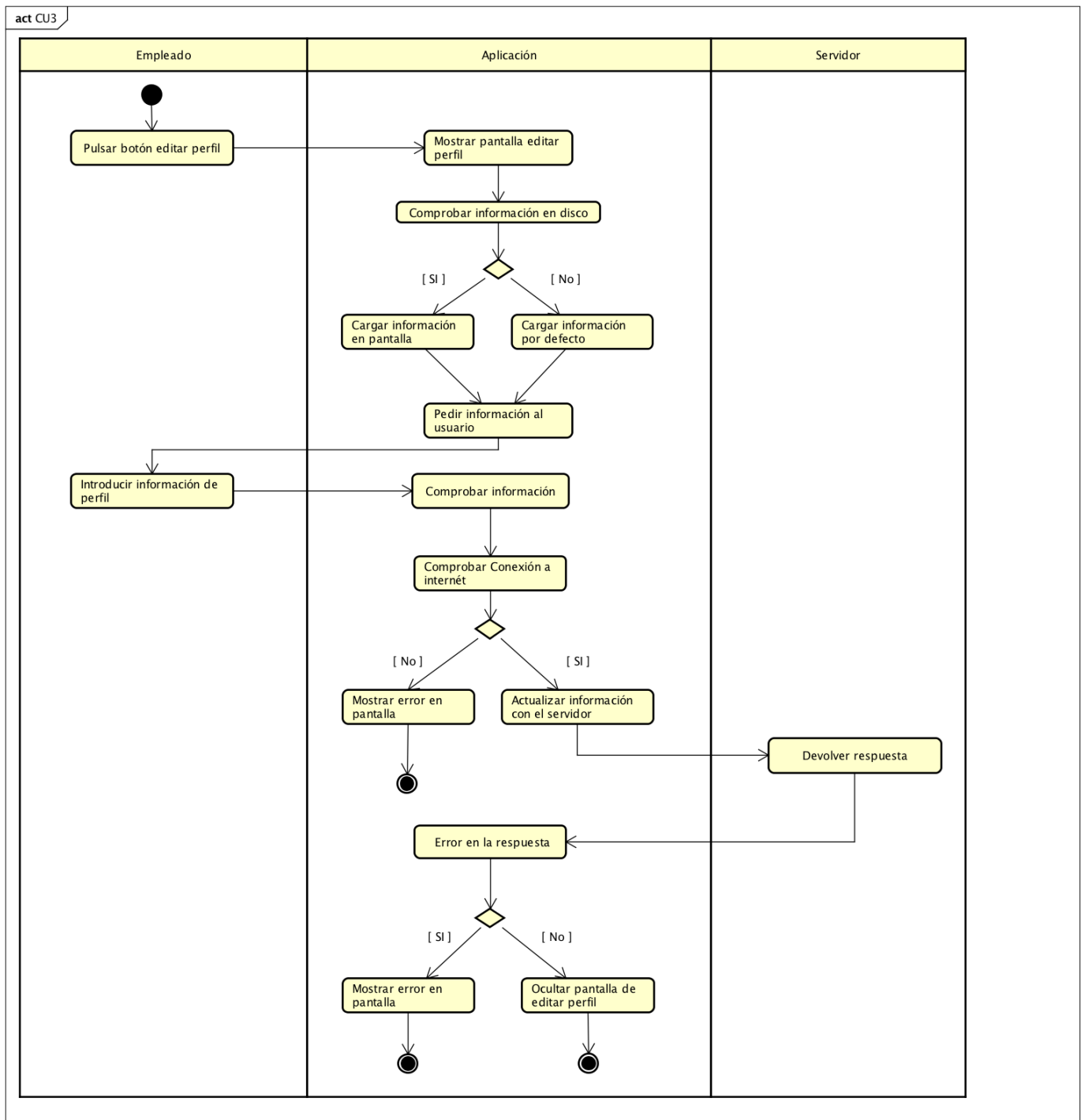


Figura 6.5: Diagrama de actividad del CU03.

6.1.6.4. CU04: Ver vacaciones restantes

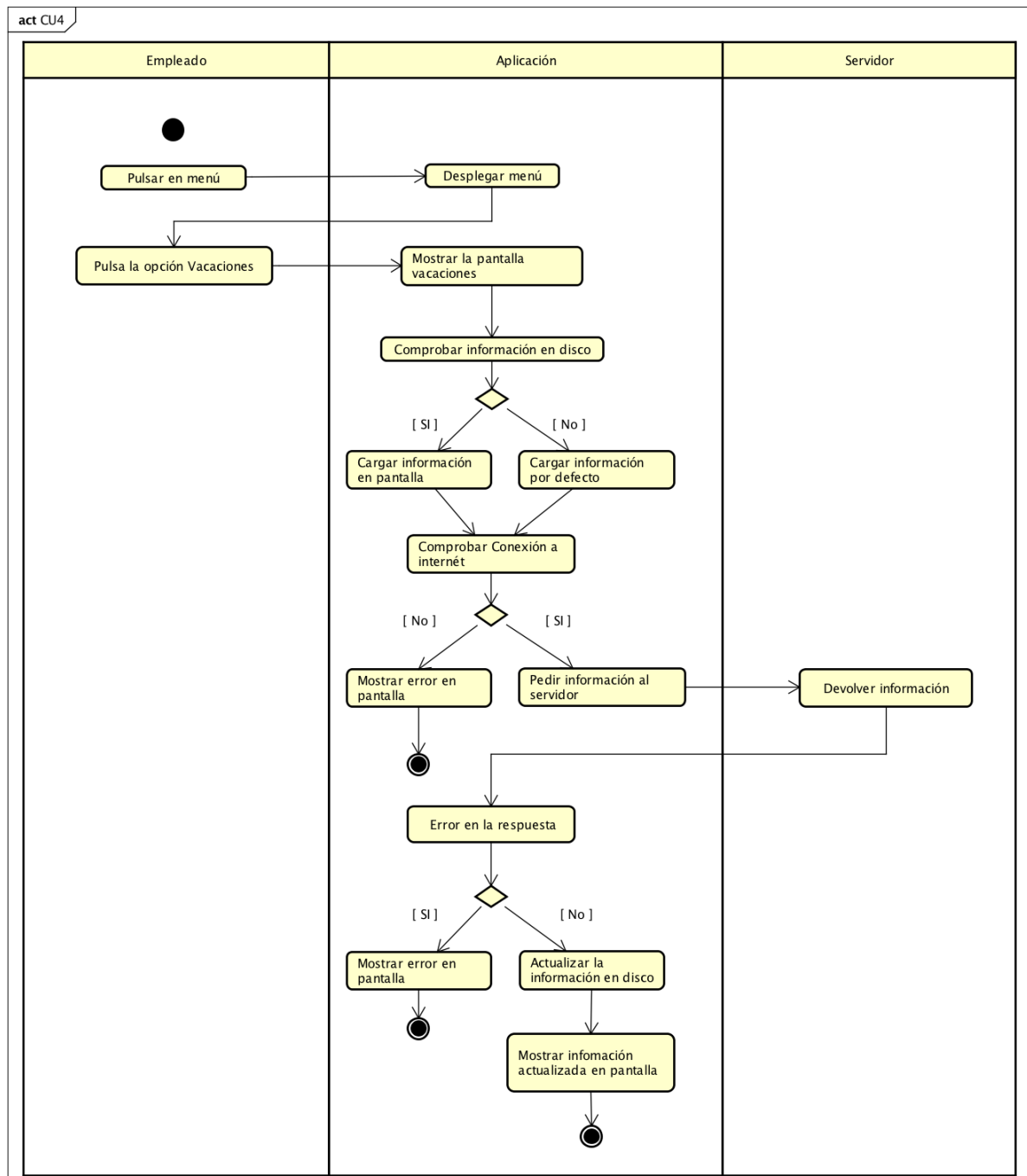


Figura 6.6: Diagrama de actividad del CU04.

6.1.6.5. CU05: Pedir vacaciones

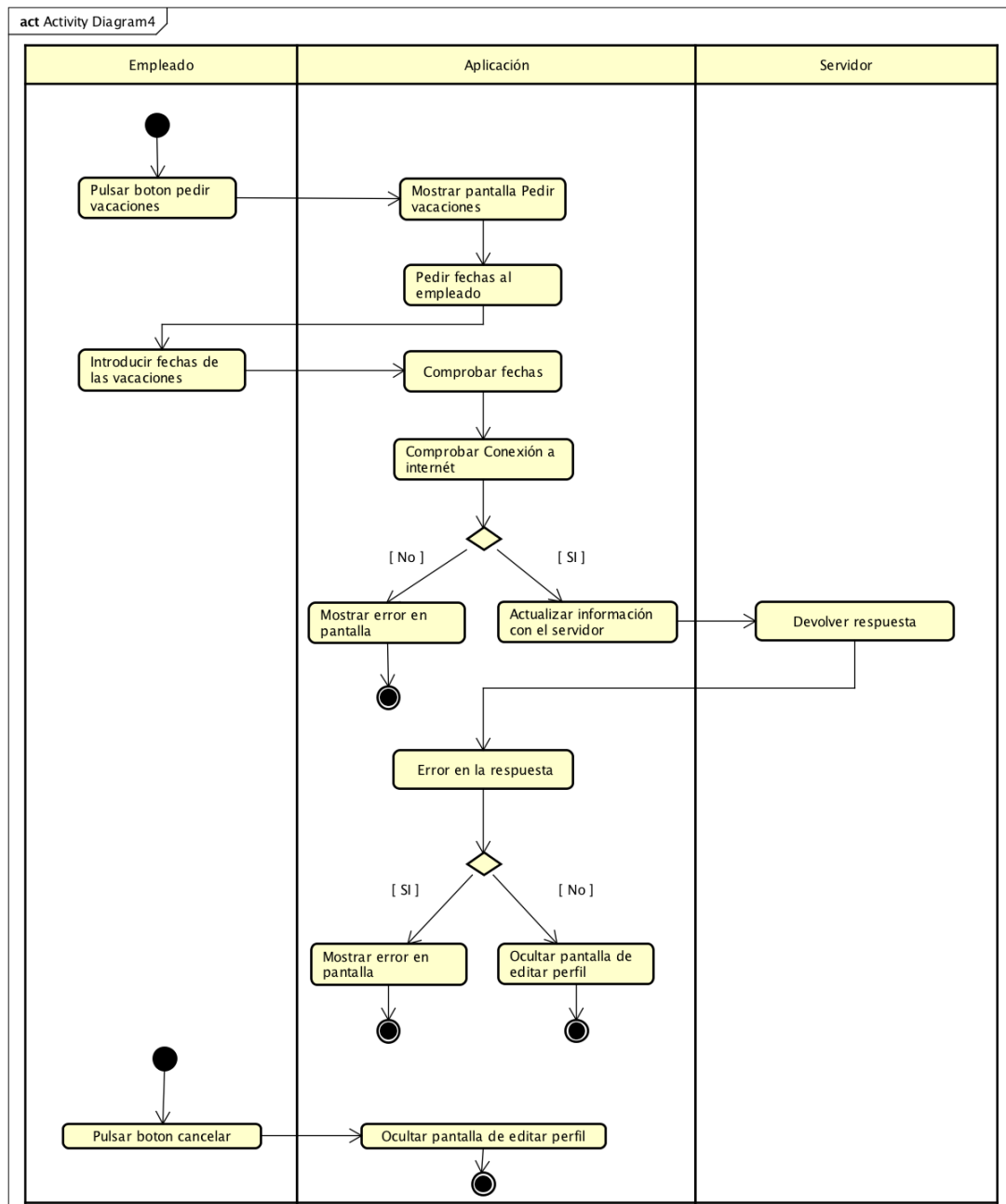


Figura 6.7: Diagrama de actividad del CU05.

6.1.6.6. CU06: Modificar estado vacaciones

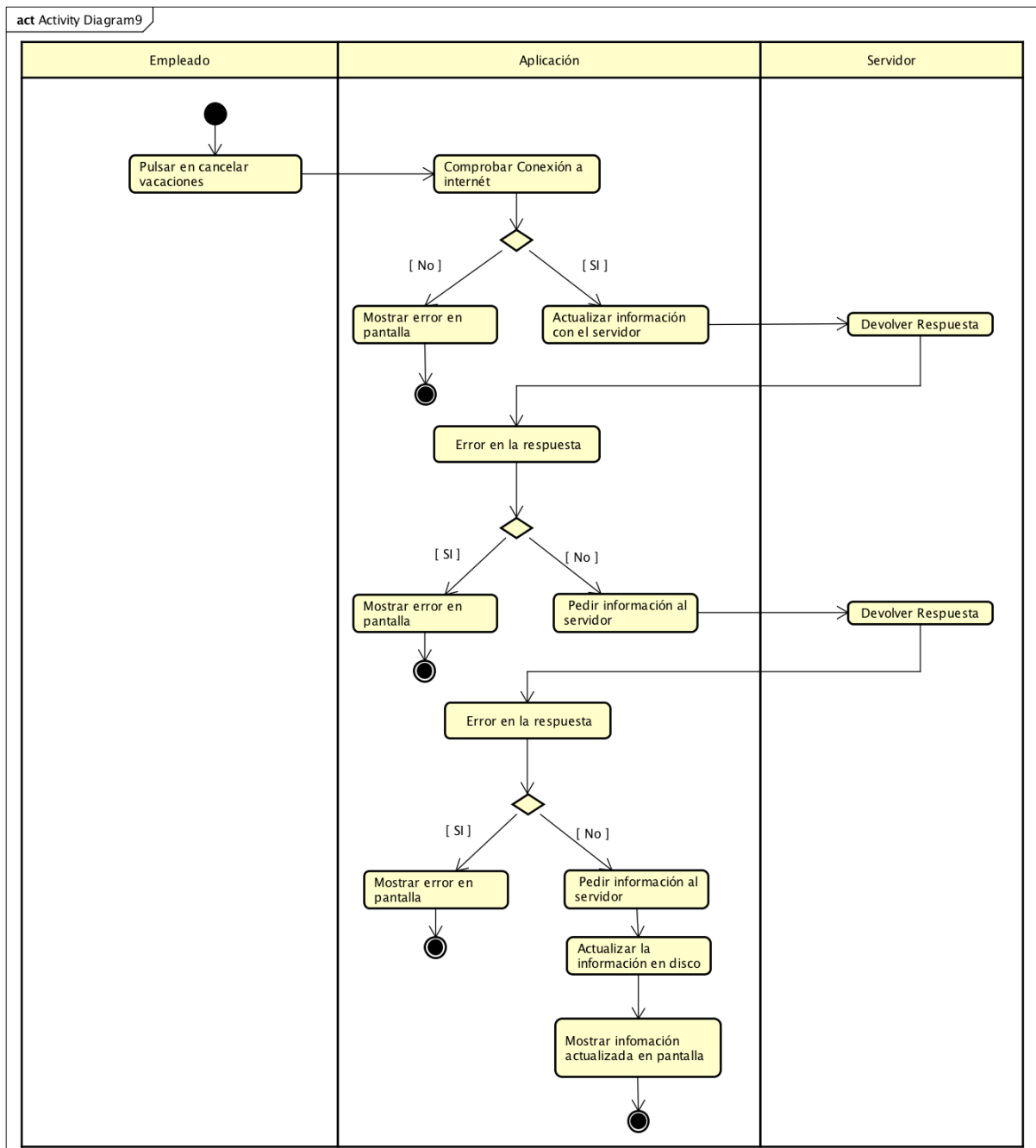


Figura 6.8: Diagrama de actividad del CU6.

6.1.6.7. CU07: Ver total avisos

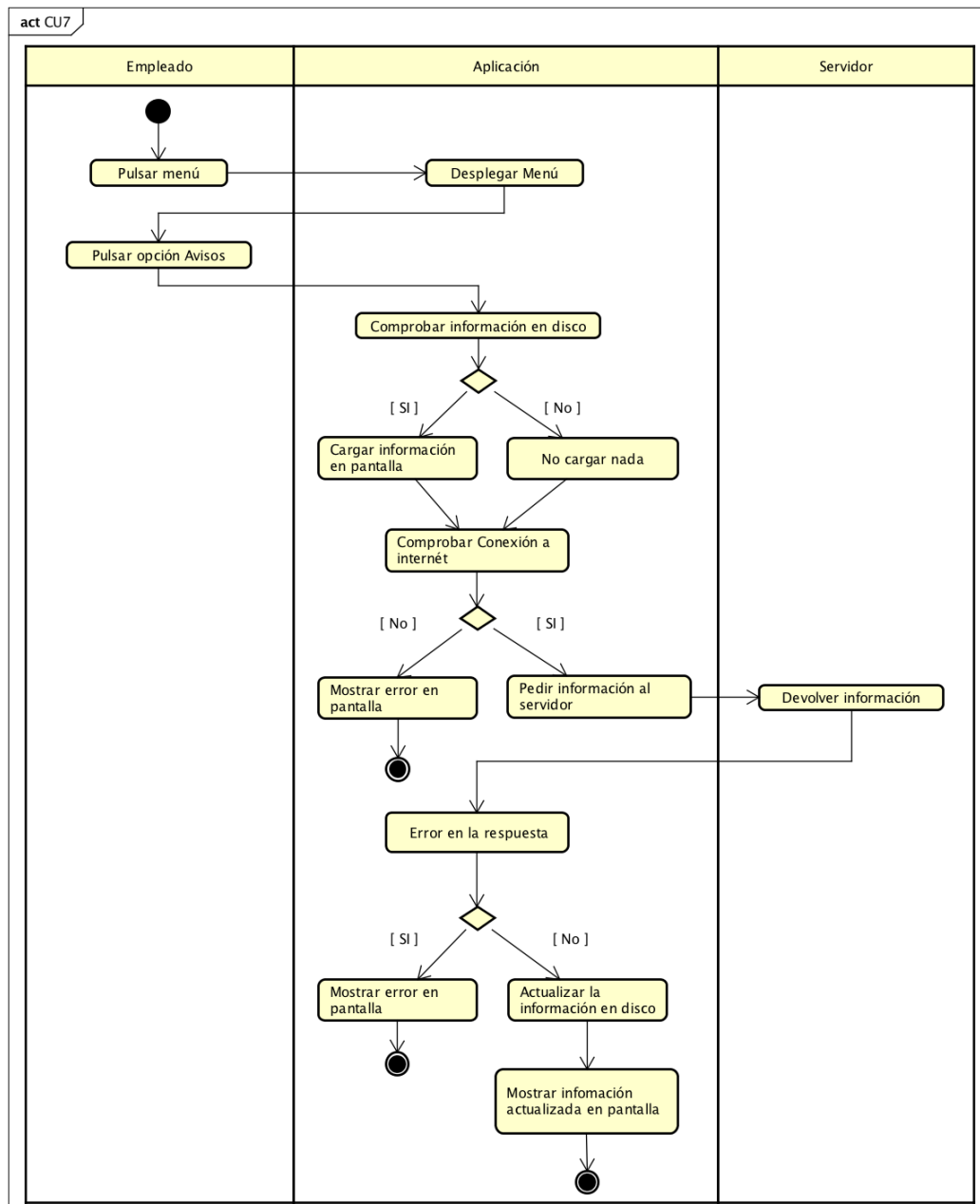


Figura 6.9: Diagrama de actividad del CU07.

6.1.6.8. CU08: Recibir notificaciones

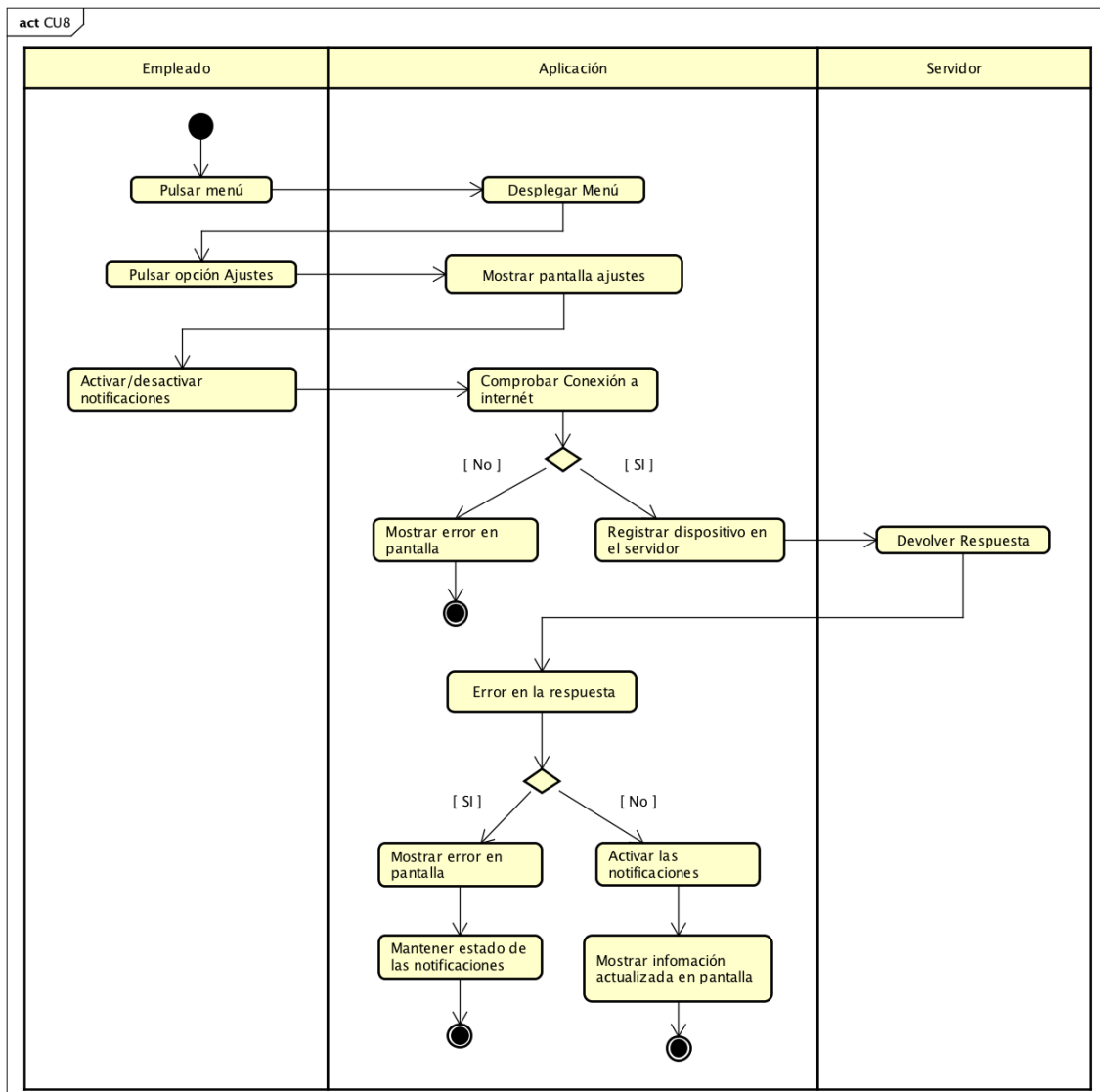


Figura 6.10: Diagrama de actividad del CU08.

6.1.6.9. CU09: Ver *Twitter* corporativo.

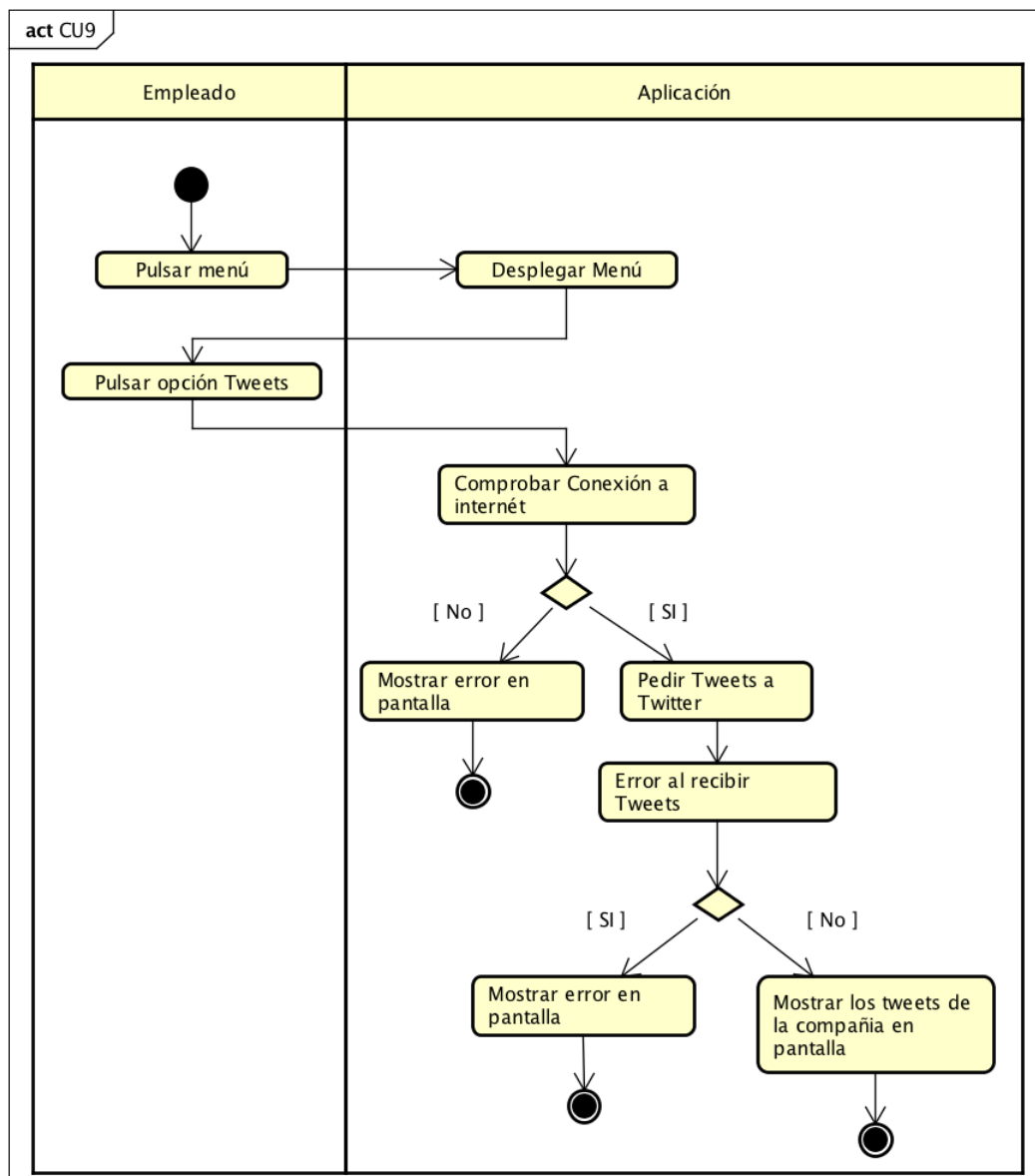


Figura 6.11: Diagrama de actividad del CU09.

6.1.6.10. CU10: Ver compañeros de trabajo.

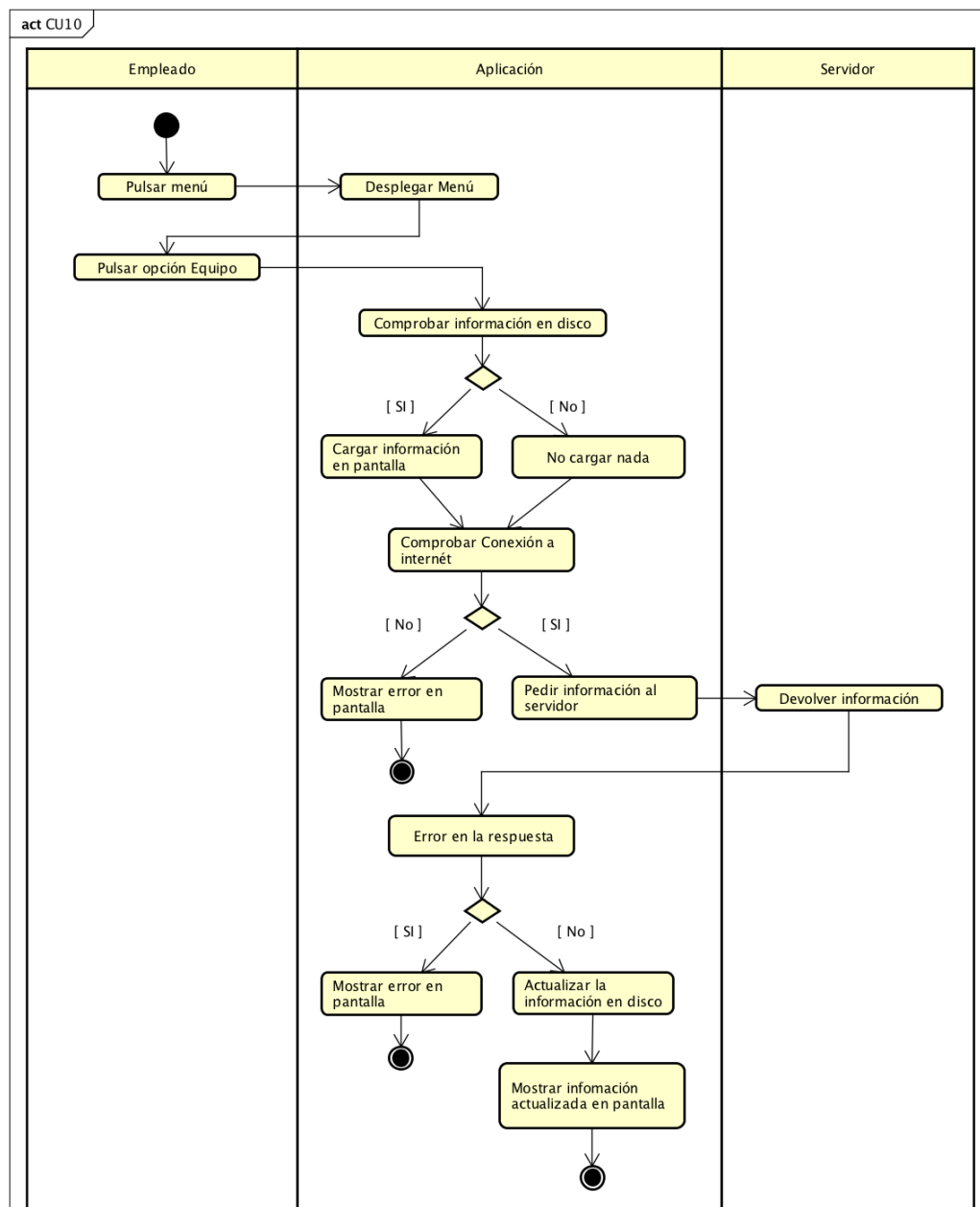


Figura 6.12: Diagrama de actividad del CU10.

6.1.6.11. CU11: Buscar compañeros de trabajo.

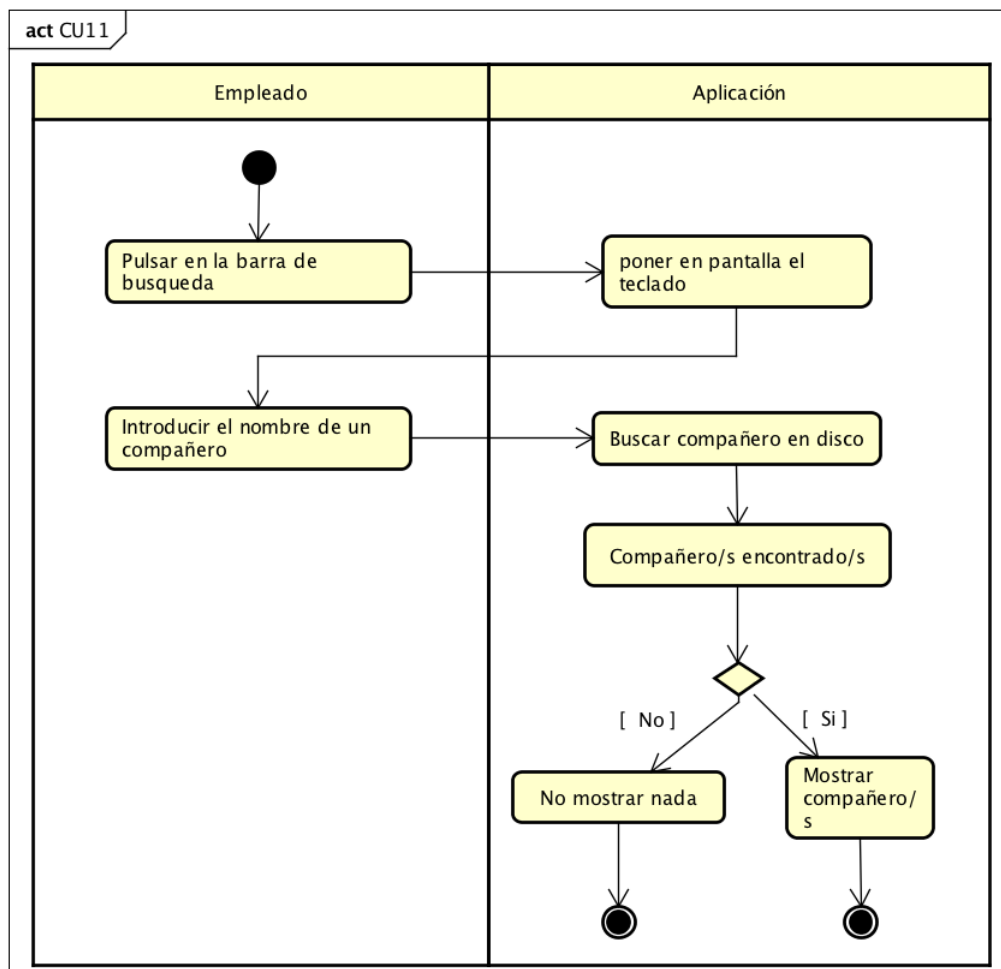


Figura 6.13: Diagrama de actividad del CU11.

Capítulo 7

Diseño

7.1. Diseño de la arquitectura

7.1.1. Patrón arquitectónico utilizado

En la actualidad existen numerosos patrones arquitectónicos que son utilizados para el desarrollo de aplicaciones iOS. Desde variaciones de la arquitectura CLEAN (propuesta por *Robert Cecil Martin*, mas conocido como *Uncle Bob*) como VIPER o CLEAN Swift, a otras más clásicas como MVVM (Modelo Vista Vista-Modelo), MVC (Modelo Vista Controlador), MVP (Modelo Vista Presentador), etc[14].

Muchas de las grandes aplicaciones que actualmente se desarrollan utilizan las arquitecturas anteriormente citadas o alguna variación de estas como es el caso de la aplicación de Uber[15].

El patrón arquitectónico que propone Apple para sus aplicaciones es el patrón MVC o Modelo Vista Controlador. El propio sistema iOS y los distintos *frameworks* que lo componen y extienden aplican este modelo arquitectónico.

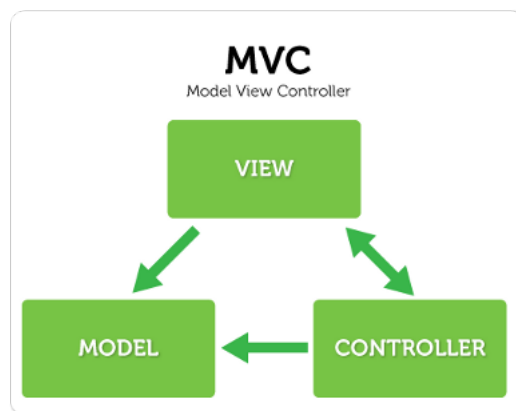


Figura 7.1: Descripción del patrón MVC

Como se ha mencionado antes, existen diversas arquitecturas que se pueden aplicar al desarrollo de una aplicación iOS. En la actualidad las grandes aplicaciones utilizan variantes de esas arquitecturas para adaptarlas a cada escenario y modelo de negocio.

Para este proyecto se eligió MVC, pero esta arquitectura, tal y como la propone Apple, presenta un potencial problema: Los controladores crecen y crecen a medida que asumen responsabilidades tanto de acceso a datos como de acceso y llamadas a la red. Por eso, después

de comentar las posibles soluciones con varios profesionales del sector y estudiar las diferentes arquitecturas, se llegó a la conclusión de usar una variación de MVC[16][17].

En esta variación se desacoplará el acceso y llamadas a la red de los controladores para evitar que estos crezcan demasiado obteniendo así una arquitectura más clara. Este no será el único cambio ya que, aprovechando las capacidades tanto de Swift como del SDK de iOS, se utilizará una comunicación asíncrona para avisar a los controladores cuando tienen los datos actualizados, mostrando así la última información al usuario de manera óptima.

7.1.2. Arquitectura general

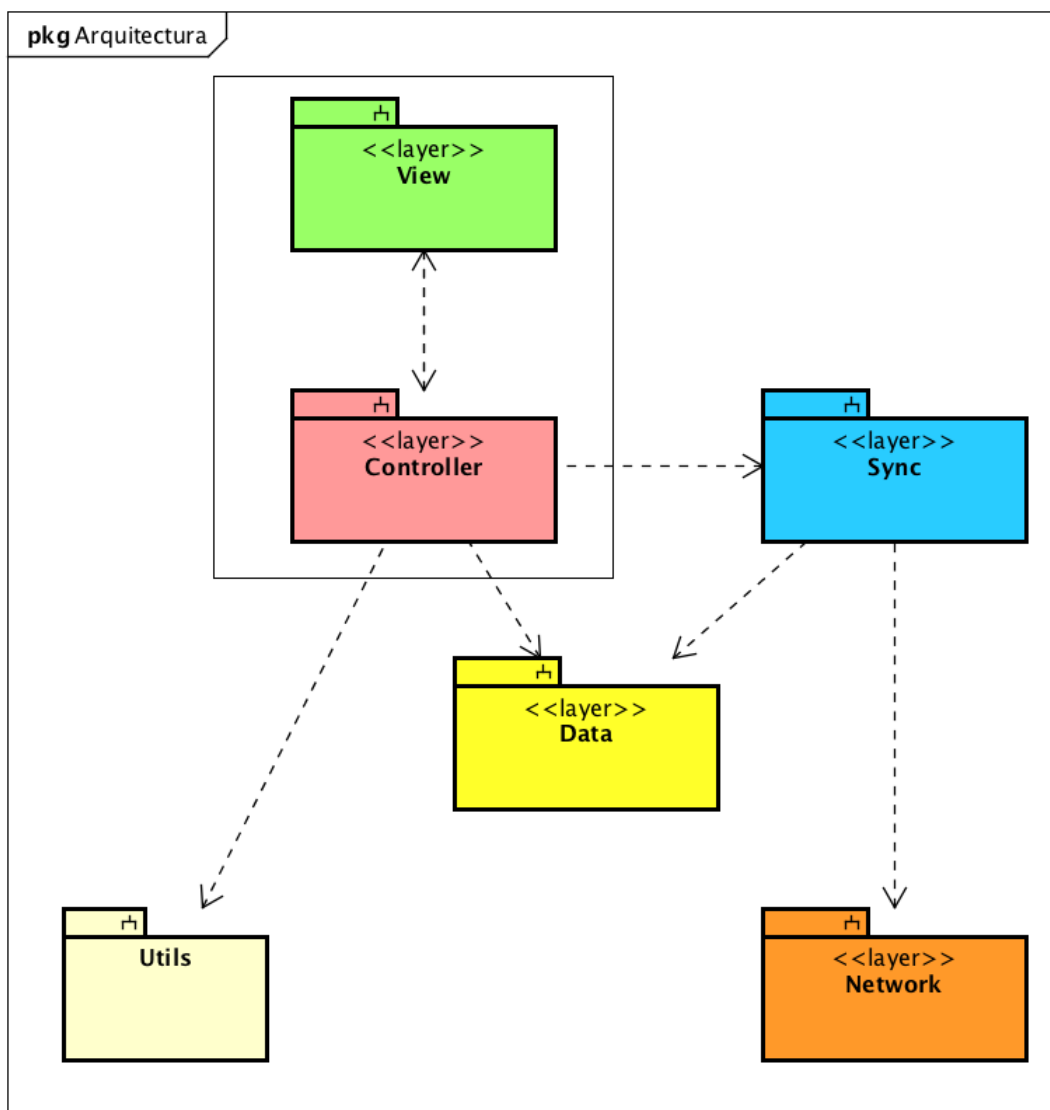


Figura 7.2: Arquitectura general de la aplicación

A continuación se desglosan las capas de la arquitectura:

- **Vista y Controlador:** Están muy acoplados, la vista está embebida en el controlador de vista.
- **Sync:** Se encarga de realizar las peticiones a la clase de red, recibir la respuesta , actualizar los datos recibidos con el modelo y avisar , de manera asíncrona al controlador de vista de que los datos han sido actualizados y puede mostrarlo en pantalla.
- **Data:** En esta capa se encuentra el modelo y los *managers* que se encargan de realizar las operaciones *CRUD*
- **Network:** Es la encargada de realizar las llamadas HTTP sobre la API REST que proporciona el servidor. La respuesta es pasada a la capa Sync para que decida que hacer.

El flujo general del funcionamiento de la arquitectura sería:

1. El usuario realiza una acción en la vista
2. Como la vista y el controlador están muy acoplados, se realiza una acción en el controlador
 - a) Si el controlador necesita información del disco, habla con la capa Data.
 - b) Si el controlador necesita información desde el servidor, habla con la capa Sync y queda a la espera.
3. La capa Sync realiza una llamada asíncrona a la capa Network pidiéndole que realice una petición a la red.
4. La capa Network hace una petición de red al servidor de manera asíncrona y cuando finaliza, avisa a la capa Sync y la devuelve la respuesta del servidor.
5. La capa Sync dependiendo de si ha habido error o no, se comunica con la capa Data y actualiza los datos del disco. A continuación de manera asíncrona avisa al controlador que había quedado a la espera.
6. El controlador recibe de manera asíncrona el aviso de la capa Sync, dispara una función y hace lo correspondiente.

7.1.3. Descripción modular

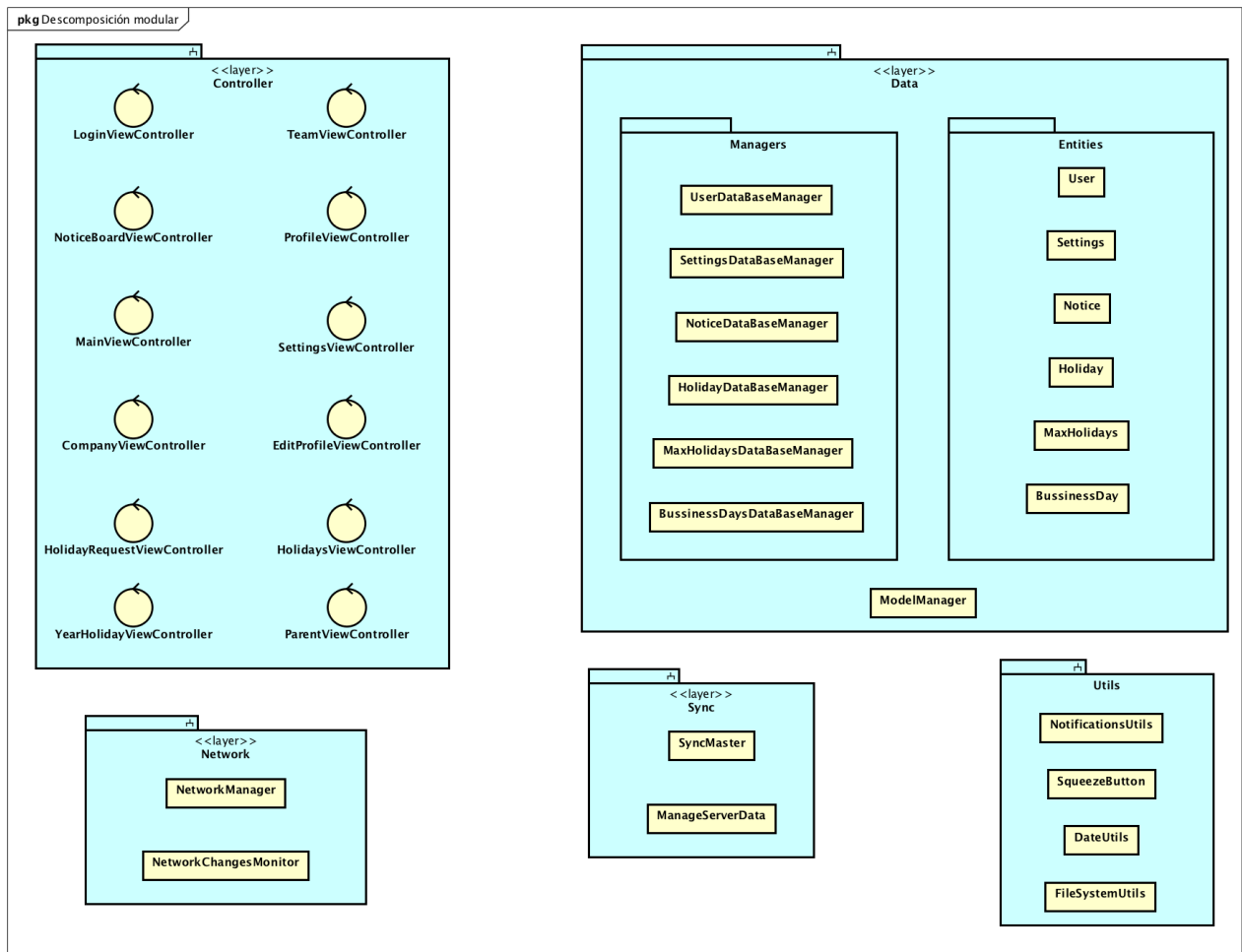


Figura 7.3: Descomposición modular de la aplicación

7.1.3.1. Descripción de la descomposición modular

El objetivo de la figura 7.3 es representar de manera gráfica la división del sistema en partes diferenciadas, enumerando las principales clases implicadas en cada una de las partes, sin entrar en detalle de las relaciones que existen entre ellas.

7.1.4. Diseño de los módulos o subsistemas

En este apartado se detallan cada uno de los módulos identificados en la figura 7.3, con las relaciones existentes entre las clases que lo forman. El único módulo que no se detallará será el de *utils*.

7.1.4.1. Diseño detallado del módulo *Controller*

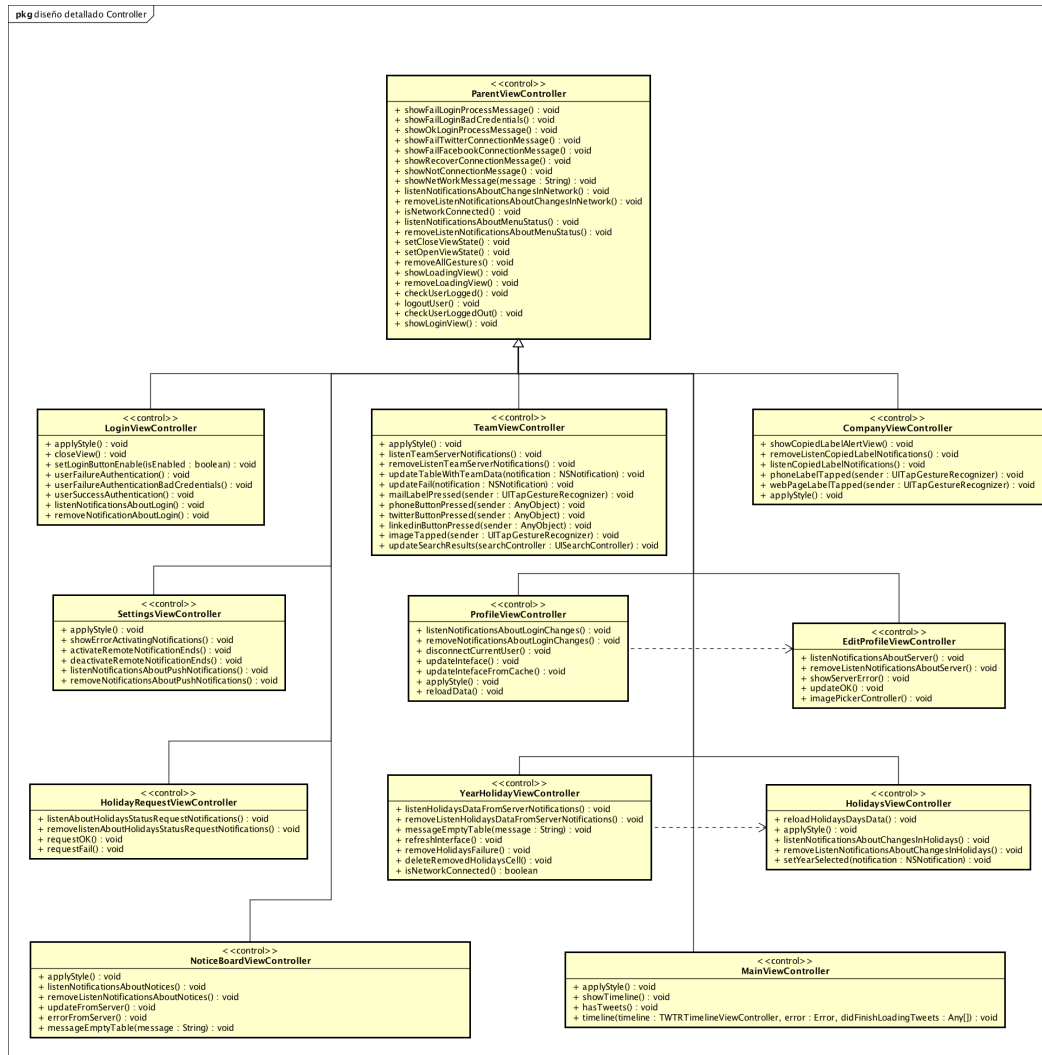


Figura 7.4: Diseño detallado del módulo *Controller*

7.1.4.2. Diseño detallado del módulo *Sync*

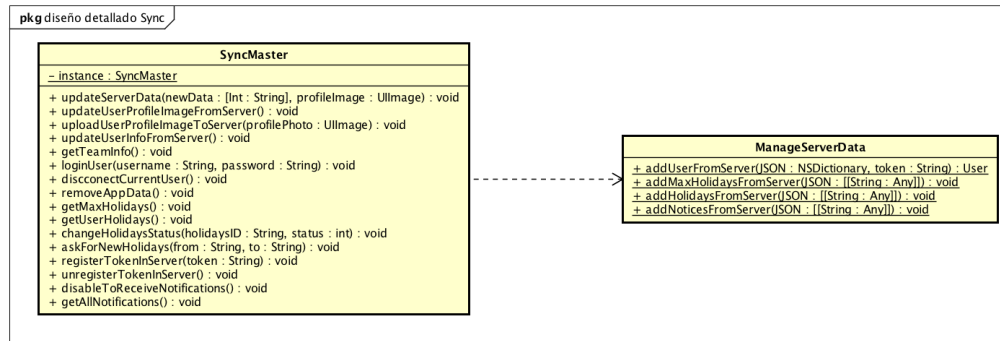


Figura 7.5: Diseño detallado del módulo *Sync*

7.1.4.3. Diseño detallado del módulo *Network*

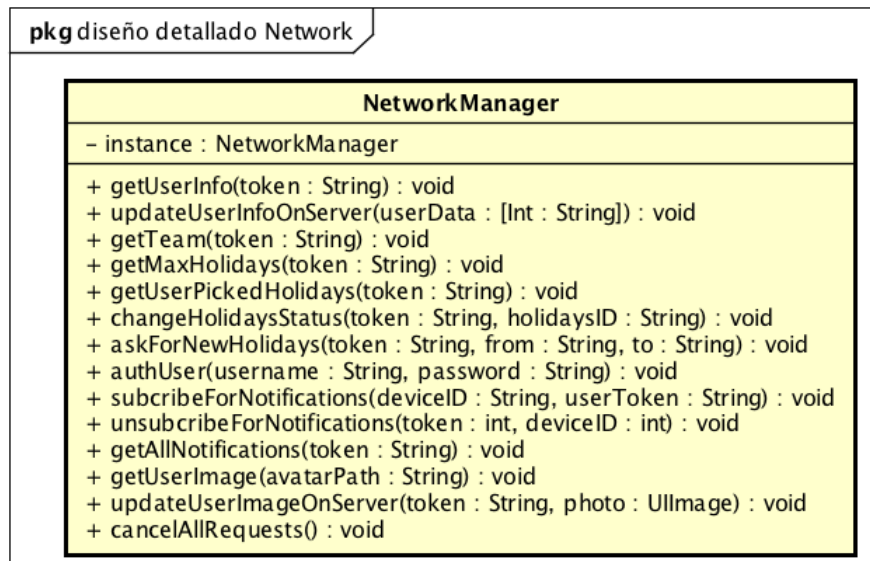


Figura 7.6: Diseño detallado del módulo *Network*

7.1.4.4. Diseño detallado del módulo *Data*

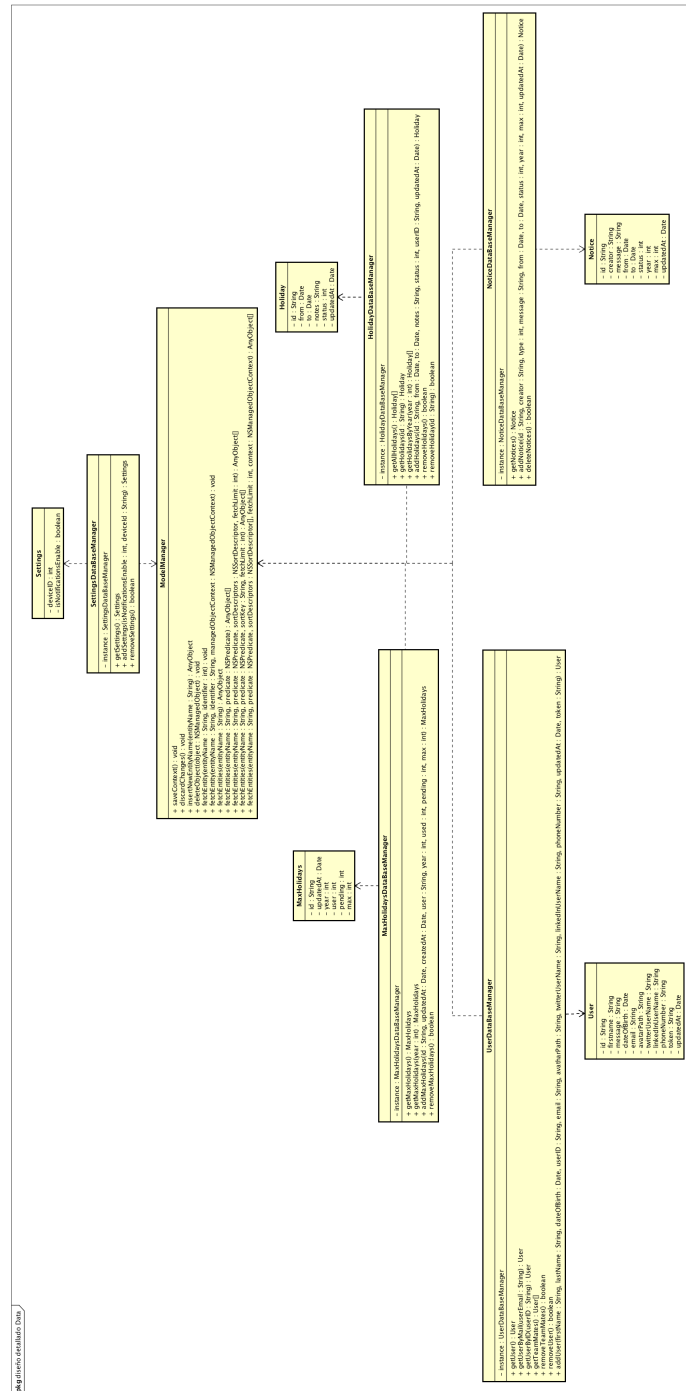


Figura 7.7: Diseño detallado del módulo *Data*

7.1.5. Relación entre módulos

En esta apartado se muestran las relaciones entre las diferentes clases de cada módulo. Para facilitar la comprensión, se ha dividido por casos de uso.

7.1.5.1. CU01: Identificarse

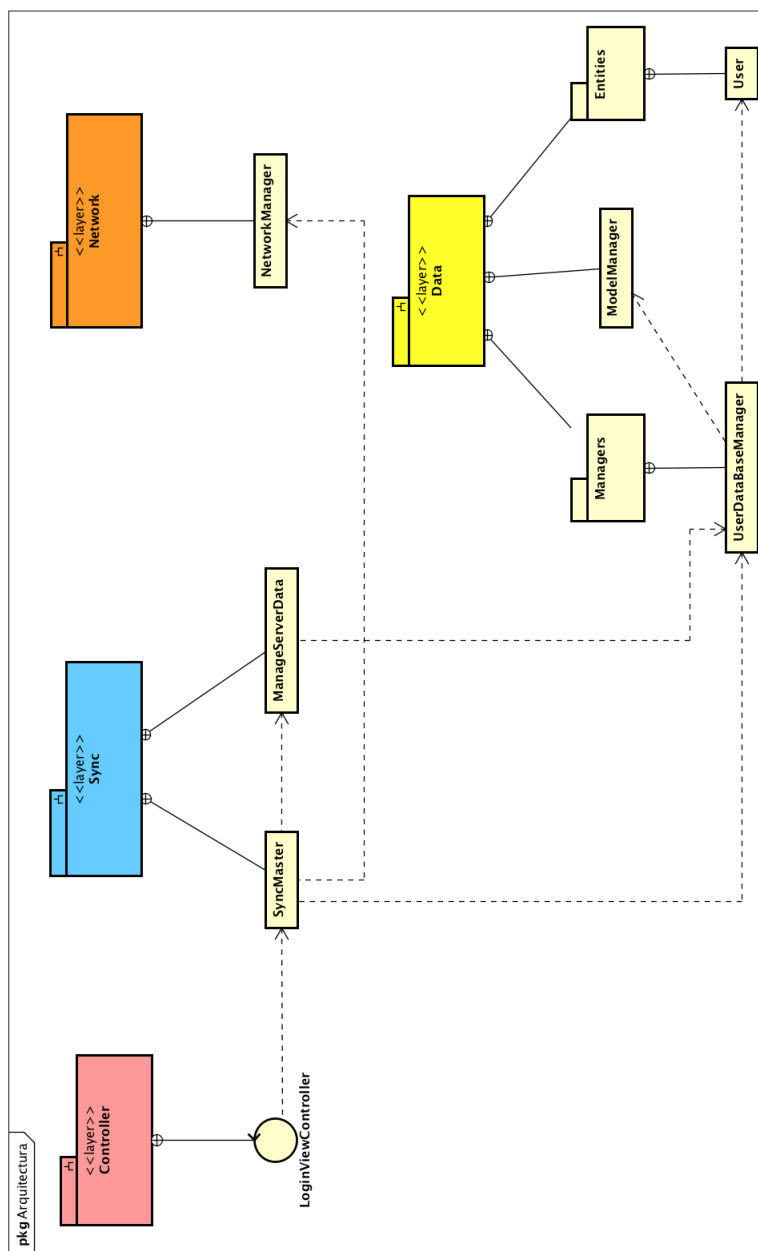


Figura 7.8: Relación entre módulos para el CU01

7.1.5.2. CU02: Ver perfil

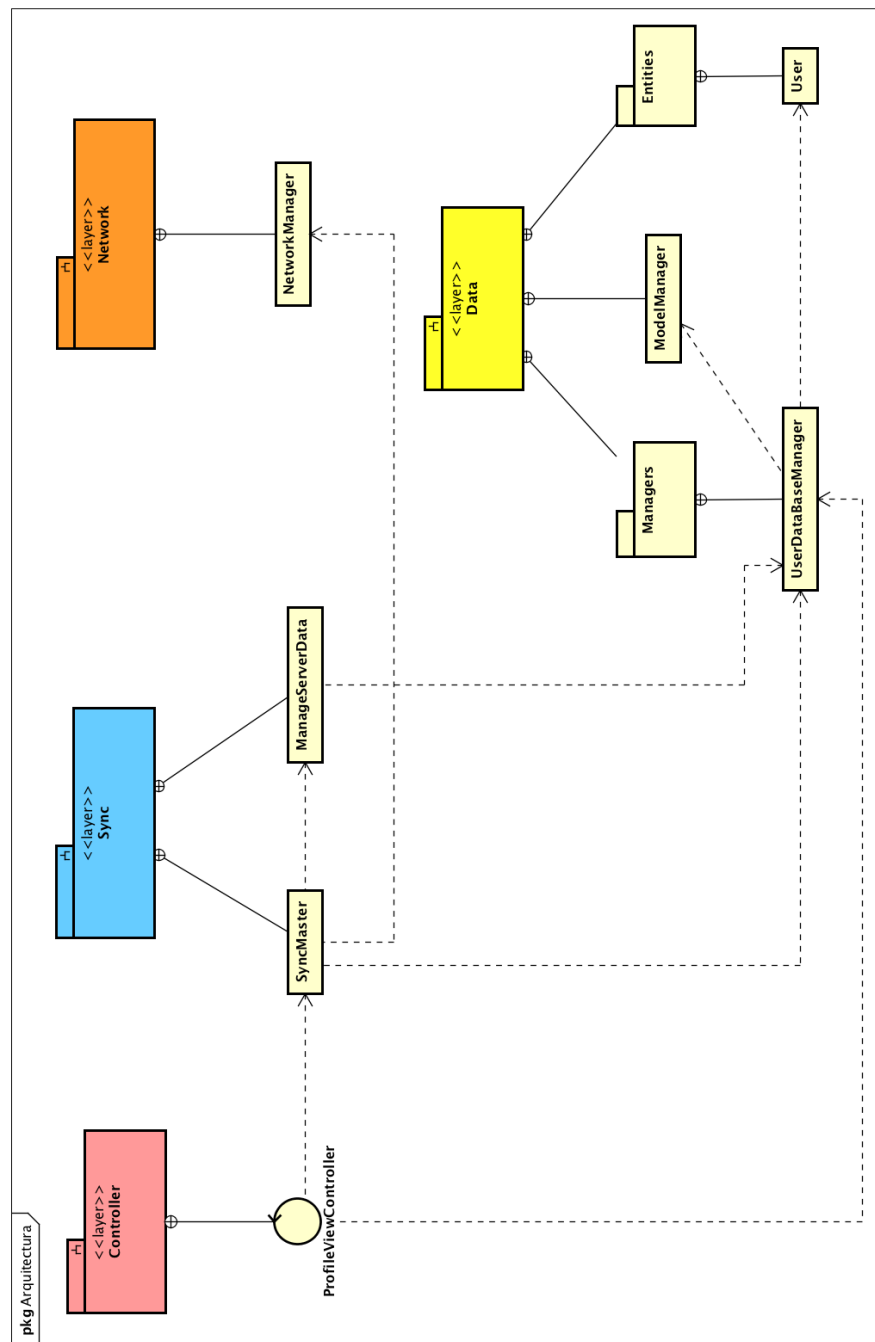


Figura 7.9: Relación entre módulos para el CU02

7.1.5.3. CU03: Modificar perfil

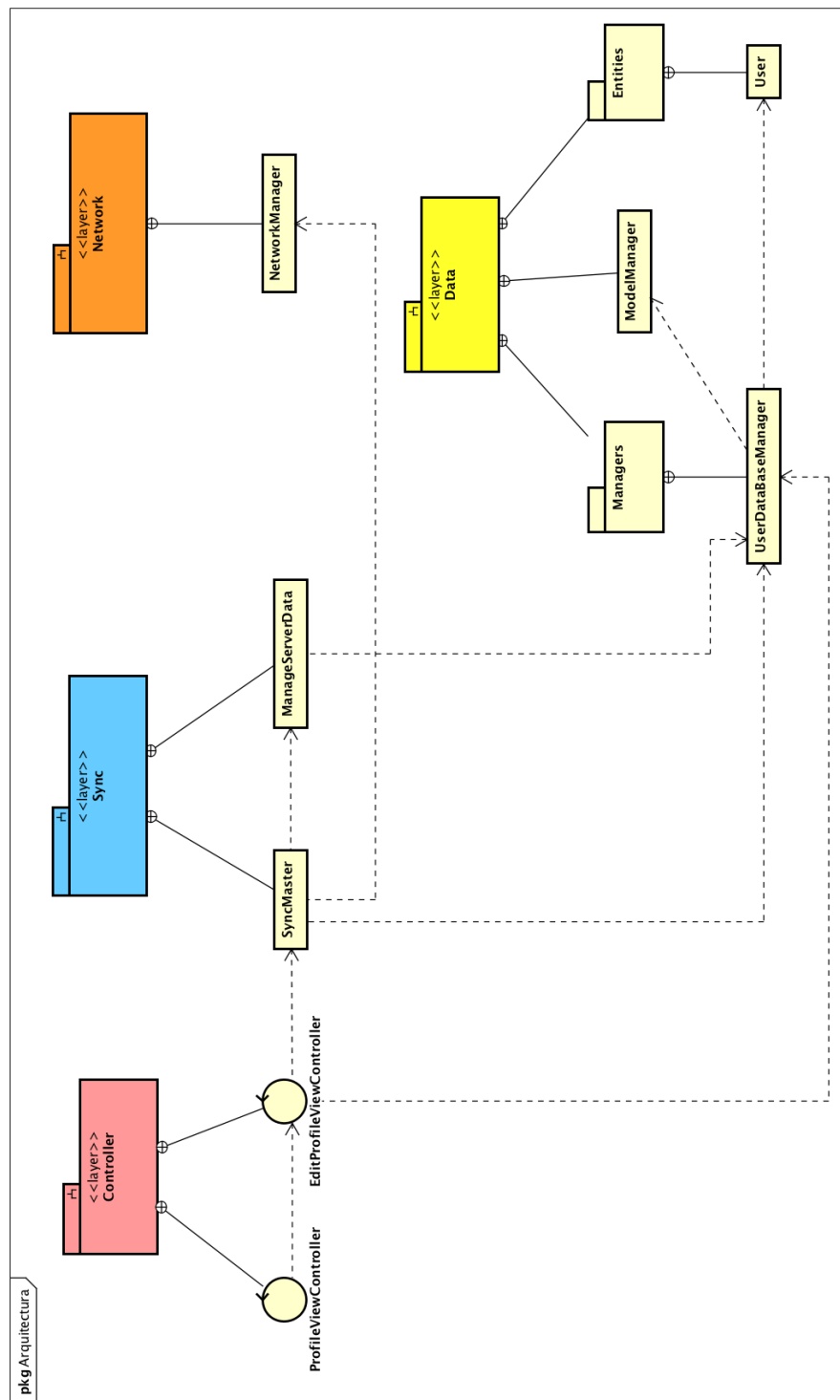


Figura 7.10: Relación entre módulos para el CU03

7.1.5.4. CU04: Ver vacaciones restantes

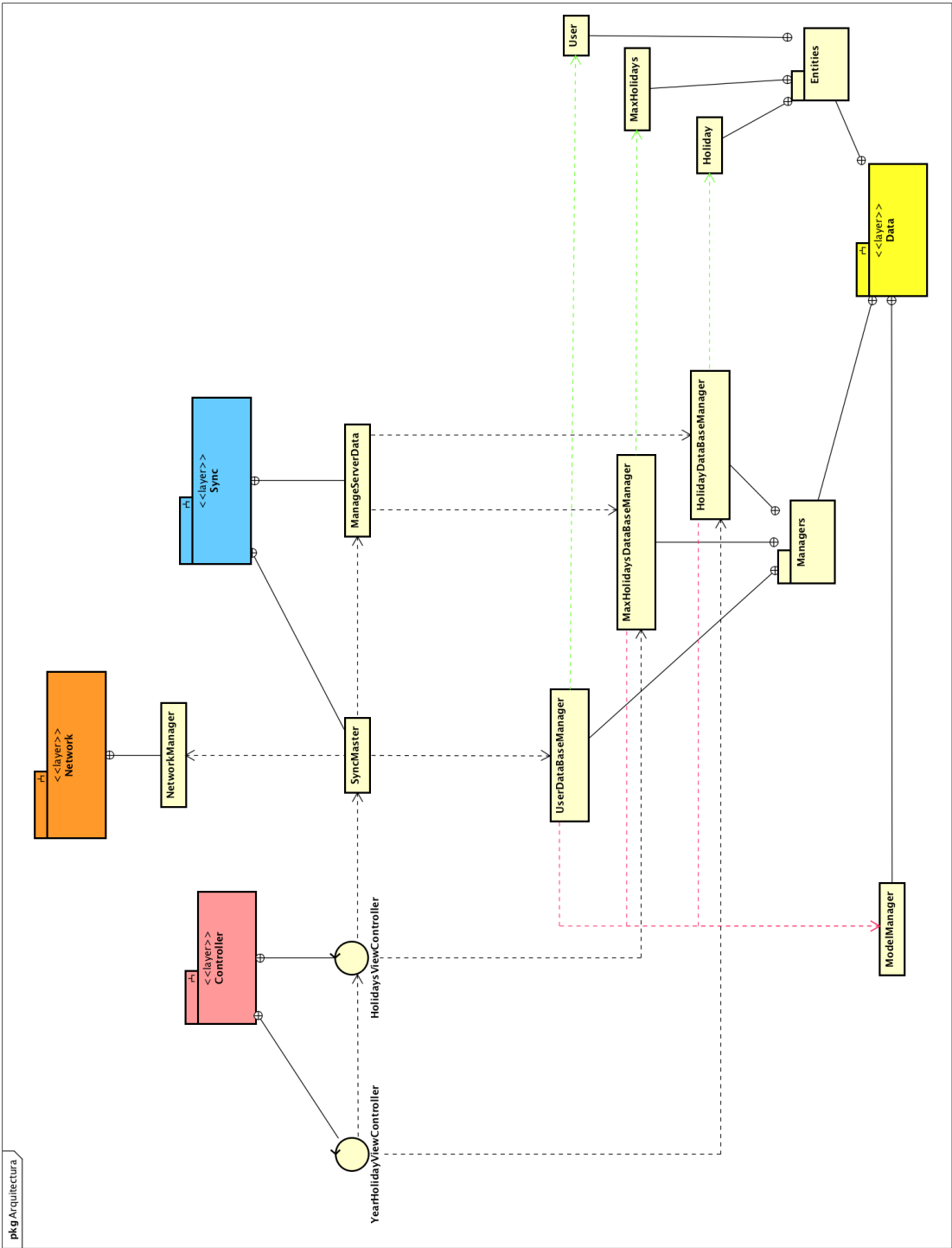


Figura 7.11: Relación entre módulos para el CU04

7.1.5.5. CU05: Pedir Vacaciones

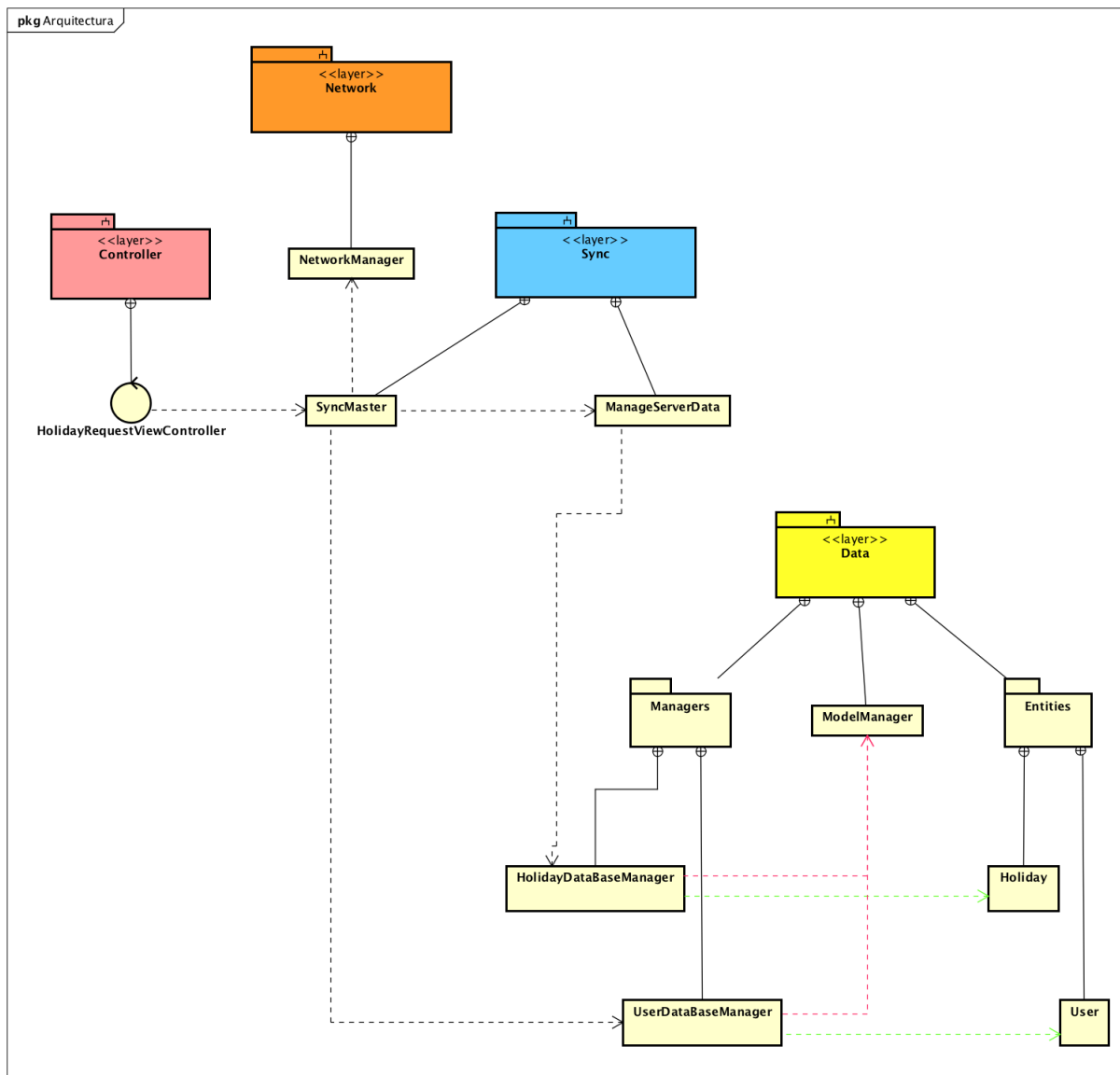


Figura 7.12: Relación entre módulos para el CU05

7.1.5.6. CU06: Modificar estado vacaciones

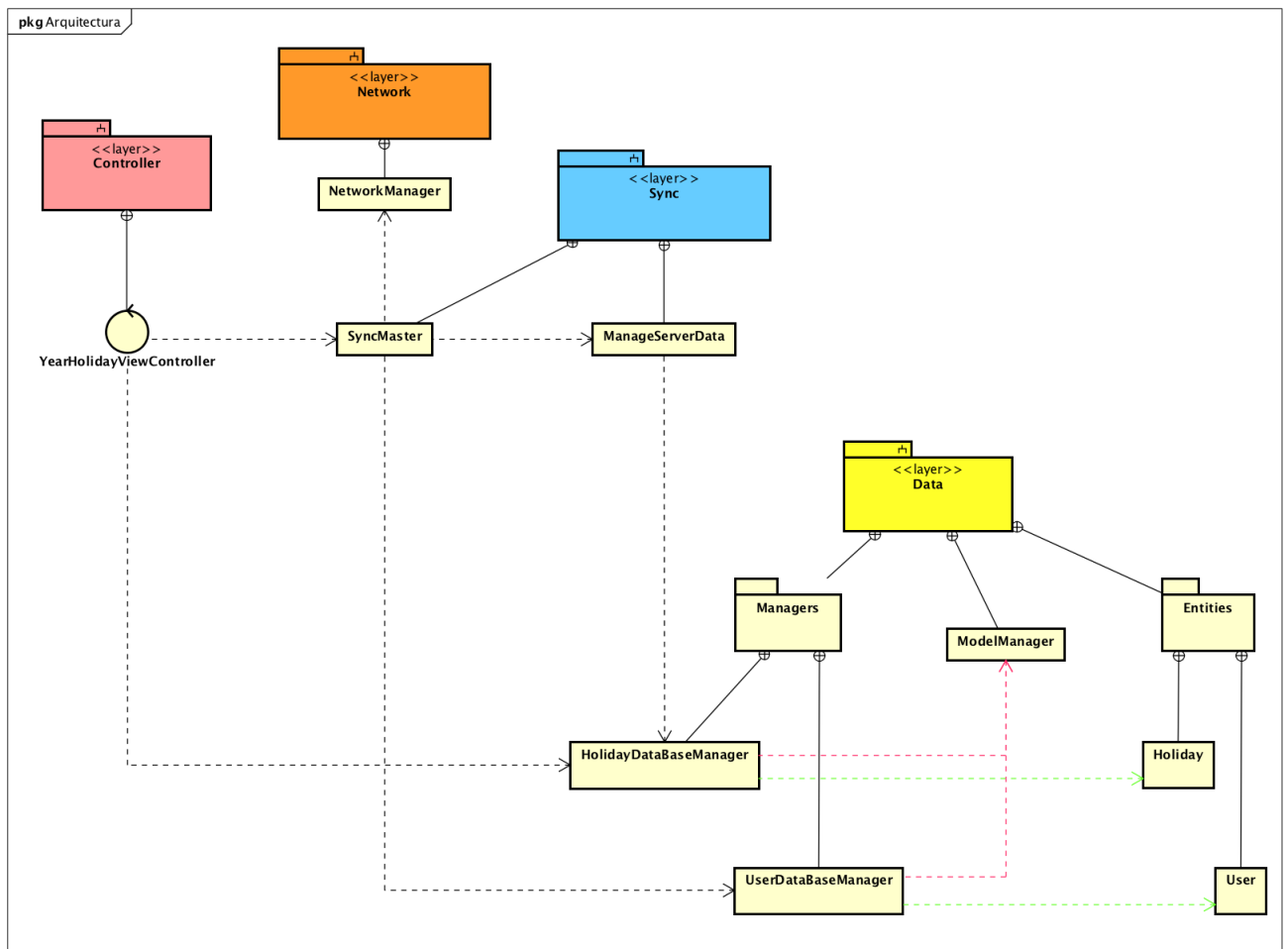


Figura 7.13: Relación entre módulos para el CU06

7.1.5.7. CU07: Ver total avisos

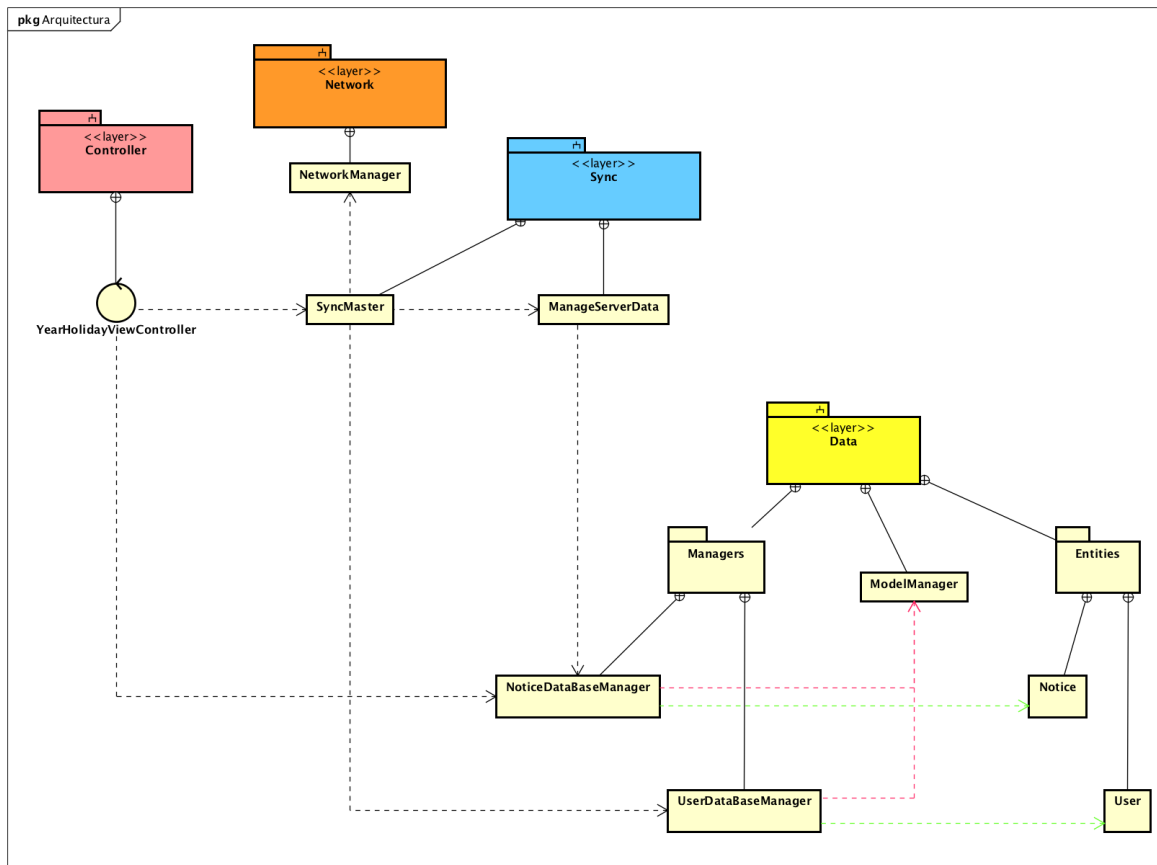


Figura 7.14: Relación entre módulos para el CU07

7.1.5.8. CU10: Ver compañeros de trabajo

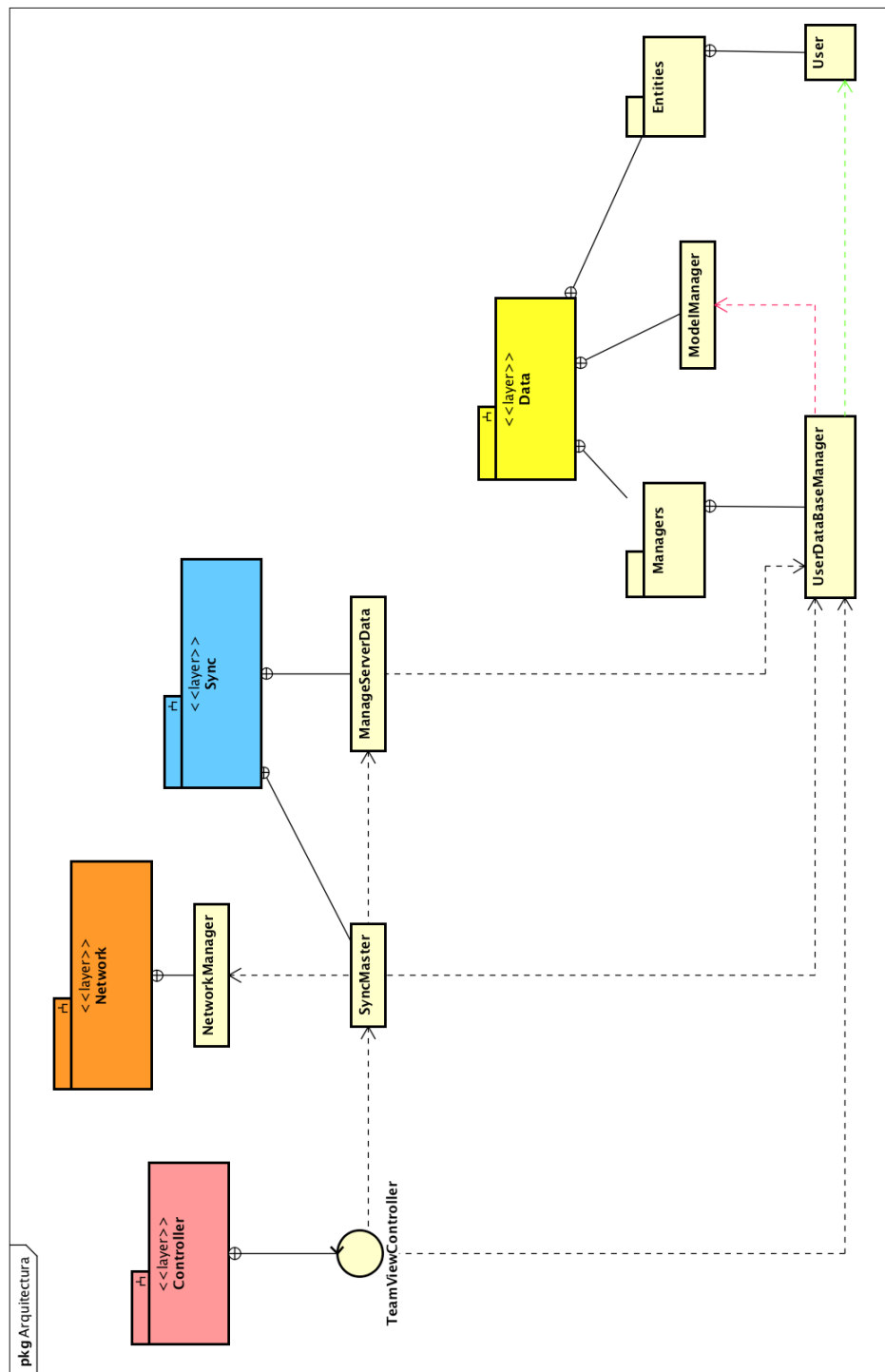


Figura 7.15: Relación entre módulos para el CU10

7.1.6. Realización en diseño de los Casos de Uso

A continuación se dará la realización en diseño de los casos de uso más significativos del sistema.

En los diagramas que se mostrarán a continuación, la parte de acceso a red y a datos se realiza siempre de la misma manera, por lo que solamente se ha desgranado un par de ejemplos para no repetir diagramas y exponer con más claridad el funcionamiento de la aplicación.

En estos diagramas se muestra el uso de una clase desconocida hasta ahora, como es la clase *UserDefaults*. Esta clase la proporciona el sistema operativo y su utilidad es almacenar pares clave-valor que puedan ser utilizados en la aplicación. En esta aplicación esta característica se utilizará para almacenar el valor del email del usuario que se identifica en la aplicación para poder realizar la búsqueda y diferenciación de entre los demás usuarios. También para poder comprobar si un usuario está identificado o no.

7.1.6.1. Detalle de SaveUserInDataBase

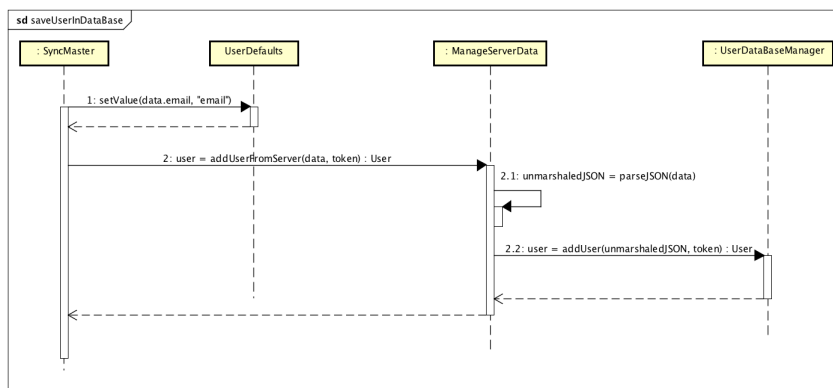


Figura 7.16: Detalla en diseño de SaveUserInDataBase

7.1.6.2. Detalle de GetUserFromDataBase

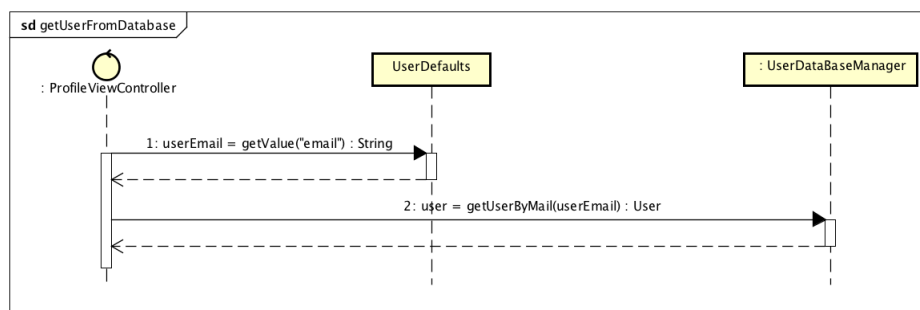


Figura 7.17: Detalla en diseño de GetUserFromDataBase

7.1.6.3. CU01: Identificarse

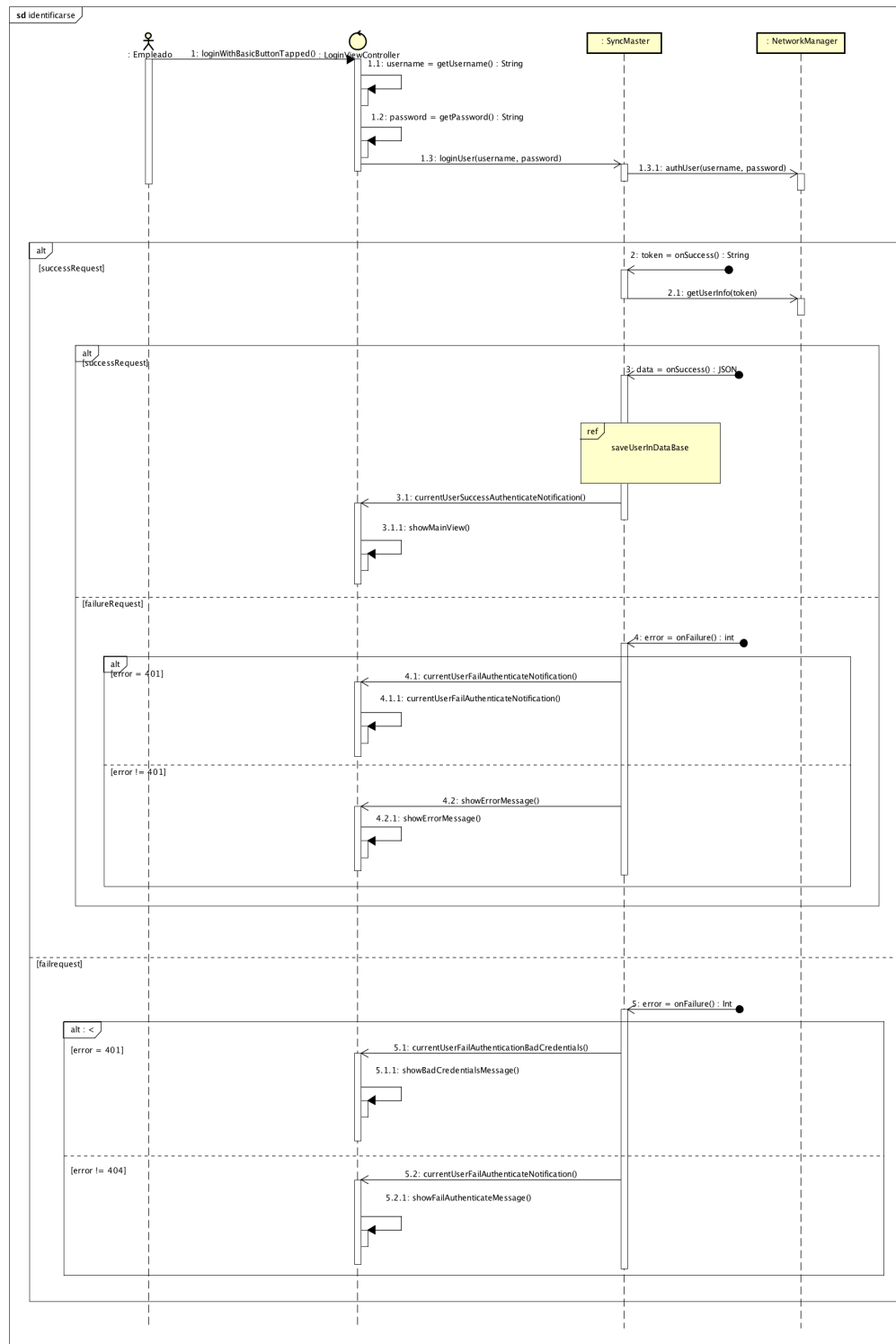


Figura 7.18: Relación en diseño del CU01

7.1.6.4. CU02: Ver perfil

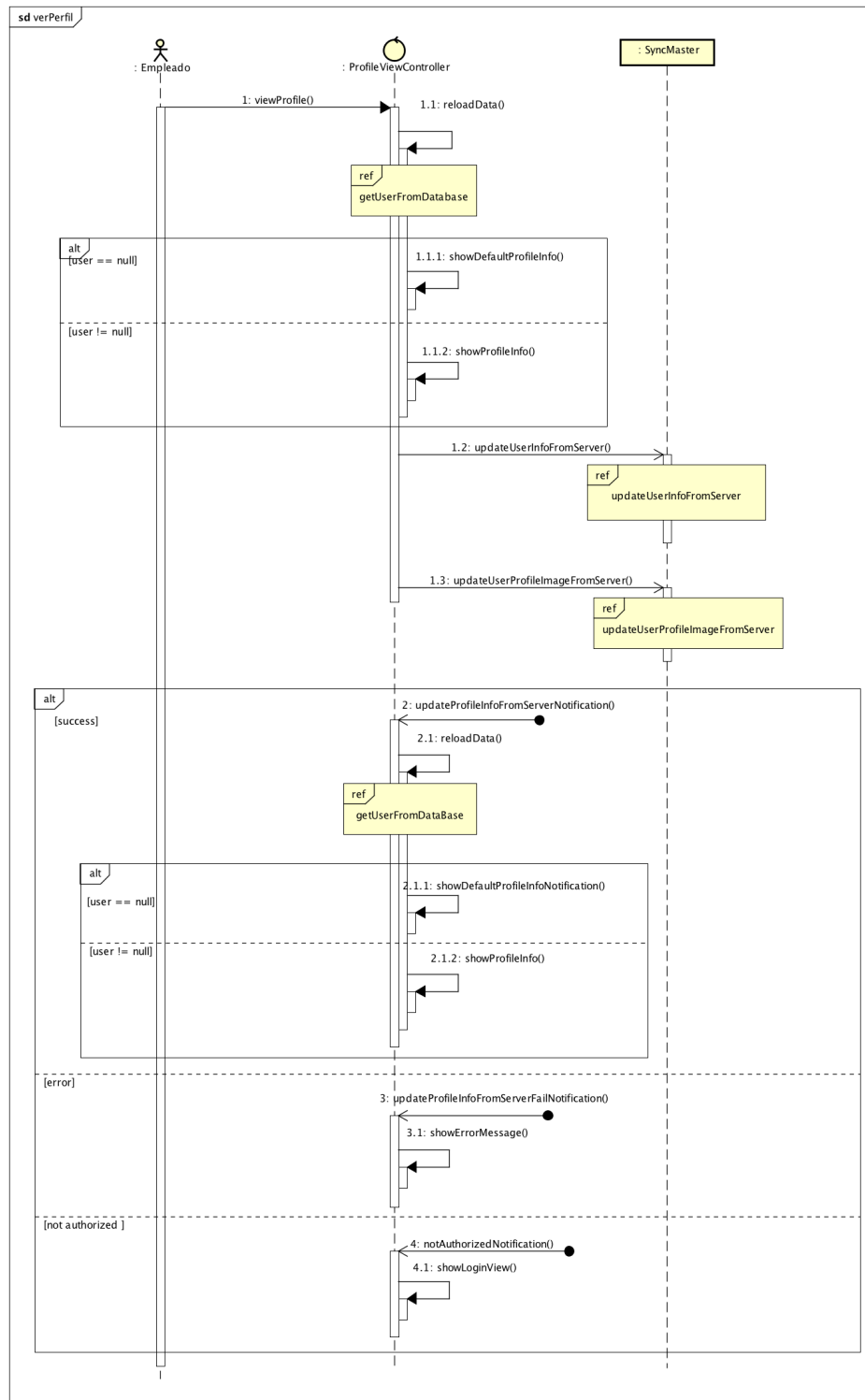


Figura 7.19: Relación en diseño del CU02

7.1.6.5. Detalle de updateUserInfoFromServer

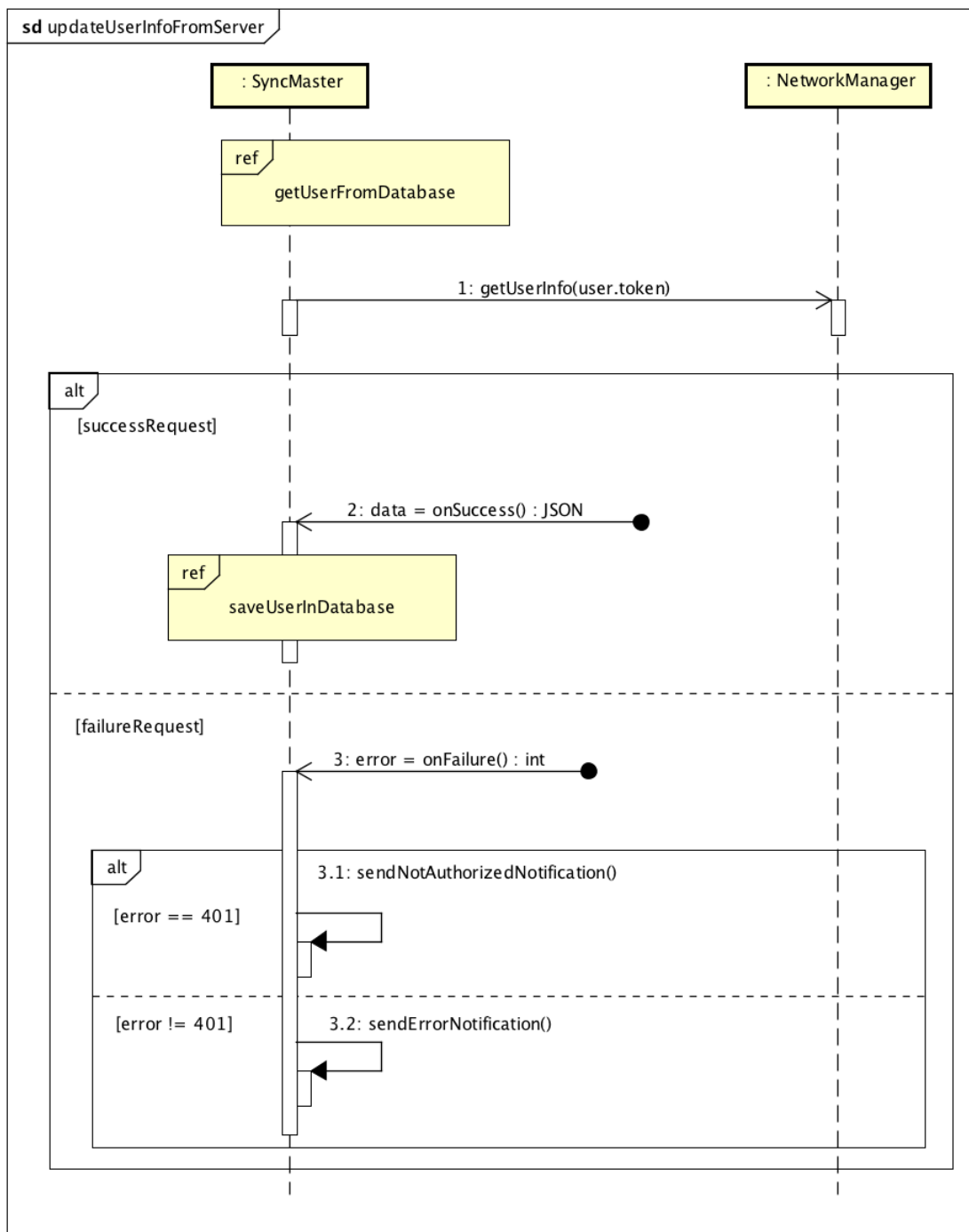


Figura 7.20: Detalla en diseño de updateUserInfoFromServer

7.1.6.6. CU03: Editar perfil

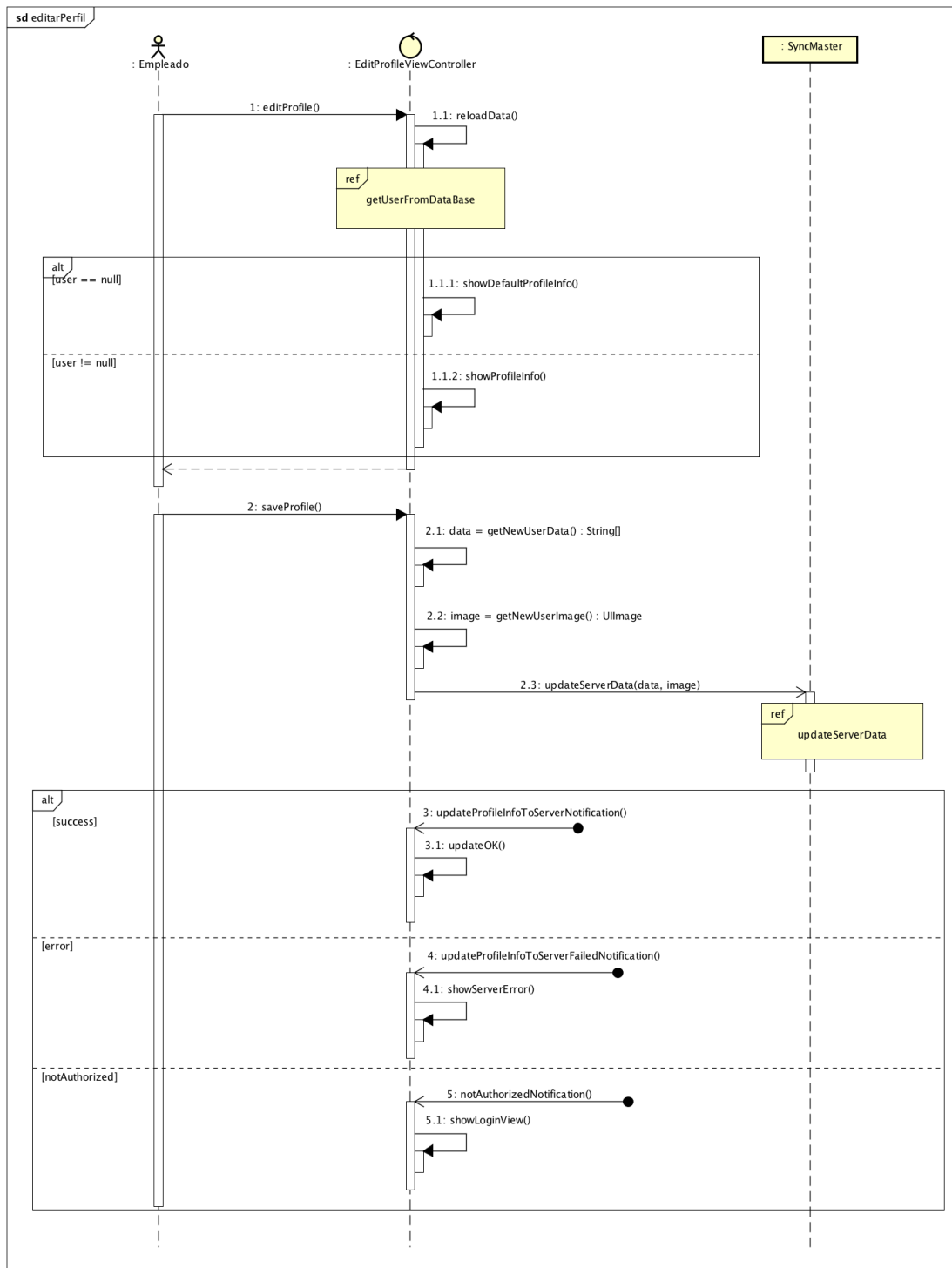


Figura 7.21: Relación en diseño del CU03

7.1.6.7. CU04: Ver vacaciones

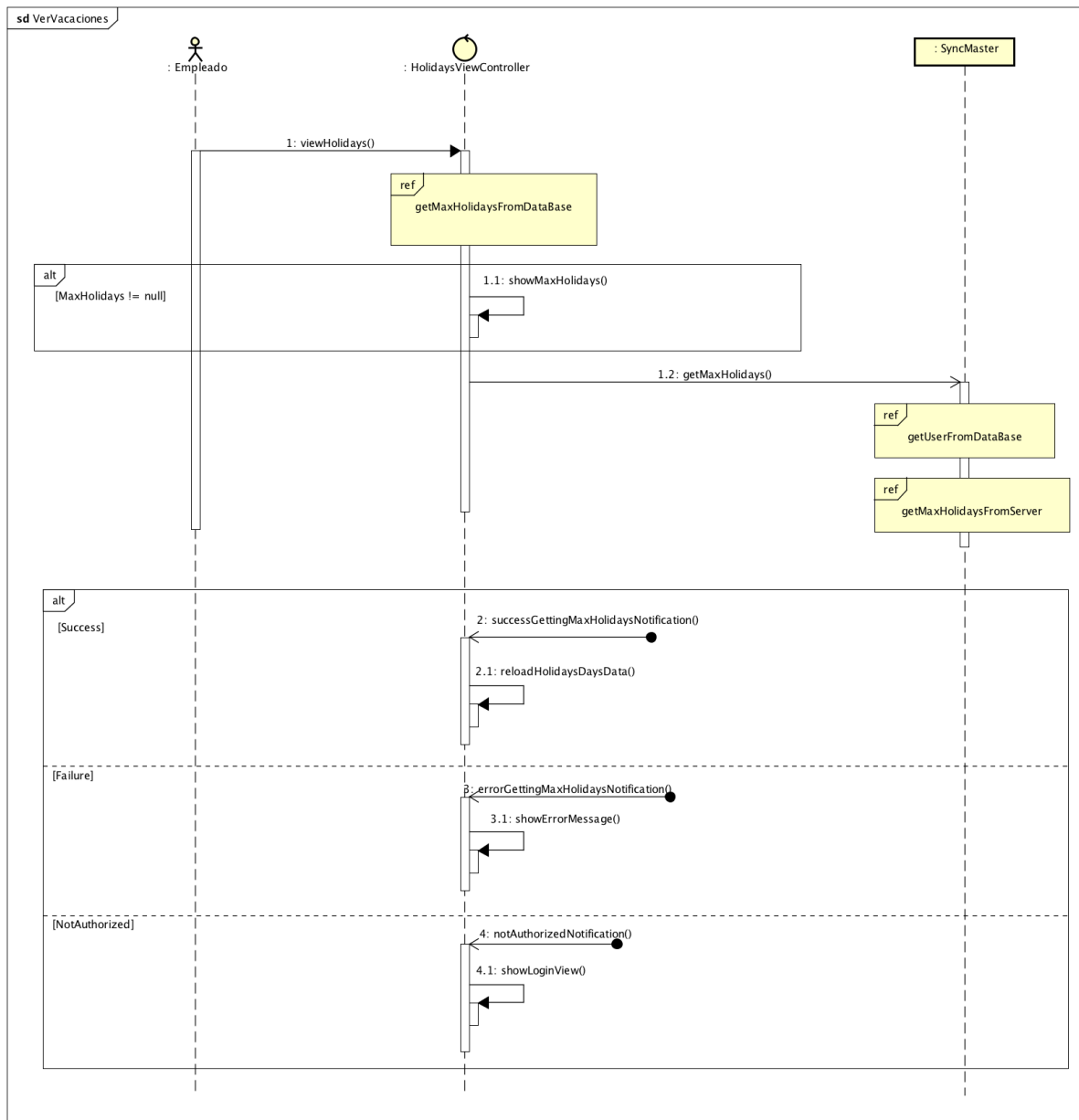


Figura 7.22: Relación en diseño del CU04

7.1.6.8. CU05: Pedir vacaciones

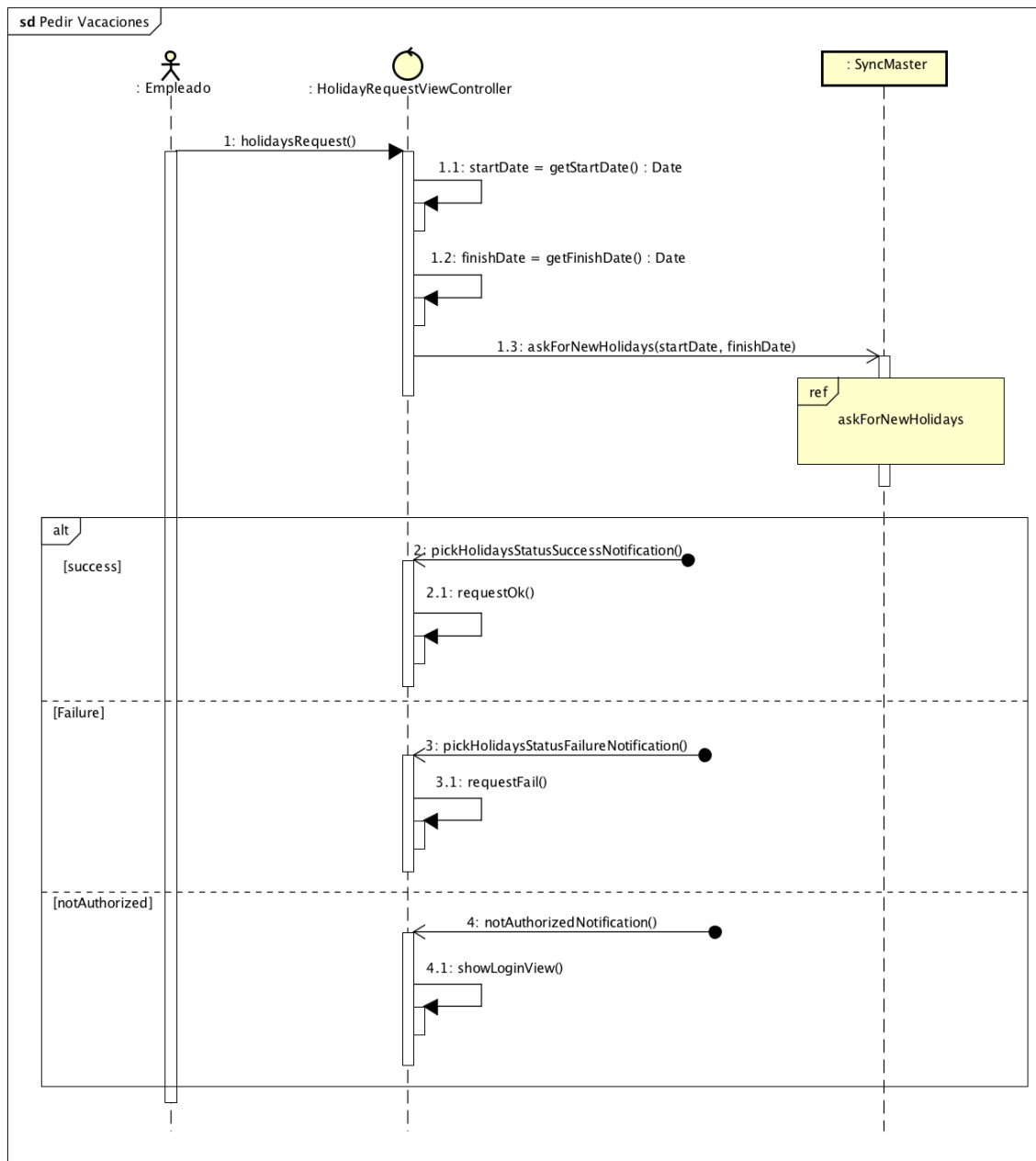


Figura 7.23: Relación en diseño del CU05

7.1.6.9. CU05: Pedir vacaciones

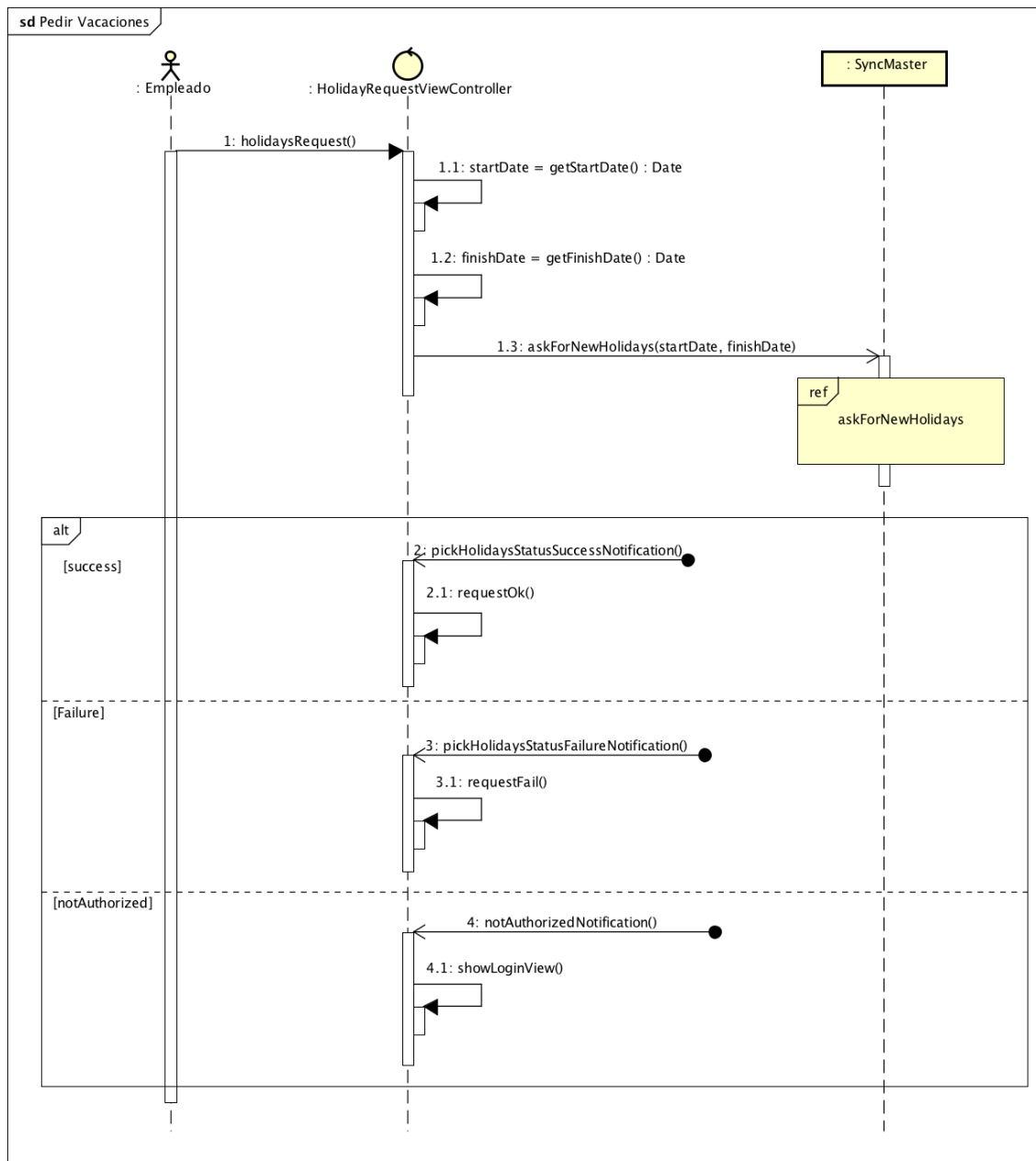


Figura 7.24: Relación en diseño del CU05

7.1.6.10. CU06: Modificar vacaciones

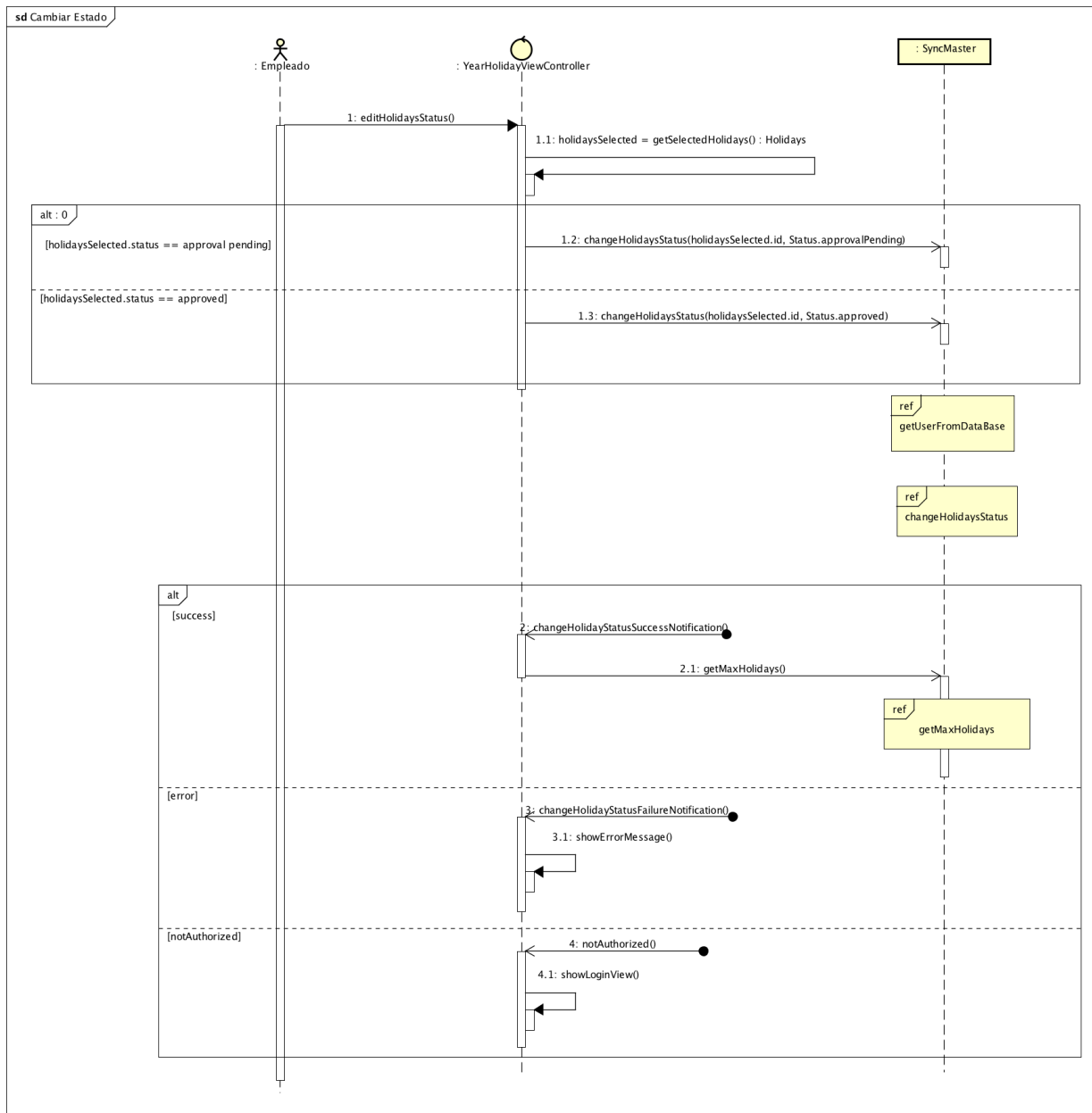


Figura 7.25: Relación en diseño del CU06

7.1.6.11. CU010: Ver compañeros de trabajo

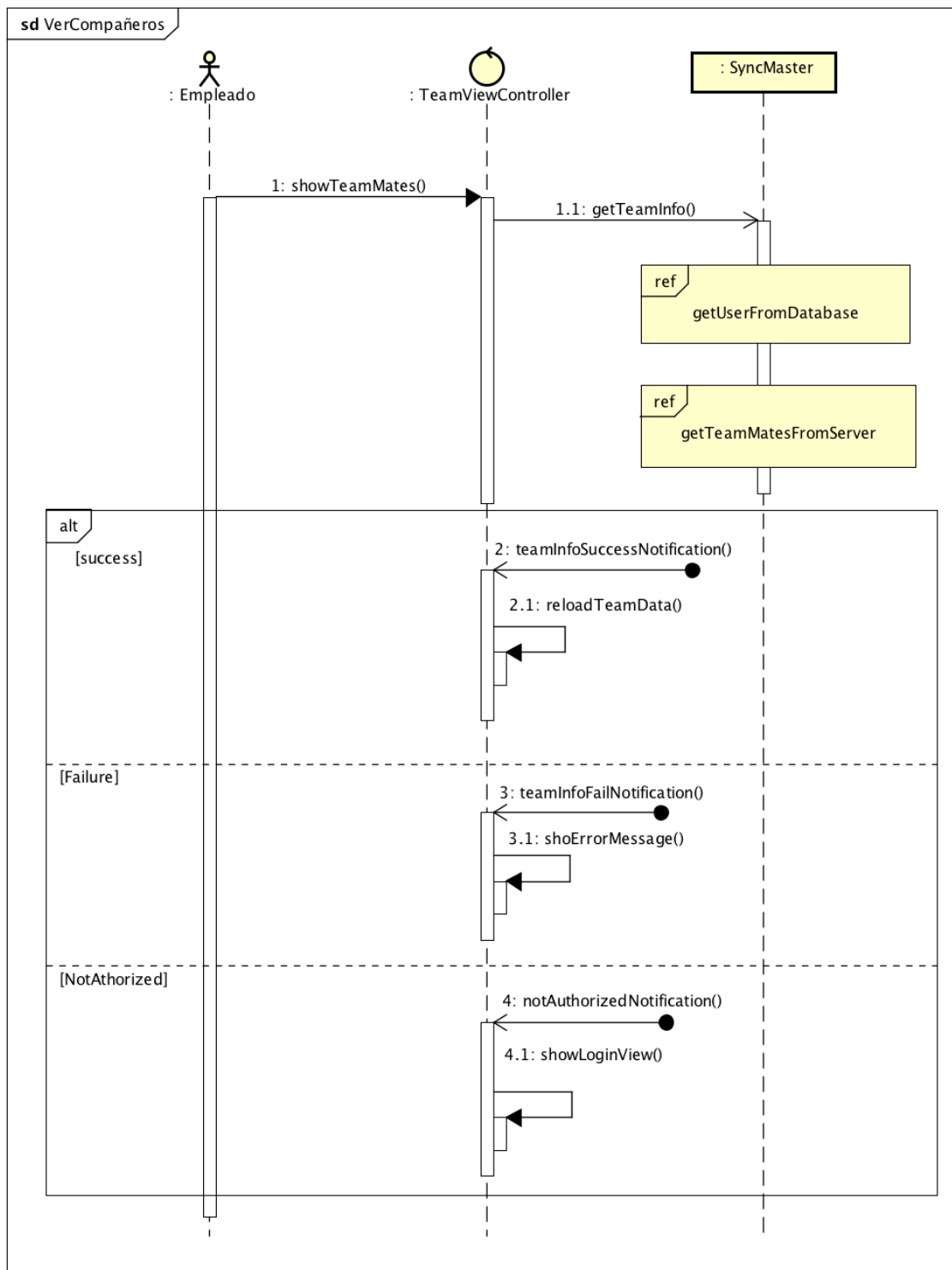


Figura 7.26: Relación en diseño del CU10

7.1.6.12. Detalle de getTeamMatesFromServer

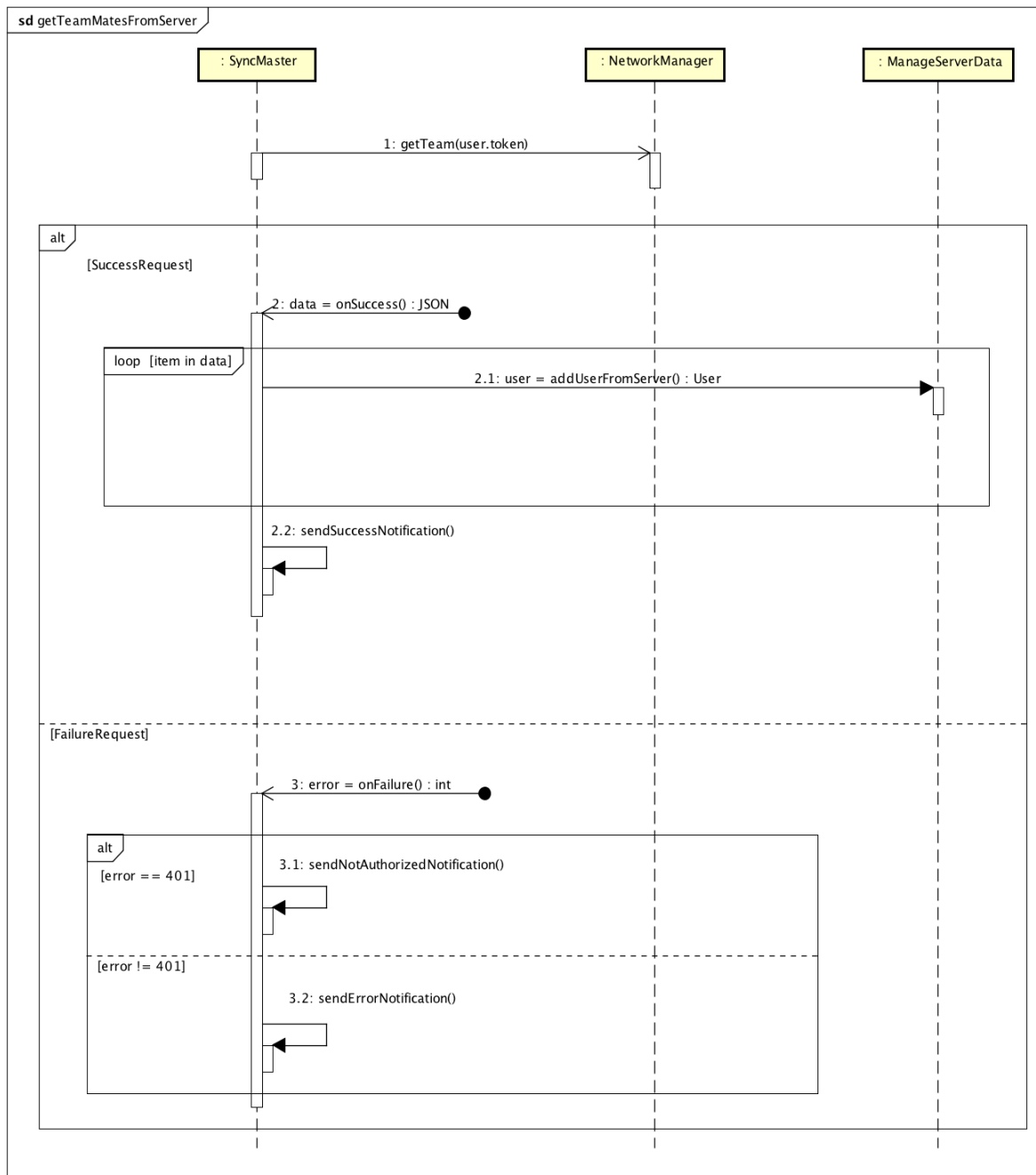


Figura 7.27: Detalla en diseño de `getTeamMatesFromServer`

7.2. Modelo de Datos

7.2.1. Esquema relacional de la base de datos

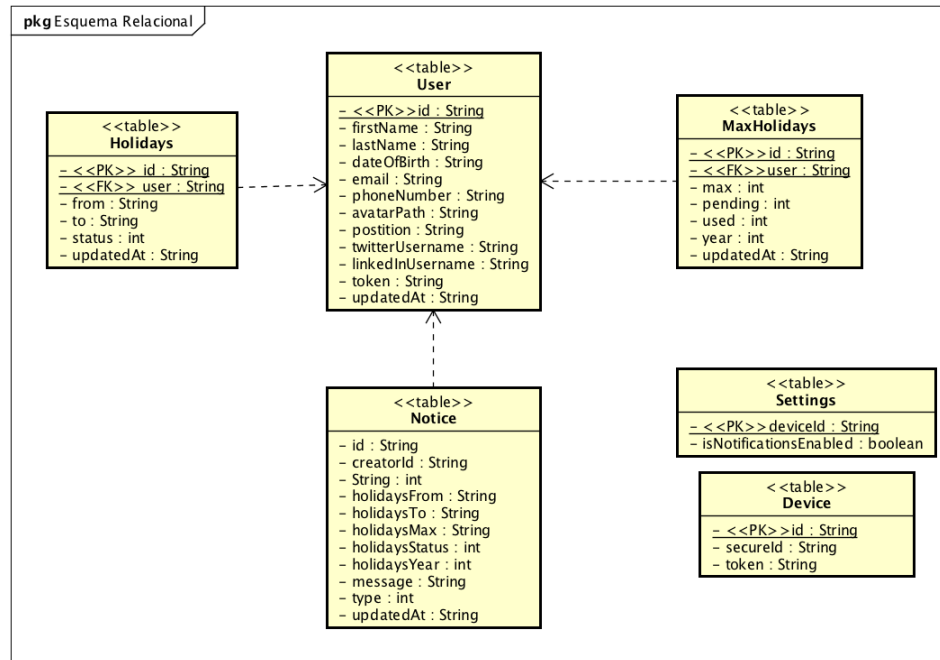


Figura 7.28: Esquema relacional de la base de datos

Nota: Las tablas *Settings* y *Device* pertenecen a la gestión de las notificaciones y al registro del dispositivo con el servidor.

7.3. Diseño de la interfaz de usuario

Al existir ya una versión de esta aplicación para el sistema operativo Android, se realizó la misma interfaz de usuario para unificar la experiencia en ambas plataformas. Todo esto teniendo en consideración siempre las guías de diseño que proporciona Apple[18]

Capítulo 8

Implementación y Pruebas

8.1. Implementación

La implementación de *SGEmployee* se ha realizado en diferentes fases, beneficiándose de la metodología utilizada y el agilismo obteniendo así, en cada nueva iteración, una nueva versión de la aplicación totalmente funcional y parcialmente probada.

En cada historia de usuario, se implementaba la funcionalidad planificada para ella, se realizaban unas pequeñas pruebas sobre ella para comprobar que la aplicación se comportara como se esperaba al añadir cambios.

Así, cuando llegó el final de las historias de usuario se tuvo una *release beta* a la que se le realizaron pruebas para finalmente tener una *release* final.

Para la implementación de la aplicación se utilizaron varias características propias del lenguaje que junto con la decisión de utilizar una variación de la arquitectura MVC, hicieron que el desarrollo fuera más sencillo y favoreciera la usabilidad. Sobre todo el uso del *NotificationCenter*[19] que provee Apple como implementación del patrón Observador, permitiendo la asincronicidad de las peticiones a la capa de red, para que la aplicación fuera en todo momento fluida y no se quedara a la espera en ningún momento.

No solo se utilizaron librerías y bibliotecas propias del sistema, también se han utilizado bibliotecas externas para hacer más fácil el desarrollo y aumentar su velocidad. Algunas de las utilizadas son:

- ***Alamofire***[20]: biblioteca que permite hacer llamadas asíncronas a la red.
- ***AlamofireImage***: extensión de la biblioteca *Alamofire* para la gestión de imágenes a través de la red.
- ***JHSpinner***: biblioteca para la carga de pantallas y espera entre medias.
- ***KDCircularProgress***: biblioteca de UI para favorecer la representación de datos.
- ***Whisper***: biblioteca de notificaciones *push in app*.

Para la base de datos del dispositivo se eligió utilizar el propio ORM que proporciona el sistema, *Core Data*[21].

Del total de las horas dedicadas al proyecto, el desarrollo del software se lleva el 65 % del tiempo dedicado a todo el proyecto.

En el CD que se entrega con esta memoria se entrega el código fuente de la aplicación.

8.2. Pruebas

8.2.1. Introducción

En un proyecto software, la realización de pruebas es fundamental para garantizar el correcto funcionamiento del proyecto en su conjunto. Garantizando así su fiabilidad y robustez.

Es de importancia realizar pruebas durante toda la duración del proyecto para detectar posibles fallos lo antes posible, evitando así, la propagación de ese error a lo largo del proyecto y del tiempo.

8.2.2. Metodología

Como ya se ha expuesto en capítulos y secciones anteriores se han hecho fases de pruebas para esta aplicación:

- **Pruebas durante una historia de usuario:** Durante el desarrollo de una funcionalidad se realizaban pruebas para comprobar que el desarrollo avanzaba bien y que la nueva funcionalidad no causaba colisiones ni errores con lo ya existente. También se probaban los casos de uso especificados para esa funcionalidad para comprobar que se comportaba como se esperaba.
- **Cross QA:** Se entrego la Beta de la aplicación a posibles usuarios de una empresa, proporcionándoles datos simulados de prueba, para que la aplicación fuera probada en un entorno real controlado para la detección de fallos. Los errores o sugerencias fueron reportados a través de la plataforma Jira para su posterior valoración y si procedía, su corrección.

8.2.3. Errores obtenidos en el Cross QA

Error 1	
Descripción	Aleatoriamente el perfil del usuario identificado cambiaba a otro compañero
Secuencia	Al navegar por las distintas secciones el usuario identificado cambiaba.
Prioridad	Urgente

Tabla 8.1: Descripción de un error encontrado durante el X-QA

Error 2	
Descripción	Cuando se trata de cambiar de estado unas vacaciones en estado <i>pendiente de aprobación</i> , la aplicación termina inesperadamente.
Secuencia	Seleccionar unas vacaciones con estado <i>pendiente de aprobación</i> , cambiar de estado esas vacaciones.
Prioridad	Urgente

Tabla 8.2: Descripción de un error encontrado durante el X-QA

Error 3	
Descripción	Cuando se pulsa dos veces en el botón <i>Cambiar estado</i> la aplicación se cierra inesperadamente.
Secuencia	Seleccionar unas vacaciones, pulsar dos veces el botón <i>Cambiar estado</i>
Prioridad	Urgente

Tabla 8.3: Descripción de un error encontrado durante el X-QA

Error 4	
Descripción	Cuando se tiene una fecha seleccionada para pedir unas vacaciones y vuelves a pulsar en el selector, la fecha cambia a la fecha actual
Secuencia	Seleccionar una fecha en la pantalla de pedir vacaciones, volver a pulsar en el selector.
Prioridad	Media

Tabla 8.4: Descripción de un error encontrado durante el X-QA

Error 5	
Descripción	Al usar Emojis en los datos al editar perfil la aplicación se cierra inesperadamente.
Secuencia	Pulsar editar perfil, introducir Emojis en alguno de los campos, aceptar, recargar la página de ver perfil.
Prioridad	Urgente

Tabla 8.5: Descripción de un error encontrado durante el X-QA

Error 6	
Descripción	El orden de las secciones del menú es distinto en inglés que en español.
Secuencia	-
Prioridad	Urgente

Tabla 8.6: Descripción de un error encontrado durante el X-QA

Error 7	
Descripción	El estado de las vacaciones se sale del espacio dado para ello cuando el idioma es Español.
Secuencia	Dispositivo en Español, pulsar la sección vacaciones.
Prioridad	Medio

Tabla 8.7: Descripción de un error encontrado durante el X-QA

Error 8	
Descripción	Sesión errónea en Twitter
Secuencia	-
Prioridad	Medio

Tabla 8.8: Descripción de un error encontrado durante el X-QA

Error 9	
Descripción	La <i>Splash Screen</i> no contiene nada.
Secuencia	Iniciar la aplicación
Prioridad	leve

Tabla 8.9: Descripción de un error encontrado durante el X-QA

A parte de estos errores, se detectaron aproximadamente otros 10 errores de interfaz tales como falta de mayúsculas, falta de acentos o errores a la hora de escribir palabras como *LinkedIn*.

También se recibieron sugerencias sobre la interfaz de usuario que, tras estudiarse, se descartaron.

Capítulo 9

Conclusiones

9.1. Objetivos alcanzados

Una vez que se ha finalizado el proyecto, resulta interesante enumerar una serie de objetivos que se han alcanzado:

- Se ha aprendido a utilizar las herramientas de desarrollo más comunes para el desarrollo de aplicaciones par el sistema operativo iOS.
- Se ha adquirido conocimientos y habilidades sobre el desarrollo de software con el lenguaje Swift. En concreto la versión Swift 3.2
- Se ha obtenido un amplio conocimiento sobre el funcionamiento del sistema operativo iOS y sus bibliotecas más comunes.
- Se han obtenido y reforzado conocimientos sobre las metodologías ágiles y más en concreto Scrum, que ha sido la utilizada en este proyecto.
- Se ha aprendido a utilizar y consumir datos de una API de tipo REST externa el proyecto.
- Se ha conseguido hacer una aplicación móvil que gestione las vacaciones laborales de los empleados de una empresa, así como más funcionalidades adicionales.
- Se ha aprendido a utilizar herramientas para la gestión de proyectos, en este caso JIRA, una de las más utilizadas en el sector.
- Se ha obtenido conocimientos sobre distintas arquitecturas para el desarrollo de aplicaciones iOS, así como la elección de cada una para cada tipo de proyecto y la adaptación de las mismas a las necesidades de cada negocio.
- Se ha obtenido conocimientos sobre distintos patrones de diseño que utiliza el sistema.
- Se ha planificado un proyecto software y se han gestionado los riesgos que han ido apareciendo en el mismo.
- Se ha reforzado el conocimiento que ya se tenía previamente en LaTeX y en la redacción de documentos.

9.2. Valoración personal

Con la realización de este TFG, he adquirido nuevos conocimientos, sobre todo en el mundo del desarrollo iOS, y he afianzado los que el grado de Ingeniería Informática, pudiendo ver sentido a muchas de las cosas aprendidas en este y pudiendo ponerlas en práctica en un entorno casi real.

El proyecto se presentaba como un proyecto largo pero a la vez interesante y, una vez acabado, puedo afirmar que así ha sido. Inicialmente la parte del desarrollo se planteaba como no muy extensa, sin embargo, a medida que me iba metiendo en la materia y desarrollando resultó que lo que inicialmente parecía sencillo se convertía en muchas horas de desarrollo.

Durante todo el desarrollo del proyecto se ha tenido que ir adaptando el alcance del mismo a la vez que se gestionaban los riesgos y los tiempos de desarrollo. Esto ultimo gracias al agilismo ha sido más llevadero y me he podido adaptar mejor a los continuos retrasos que se producían en la parte de la implementación.

A pesar de los continuos retrasos, la experiencia de trabajar con una tecnología desconocida, de trabajar con una API REST, y en definitiva de ofrecer un producto vistoso y de calidad aplicando nuevas tecnologías ha sido más que satisfactorio a nivel personal y educativo.

Para acabar ya, el proyecto ha acabado de manera satisfactoria cumpliendo todos los objetivos propuestos al inicio de este.

9.3. Lineas Futuras

Como trabajo futuro se podrían realizar los siguientes puntos:

- Añadir más idiomas a la aplicación.
- Rediseño de interfaz de usuario para las nuevas pautas que promueve Apple.
- Actualización de la aplicación a Swift 4 (Actualmente en fase beta).
- En el caso de que la empresa cuente con OwnCloud, crear una sección para ver los ficheros.
- En el caso de que la aplicación crezca en funcionalidades, sería interesante estudiar un cambio de arquitectura.
- Posible nueva sección: *Newsletter*.

Bibliografía

- [1] Apple , «*Swift*». [En línea, última vez consultado: 10/06/2017] Disponible en: <https://developer.apple.com/swift/>
- [2] Wikipedia, «*Representational state transfer*». [En línea, última vez consultado: 15/04/2017] Disponible en: https://en.wikipedia.org/wiki/Representational_state_transfer
- [3] Scrum, «*Que es Scrum*». [En línea, última vez consultado: 11/04/2017] Disponible en: <https://proyectosagiles.org/que-es-scrum/>
- [4] Wikipedia, «*Scrum*». [En línea, última vez consultado: 11/04/2017] Disponible en: [https://en.wikipedia.org/wiki/Scrum_\(software_development\)](https://en.wikipedia.org/wiki/Scrum_(software_development))
- [5] Gitlab, «*The platform for modern developers* ». [En línea, última vez consultado: 05/07/2017] Disponible en: <https://about.gitlab.com/>
- [6] Github, «*Introducing GitFlow*». [En línea, última vez consultado: 21/04/2017] Disponible en: <https://datasift.github.io/gitflow/IntroducingGitFlow.html>
- [7] JIRA, «*La herramienta de desarrollo de software líder de los equipos ágiles*». [En línea, última vez consultado: 05/07/2017] Disponible en: <https://es.atlassian.com/software/jira>
- [8] Overleaf , «*Collaborative Writing and Publishing* ». [En línea, última vez consultado: 05/07/2017] Disponible en: <https://www.overleaf.com/>
- [9] Ian F. Sommerville, *Ingeniería del Software, Novena edición, Capítulo 2.1.1, El modelo en cascada.*
- [10] Ian F. Sommerville, *Ingeniería del Software, Novena edición, Capítulo 2.4, El Proceso Unificado Racional.*
- [11] Proyectos Agiles, «*Cliente (Product Owner)*». [En línea, última vez consultado: 11/04/2017] Disponible en: <https://proyectosagiles.org/cliente-product-owner/>
- [12] Scrum Manager, «*Gestion de proyectos Scrum Manager*». [En línea, última vez consultado: 11/04/2017] Disponible en: http://www.scrummanager.net/files/sm_proyecto.pdf
- [13] Laura Daly, «*Break it down: decomposing user stories in JIRA*». [En línea, última vez consultado: 16/03/2017] Disponible en: <https://www.atlassian.com/blog/jira-software/break-decomposing-user-stories-jira>

- [14] Bohdan Orlov, «*iOS Architecture Patterns*». [En línea, última vez consultado: 10/06/2017] Disponible en: <https://medium.com/ios-os-x-development/ios-architecture-patterns-ecba4c38de52>
- [15] Vivian Tran & Yixin Zhu , «*Engineering the Architecture Behind Uber New Rider App*». [En línea, última vez consultado: 07/05/2017] Disponible en: <https://eng.uber.com/new-rider-app/>
- [16] Rui Peres, «*Model-View-Controller (MVC) in iOS: A Modern Approach*». [En línea, última vez consultado: 10/06/2017] Disponible en: <https://www.raywenderlich.com/132662/mvc-in-ios-a-modern-approach>
- [17] Marcus Zarra, «*MVC-N: Isolating network calls from View Controllers*». [En línea, última vez consultado: 10/06/2017] Disponible en: <https://news.realm.io/news/slug-marcus-zarra-exploring-mvcn-swift/>
- [18] Apple, «*Human Interface Guidelines*». [En línea, última vez consultado: 06/05/2017] Disponible en: <https://developer.apple.com/design/>
- [19] Apple, «*NSNotificationCenter*». [En línea, última vez consultado: 05/07/2017] Disponible en: <https://developer.apple.com/documentation/foundation/nsnotificationcenter>
- [20] Aaron Douglas, «*Alamofire Tutorial: Getting Started*». [En línea, última vez consultado: 05/06/2017] Disponible en: <https://www.raywenderlich.com/147086/alamofire-tutorial-getting-started-2>
- [21] Apple , «*What Is Core Data?*». [En línea, última vez consultado: 23/05/2017] Disponible en: <https://developer.apple.com/library/content/documentation/Cocoa/Conceptual/CoreData/index.html>

Anexo A

Glosario de Términos

- **iOS:** es un sistema operativo móvil de la multinacional Apple Inc.
- **IDE:** siglas de *Integrated Development Environment* o entorno de desarrollo integrado. Es una aplicación informática que proporciona servicios integrales para facilitarle al desarrollador o programador el desarrollo de software.
- **Xcode:** es un entorno de desarrollo integrado(IDE) para macOS que contiene un conjunto de herramientas creadas por Apple destinadas al desarrollo de software para macOS, iOS, watchOS y tvOS.
- **API:** siglas de *interfaz de programación de aplicaciones*. Conjunto de subrutinas, funciones y procedimientos que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.
- **REST:** describe cualquier interfaz entre sistemas que utilice directamente HTTP para obtener datos o indicar la ejecución de operaciones sobre los datos, en cualquier formato sin las abstracciones adicionales de los protocolos basados en patrones de intercambio de mensajes
- **LaTeX:** sistema de composición de textos, orientado a la creación de documentos escritos que presenten una alta calidad tipográfica.
- **Pila del producto o Product Backlog:** lista de los requisitos desde el punto de vista del cliente. Está formada por las funcionalidades o historias de usuario que desea obtener el cliente, ordenadas por la prioridad que el mismo le otorga a cada una.
- **SGEmployee:** Nombre de la aplicación para este proyecto.
- **Scrum:** modelo de desarrollo ágil basado en una colaboración estrecha con el cliente y una predisposición y respuesta ante los cambios.
- **Sprint:** período de un proyecto Scrum en el cual se lleva a cabo el trabajo en sí. Es recomendado que la duración de los sprints sea constante y definida por el equipo en base a su propia experiencia (2 o 3 semanas).
- **TFG:** siglas de *trabajo fin de grado*. Proyecto o trabajo exigido como condición para obtener finalmente una titulación universitaria.

Anexo B

Plan de Seguimiento

B.1. Introducción

Para que un proyecto software tenga éxito y cumpla con los requisitos tanto de tiempo como de alcance y calidad es necesario hacer un correcto seguimiento de todo el trabajo llevado a cabo durante la duración del mismo.

Gracias al uso de la herramienta de gestión de proyectos ágiles JIRA ahora podemos extraer de ella detalles importantes para ver los detalles de seguimiento del proyecto.

A continuación, y antes de entrar en detalles sobre el desarrollo del proyecto vamos a mostrar un pequeño resumen de la metodología de trabajo que seguimos con JIRA para ilustrar el flujo de trabajo de la metodología ágil Scrum en conjunto con la propia herramienta.

Para apoyarnos en la explicación de una manera visual disponemos de la Figura B.1 en la que podemos ver gráficamente todo el flujo de trabajo de Scrum.

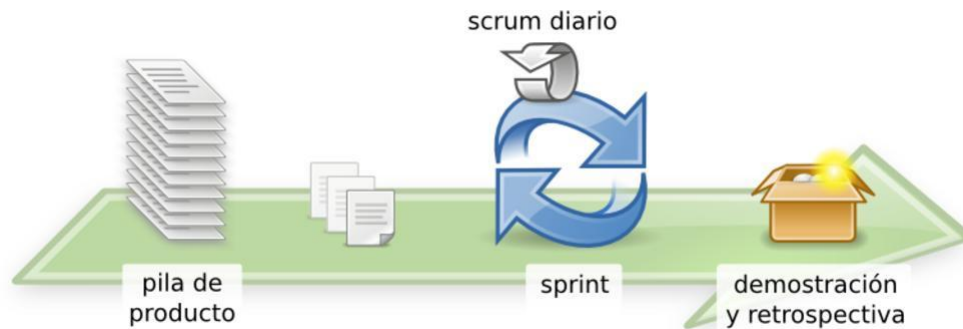


Figura B.1: Explicación gráfica del flujo de Scrum

Como podemos apreciar en la figura B.1 el proceso normal de desarrollo en Scrum tiene las siguientes fases:

1. Se parte de la pila de producto o *Product Backlog*, en la que se registra una lista de requisitos que el cliente ha solicitado o requerido. Estos requisitos se traducen en funcionalidad e historias de usuario o *User Stories* ordenadas por prioridad. Esta prioridad la da el cliente siempre guiado por el equipo de desarrollo.
2. De esta pila de producto se seleccionan las *User Stories* que la componen aquellas que más prioridad tienen. Las que se seleccionen en este punto serán las que compongan la nueva funcionalidad para empezar el *Sprint*

3. Una vez que se tiene seleccionadas las *User Stories* que se van a desarrollar en el siguiente *Sprint* lo que haremos será descomponer cada una de ellas en tareas pequeñas que el equipo de desarrollo deberán realizar. A esta lista de tareas pequeñas por cada historia de usuario se le denomina *Sprint Backlog*
4. Una vez que se tiene decidida la funcionalidad a realizar y se ha subdividido en pequeñas tareas que poder asignar a cada miembro del equipo comienza el *Sprint*. Normalmente tiene una duración de unas 2 o 3 semanas en las cuales se trabaja en las tareas que se definieron.
5. Durante cada día de cada *Sprint*, al empezar la jornada, se realiza una reunión diaria o *Daily* donde cada miembro del equipo expone el trabajo realizado y a realizar y se comenta de forma breve el avance del proyecto o si hay algún impedimento o bloqueo en alguna de las tareas.
6. Para finalizar, cada iteración o *Sprint* acaba con una nueva entrega que se suma a las anteriores de una parte operativa del producto o un incremento del mismo. Se realiza una demostración con el cliente para mostrar el avance y detectar posibles cambios o cosas que han de cambiarse
7. Además el equipo hace una reunión llamada Retrospectiva para ver que cosas han ido bien y que cosas han ido mal en el *Sprint* que acaban de acabar para ponerlas en común e intentar corregirlas para que no se vuelvan a repetir.

B.1.1. Propósito

Este documento de seguimiento incluye de una forma detallada los diferentes *Sprints* de los que se compuso el proyecto. Para este proyecto se han estimado 14 *Sprints* con una duración aproximada de 1 semana cada uno. Dependiendo del avance del proyecto y de si se producen retrasos se incluirán entre los días de trabajo el sábado y el domingo.

Para cada *Sprint* se expondrán las historias de usuario que se van a desarrollar y para cada historia de usuario se realizarán estimaciones de esfuerzo basado en los puntos de historias de usuario. Estos puntos representan de una forma numérica la estimación tanto en tamaño como en complejidad de cada una de las historias de usuario. Como es lógico, a mayor dificultad y tamaño, más puntos.

B.1.2. Alcance

El seguimiento del proyecto se realiza durante todas las etapas del mismo. Durante todo el desarrollo del proyecto, apoyándonos en la plataforma JIRA, se anotan todas las tareas que se han de realizar y su progreso, anotando si fuera necesario comentarios acerca de las tareas y su desarrollo.

Mediante este documento lo que se pretende es dar una visión clara y detallada de todo lo que ha sucedido en el proyecto.

B.2. Sprint 0: 20 de febrero - 1 de marzo

Este primer *Sprint* no forma parte de la implementación de la funcionalidad de la aplicación sino que se centra en preparar la documentación y los entornos de desarrollo así como asegurarse que el proyecto es viable. Sirve como una primera toma de contacto con las herramientas tanto de desarrollo de software como de documentación.

B.2.1. *User Stories* a desarrollar

ID	Descripción
US00	Como Equipo quiero tener disponibles tanto los entornos de desarrollo de software como de documentación listos para empezar el proyecto.
US01	Como <i>Product Owner</i> quiero disponer de la documentación inicial del proyecto para tener una visión global del mismo.

Tabla B.1: *User Stories* para el Sprint 0

B.2.2. US00 - Como Equipo quiero tener disponibles tanto los entornos de desarrollo de software como de documentación listos para empezar el proyecto.

B.2.2.1. Descripción

Es primordial tener los equipos correctamente configurados y listos para empezar el trabajo, en esta historia de usuario se instalará el software necesario para el desarrollo como los IDE, las maquinas virtuales o los SDK. También se prepararán los proyectos en ShareLatex para poder escribir la memoria. Se decide también descargar una versión nativa de Latex por si algún impedimento apareciera con la plataforma online.

B.2.2.2. Puntos

Esta *User Story* se decide que tenga 2 puntos ya que instalar y configurar los entornos no tiene dificultad. Lo que si conlleva es tiempo ya que la conexión ADSL utilizada no es muy rápida.

B.2.2.3. Tareas

- **T01** - Descarga e instalación del IDE XCode
- **T02** - Descarga e instalación del SDK de iOS 9 e iOS 10
- **T03** - Descarga e instalación de MacTex
- **T04** - Creación de un nuevo proyecto en la plataforma ShareLatex

B.2.2.4. Comentarios

Nada que destacar en esta historia de usuario, todo fue instalado correctamente y sin ninguna dificultad.

B.2.3. US01 - Como *Product Owner* quiero disponer de la documentación inicial del proyecto para tener una visión global del mismo.

B.2.3.1. Descripción

Al ser una gestión de proyecto ágil, la documentación se irá desarrollando a lo largo de todo el proyecto. Sin embargo, en esta historia de usuario, se desarrollará la documentación inicial, tal como empezar el plan de desarrollo del software, el plan de gestión de riesgos o la primera versión de la especificación de requisitos.

B.2.3.2. Puntos

Esta *User Story* se decide que tenga 5 puntos ya que hay que comenzar a desarrollar los diferentes planes de desarrollo y gestión de riesgo, así como la primera recogida de requisitos.

B.2.3.3. Tareas

- **T01** - Iniciar el plan de desarrollo del software.
- **T02** - Iniciar el plan de gestión de riesgos.
- **T03** - Primera toma de requisitos funcionales, no funcionales y de información.

B.2.3.4. Comentarios

Dado que se usa una API y que había que adaptarse a una solución de la parte de servidor ya especificada hubo que adaptar los requisitos de información a la información que el servidor proporcionaba.

B.2.4. Riesgos

En este *Sprint* no se ha disparado ningún riesgo.

B.3. Sprint 1: 2 de marzo - 8 de marzo

B.3.1. *User Stories* a desarrollar

ID	Descripción
US02	Como <i>Product Owner</i> quiero tener un modelo de dominio de la aplicación.
US03	Como Usuario quiero poder identificarme en la aplicación con mis credenciales corporativas.

Tabla B.2: *User Stories* para el Sprint 1

B.3.2. US02 - Como *Product Owner* quiero tener un modelo de dominio de la aplicación.

B.3.2.1. Descripción

Para continuar con la elaboración de la aplicación y con el desarrollo de la documentación una primera versión del modelo de dominio es necesario dar una idea global y para así más tarde completarlo a medida que la creación de funcionalidad avanza.

B.3.2.2. Puntos

Esta *User Story* se decide que tenga 3 puntos ya que como nos encontramos en una fase temprana del desarrollo el modelo de dominio de la aplicación no será muy extenso ni complicado.

B.3.2.3. Tareas

- **T01** - Instalación del software Astah Professional para la creación de los diagramas.
- **T02** - Identificación de las clases que forman el dominio de la aplicación.
- **T03** - Desarrollo de la primera versión del modelo de dominio.

B.3.2.4. Comentarios

Como es un desarrollo ágil el modelo de dominio de la aplicación no estará completo, solo reflejará el dominio de la aplicación para la funcionalidad desarrollada durante este *Sprint*.

B.3.3. US03 - Como Usuario quiero poder identificarme en la aplicación con mis credenciales corporativas.

B.3.3.1. Descripción

Para poder usar la aplicación y todas sus funcionalidades es necesario entrar en ella con las credenciales corporativas con lo que sin esta funcionalidad la aplicación pierde sentido.

B.3.3.2. Puntos

Esta *User Story* se decide que tenga 7 puntos ya que se cuenta con poca experiencia con el desarrollo para iOS, con la utilización de llamadas a la red a través del sistema y con la comunicación con el servidor.

B.3.3.3. Tareas

- **T01** - Realizar la comunicación con el servidor.
- **T02** - Gestionar el proceso de identificación mediante la creación de una pantalla de identificación.
- **T03** - Gestionar el proceso de olvido de contraseña.
- **T04** - Gestionar las posibles respuestas de error del servidor.
- **T05** - Gestionar las llamadas a la red cuando el dispositivo carece de red.

B.3.3.4. Comentarios

Se eligió la librería más famosa de comunicación con el servidor en Swift llamada *Alamofire* ya que era la que más documentación y ejemplos tiene, siendo la más recomendada dentro de la comunidad.

B.3.4. Riesgos

En este *Sprint* no se ha disparado ningún riesgo.

B.4. Sprint 2: 9 de marzo - 20 de marzo

B.4.1. *User Stories* a desarrollar

ID	Descripción
US04	Como Usuario quiero poder ver una lista de mis compañeros de trabajo, poder ver su foto de perfil, ver su <i>Twitter</i> o <i>LinkedIn</i> y llamar a su teléfono.

Tabla B.3: *User Stories* para el Sprint 2

B.4.2. US04 - Como Usuario quiero poder ver una lista de mis compañeros de trabajo, poder ver su foto de perfil, ver su *Twitter* o *LinkedIn* y llamar a su teléfono.

B.4.2.1. Descripción

Mediante esta funcionalidad será posible ver una lista de los compañeros de trabajo, poder ver su foto de perfil, sus redes sociales o incluso llamar a su teléfono. Todo esto siempre y cuando cada compañero de trabajo lo tenga introducido en su perfil. En el caso de que un compañero de trabajo no lo haya introducido en su perfil, se ofrecerá contenido por defecto, siendo este el nombre y email del compañero, una foto por defecto, y los iconos deshabilitados de *Twitter*, *LinkedIn* y teléfono.

B.4.2.2. Puntos

Esta *User Story* se decide que tenga 10 puntos ya que involucra un conocimiento exhaustivo del SKD de iOS, así como nuevas comunicaciones con el servidor.

B.4.2.3. Tareas

- **T01** - Actualizar el modelo de dominio.
- **T02** - Realizar la comunicación con el servidor.
- **T03** - Cachear la respuesta del servidor.
- **T04** - Mostrar la imagen de perfil de forma asíncrona.
- **T05** - Cachear las imágenes de perfil.
- **T06** - Gestionar la llamada o vista de redes sociales.
- **T07** - Gestionar las posibles respuestas de error del servidor.
- **T08** - Gestionar las llamadas a la red cuando el dispositivo carece de red.

B.4.2.4. Comentarios

Para esta funcionalidad se decidió utilizar una librería complementaria a *Alamofire* llamada *AlamofireImage* con la que la descarga de las imágenes de perfil se hace de manera sencilla y simple. Hubo problemas a la hora de gestionar los datos del servidor y también para mantener una tasa de refresco aceptable por lo que el desarrollo de esta funcionalidad se demoró más de lo debido.

B.4.3. Riesgos

Para el desarrollo de esta nueva funcionalidad se dispararon los riesgos R 4.4 y R 4.5 a causa de:

- Poco conocimiento de la APIs del sistema en lo requerido a esta funcionalidad.
- Poco conocimiento de las librerías a utilizar para el correcto funcionamiento de la funcionalidad.
- Tiempo invertido en aprendizaje.

Si bien ya estaba contemplado en la gestión de riesgos y también se le había asignado una puntuación alta en los puntos de historia de usuario, fue imposible tener la funcionalidad requerida en el tiempo planeado. Para solucionar esto se ha aplicado la medida correctora de aumentar el tiempo del *Sprint* para dar tiempo a finalizar la funcionalidad. La otra opción hubiera sido introducirla en el *Sprint* siguiente pero eso provocaría que el siguiente tampoco terminara a tiempo causando mas retrasos.

B.5. Sprint 3: 21 de marzo - 31 de marzo

B.5.1. *User Stories* a desarrollar

ID	Descripción
US05	Como Usuario quiero poder ver y editar mi perfil corporativo desde la aplicación

Tabla B.4: *User Stories* para el Sprint 3

B.5.2. US05 - Como Usuario quiero poder ver y editar mi perfil corporativo desde la aplicación

B.5.2.1. Descripción

A través de esta funcionalidad, el usuario de la aplicación podrá introducir una foto de perfil, su posición en la empresa, su número de teléfono, su fecha de cumpleaños y sus cuentas de *LinkedIn* y *Twitter*.

Esta información será remitida al servidor y sera utilizada por ejemplo para completar los datos en la lista de empleados, funcionalidad ya desarrollada.

B.5.2.2. Puntos

Esta *User Story* se decide que tenga 10 puntos ya que involucra un conocimiento exhaustivo del SKD de iOS, así como nuevas comunicaciones con el servidor y el acceso a recursos como la galería de fotos del sistema.

B.5.2.3. Tareas

- **T01** - Realizar la comunicación con el servidor.
- **T02** - Crear una pantalla para ver el perfil.
- **T03** - Mostrar la imagen de perfil de forma asíncrona.
- **T04** - Cachear las imágenes de perfil.
- **T05** - Crear una pantalla para modificar el perfil.
- **T06** - Gestionar las posibles respuestas de error del servidor.
- **T07** - Gestionar las llamadas a la red cuando el dispositivo carece de red.
- **T08** - Actualizar el modelo de dominio.

B.5.2.4. Comentarios

Esta funcionalidad involucra el manejo y cacheo de imágenes, el manejo del SDK de iOS y la creación de varias pantallas lo que la hace una historia de usuario bastante larga. Desde un punto de vista de la gestión del proyecto, esta historia de usuario podría haberse dividido en dos: una para ver el perfil y otra para modificarlo.

B.5.3. Riesgos

Para el desarrollo de esta nueva funcionalidad se dispararon los riesgos R 4.4, R 4.5 y R 4.8, al igual que en el *Sprint* anterior.

Los motivos del disparo de estos riesgos son básicamente los mismos. El *Sprint* ya comienza con retraso por culpa del anterior y no se estima bien el esfuerzo que se necesitará para poder llevar a cabo las tareas. Es evidente que la medida de 10 puntos se queda corta a la hora de estimar esta historia de usuario. Para solucionar esto se ha aplicado la misma medida correctora: aumentar el tiempo del *Sprint* para dar tiempo a finalizar la funcionalidad.

B.6. Sprint 4: 1 de abril - 6 de abril

B.6.1. *User Stories* a desarrollar

ID	Descripción
US06	Como <i>Product Owner</i> quiero tener una primera versión del modelo de la arquitectura del sistema y de su diseño, así como de los patrones que son utilizados.

Tabla B.5: *User Stories* para el Sprint 4

B.6.2. US06 - Como *Product Owner* quiero tener una primera versión del modelo de la arquitectura del sistema y de su diseño, así como de los patrones que son utilizados.

B.6.2.1. Descripción

Como ya tenemos suficiente funcionalidad y establecida ya la arquitectura que llevará la aplicación, podemos comenzar utilizando el modelo de dominio y todo el análisis para comenzar a hacer los diagramas de diseño de la aplicación, que servirán tanto para comprobar que se esta siguiendo la arquitectura definida, como para documentar la aplicación desde el punto de vista del diseño de la misma.

B.6.2.2. Puntos

Esta *User Story* se decide que tenga 10 puntos ya que implica la creación del documento de diseño y arquitectura de la aplicación y realizar el diseño hasta la fecha del sistema.

B.6.2.3. Tareas

- **T01** - Definir la arquitectura que se usa en la aplicación.
- **T02** - Definir los patrones de diseño que se usan.
- **T03** - Modelo general de la arquitectura.
- **T04** - Descomposición modular.
- **T05** - Relación entre módulos.

B.6.2.4. Comentarios

Si bien no es el diagrama de arquitectura completo, contiene la funcionalidad hasta el momento desarrollada, comprobando así que el software desarrollado se corresponde con la arquitectura que se utiliza y su validez.

B.6.3. Riesgos

En este *Sprint* no se dispara ningún riesgo.

B.7. Sprint 5: 7 de abril - 19 de abril

B.7.1. *User Stories* a desarrollar

ID	Descripción
US09	Como Usuario quiero poder recibir notificaciones cuando alguna de mis vacaciones cambie de estado.
US10	Como Usuario quiero poder recibir notificaciones que el administrador decida mandarme.
US11	Como Usuario quiero poder ver una lista con las últimas notificaciones que he recibido.

Tabla B.6: *User Stories* para el Sprint 5

B.7.2. US09 - Como Usuario quiero poder recibir notificaciones cuando alguna de mis vacaciones cambie de estado

B.7.2.1. Descripción

El usuario podrá recibir notificaciones cuando le sean asignadas vacaciones para el año actual, el año anterior y el siguiente. También recibirá una notificación cuando alguna de las vacaciones que ha pedido cambie de estado, es decir, sea aprobada o rechazada.

B.7.2.2. Puntos

Esta *User Story* se decide que tenga 5 puntos ya que ya se tiene experiencia con el sistema y la gestión de las notificaciones se apoya en la propia Apple y en librerías externas.

B.7.2.3. Tareas

- **T01** - Actualizar el diseño de la aplicación con la nueva funcionalidad.
- **T02** - Manejar el intercambio de datos con el servidor.
- **T03** - Manejar si el usuario acepta o no recibir notificaciones.
- **T04** - Manejar si el usuario tiene las notificaciones habilitadas.
- **T05** - Identificar el tipo de notificación.
- **T06** - Mostrar al usuario los cambios correspondientes dependiendo del tipo de la notificación.

B.7.2.4. Comentarios

Para esta historia de usuario se ha utilizado la librería *Whisper* ya que las notificaciones *in app* no son permitidas en iOS 9.

B.7.3. US10 - Como Usuario quiero poder recibir notificaciones que el administrador decida mandarme

B.7.3.1. Descripción

El usuario podrá recibir notificaciones que el/los administrador/es de la plataforma decida mandar.

B.7.3.2. Puntos

Esta *User Story* se decide que tenga 3 puntos ya que ya se tiene experiencia con el sistema y la gestión de las notificaciones se apoya en la propia Apple y en librerías externas.

B.7.3.3. Tareas

- **T01** - Actualizar el diseño de la aplicación con la nueva funcionalidad.
- **T02** - Manejar el intercambio de datos con el servidor.
- **T03** - Manejar si el usuario acepta o no recibir notificaciones.
- **T04** - Manejar si el usuario tiene las notificaciones habilitadas
- **T05** - Mostrar al usuario la pantalla de notificaciones y avisos si pulsa en la notificación.

B.7.3.4. Comentarios

Para esta historia de usuario se ha utilizado la librería *Whisper* ya que las notificaciones *in app* no son permitidas en iOS 9.

B.7.4. US11 - Como Usuario quiero poder ver una lista con las últimas notificaciones que he recibido

B.7.4.1. Descripción

El usuario podrá ver una lista con las última notificaciones tanto de cambios en las vacaciones como si han sido mandadas por el administrador.

B.7.4.2. Puntos

Esta *User Story* se decide que tenga 3 puntos ya que el servidor automáticamente devuelve una lista con los últimos avisos y solo hay que mostrarlos en pantalla.

B.7.4.3. Tareas

- **T01** - Actualizar el diseño de la aplicación con la nueva funcionalidad.
- **T02** - Manejar el intercambio de datos con el servidor.
- **T03** - Manejar la cache de los datos obtenidos por el servidor.
- **T04** - Manejar los mensajes de error del servidor.
- **T05** - Manejar el estado de la aplicación cuando el dispositivo no tiene red.

B.7.5. Riesgos

Para el desarrollo de esta nueva funcionalidad se dispararon los riesgos R 4.4, R 4.5 y R 4.7. Se necesitó crear certificados de Apple y lidiar con ellos ya que los desarrolladores del servidor no habían podido probar las notificaciones para iOS. Se realizaron pequeñas modificaciones para poder hacer funcional las notificaciones iOS.

Esta tarea no estaba prevista pero como se contaba con la gestión de riesgos y se tenía la consciencia de que este riesgo podría dispararse los efectos no fueron otros que el retraso a la hora de acabar la iteración.

Parte del retraso también se debe a la falta de experiencia con las notificaciones en iOS y el diferente manejo que hacen de ellas las dos versiones de software que soporta la aplicación: iOS 9.* y iOS 10.*

Debido a ese distinto manejo, hubo que integrar una librería de terceros, aprender a usarla y comprobar que funcionaba como era debido para cumplir así con la funcionalidad requerida.

Para solucionar esto se ha aplicado la misma medida correctora: aumentar el tiempo del *Sprint* para dar tiempo a finalizar la funcionalidad y sobre todo a corregir los errores de software para que estos no afectaran más adelante a las nuevas funcionalidades.

B.8. Sprint 6: 20 de abril - 5 de mayo

B.8.1. *User Stories* a desarrollar

ID	Descripción
US08	Como Usuario quiero poder pedir vacaciones para poder disfrutar de mis derechos laborales.

Tabla B.7: *User Stories* para el Sprint 6

B.8.2. US08 - Como Usuario quiero poder pedir vacaciones para poder disfrutar de mis derechos laborales.

B.8.2.1. Descripción

La funcionalidad sin la cual la aplicación no tendría sentido, el usuario podrá pedir vacaciones en un año siempre y cuando le queden días de vacaciones.

B.8.2.2. Puntos

Esta *User Story* se decide que tenga 12 puntos ya que es la funcionalidad estrella y a priori la más complicada de la aplicación.

B.8.2.3. Tareas

- **T01** - Actualizar el modelo de dominio.
- **T02** - Actualizar el diseño de la aplicación con la nueva funcionalidad.
- **T03** - Manejar el intercambio de datos con el servidor.
- **T04** - Crear una pantalla para pedir vacaciones introduciendo esta nueva información
- **T05** - Cachear los datos obtenidos del servidor.
- **T06** - Modificar la pantalla de ver vacaciones para introducir los datos de las nuevas vacaciones pedidas.
- **T07** - Manejar los cambios de estado de las vacaciones realizando las acciones correspondientes.
- **T08** - Manejar los posibles errores que puede dar el servidor.
- **T09** - Manejar el estado de la aplicación cuando no se tiene red en el dispositivo.

B.8.2.4. Comentarios

Para esta historia de usuario se tuvieron que ampliar los conocimientos sobre el manejo de tablas en iOS y sus delegados por lo que implicó tiempo de aprendizaje.

B.8.3. Riesgos

Para el desarrollo de esta nueva funcionalidad se dispararon los riesgos R 4.4, R 4.5, R 4.8 y R 4.9.

Los motivos fueron, a parte de la poca experiencia con el uso de funcionalidades específicas del API de tablas para iOS, el tratamiento con las fechas en iOS, etc. En general como se viene dando a lo largo del proyecto, la poca experiencia con la plataforma hace que se requiera una estimación superior a la que se le ha dado, siendo esta muy baja en comparación con el trabajo real llevado a cabo.

La aparición de errores en el software difíciles de solucionar consumieron mucho tiempo en esta iteración.

Para solucionar esto se ha aplicado la misma medida correctora: aumentar el tiempo del *Sprint* para dar tiempo a finalizar la funcionalidad y sobre todo a corregir los errores de software para que estos no afectaran más adelante a las nuevas funcionalidades.

B.9. Sprint 7: 08 de mayo - 15 de mayo

B.9.1. *User Stories* a desarrollar

ID	Descripción
US07	Como Usuario quiero poder ver los días de vacaciones de los que dispongo y cuantos me quedan por usar

Tabla B.8: *User Stories* para el Sprint 7

B.9.2. US07 - Como Usuario quiero poder ver los días de vacaciones de los que dispongo y cuantos me quedan por usar

B.9.2.1. Descripción

Una de las funcionalidades estrella de la aplicación, mediante esta funcionalidad el usuario puede ver cuantos días de vacaciones le han sido asignados para el año actual, el anterior y el siguiente y cuantos días de esos le quedan por disfrutar.

B.9.2.2. Puntos

Esta *User Story* se decide que tenga 10 puntos ya que implica intercambio de información con el servidor, una gestión avanzada de la interfaz de usuario y un amplio conocimiento del SDK de iOS

B.9.2.3. Tareas

- **T01** - Actualizar el modelo de dominio.
- **T02** - Actualizar el diseño de la aplicación con la nueva funcionalidad.
- **T03** - Manejar el intercambio de datos con el servidor.
- **T04** - Crear una pantalla para ver las vacaciones.
- **T05** - Cachear los datos obtenidos del servidor.
- **T06** - Manejar los posibles errores que puede dar el servidor.
- **T07** - Manejar el estado de la aplicación cuando no se tiene red en el dispositivo.

B.9.2.4. Comentarios

Se decidió usar la biblioteca *KDCircularProgress* para la interfaz de usuario ya que es una de las más extendidas.

Añadir que, como se viene realizando cada vez que se añade funcionalidad, se revisa el modelo de dominio y se actualiza con los posibles cambios que se hayan dado a la hora de añadir cosas nuevas al proyecto ya hecho.

La planificación de esta historia de usuario y de esta iteración si que se ajustó al tiempo definido por lo que la estimación fue buena.

B.9.3. Riesgos

En este *Sprint* no se dispara ningún riesgo.

B.10. Sprint 8: 16 de mayo - 24 de mayo

B.10.1. *User Stories* a desarrollar

ID	Descripción
US12	Como Usuario quiero poder ver información sobre la empresa, como por ejemplo su dirección, CIF, etc.

Tabla B.9: *User Stories* para el Sprint 8

B.10.2. US12 - Como Usuario quiero poder ver información sobre la empresa, como por ejemplo su dirección, CIF, etc.

B.10.2.1. Descripción

El usuario podrá ver la información mas importante sobre la empresa. Esa información será la razón social, el CIF, la dirección, etc.

B.10.2.2. Puntos

Esta *User Story* se decide que tenga 8 puntos ya que ya se tiene experiencia con el sistema y el contenido es estático. Se tiene en cuenta a la hora del desarrollo la posible adaptación para diferentes empresas por lo que se debe dejar el sistema preparado para personalizarlo con cualquier información y que no suponga un coste de desarrollo grande.

B.10.2.3. Tareas

- **T01** - Actualizar el diseño de la aplicación con la nueva funcionalidad.
- **T02** - Crear una pantalla para mostrar los datos de la empresa.
- **T03** - Introducir la información de la compañía.
- **T04** - Manejar el estado de la aplicación dependiendo de si el dispositivo tiene red o no.

B.10.2.4. Comentarios

Para esta historia de usuario se ha utilizado el SDK MapKit que proporciona la propia Apple para mostrar la ubicación de la empresa en un mapa. Se dejo preparado el contenido para que el coste de personalizar este apartado sea casi nulo.

B.10.3. Riesgos

En este *Sprint* no se disparó ningún riesgo.

B.11. Sprint 9: 25 de mayo - 31 de mayo

B.11.1. *User Stories* a desarrollar

ID	Descripción
US13	Como Usuario quiero tener una versión de la aplicación instalada en mi dispositivo

Tabla B.10: *User Stories* para el Sprint 9

B.11.2. US13 - Como Usuario quiero tener una versión de la aplicación instalada en mi dispositivo

B.11.2.1. Descripción

En esta historia de usuario se prepara la aplicación para sacar una primera versión. Es aquí donde se realizan las pruebas en un entorno real. El método elegido para probarlo se conoce como *Cross QA* o *Testing cruzado* y consiste en instalar la aplicación en una serie de dispositivos de prueba y ofrecérsela a una selección de potenciales usuarios para que la prueben. Durante estas pruebas, utilizando herramientas de monitorización, se detectan posibles errores en la aplicación ya sean reportados por las herramientas o por los propios usuarios de prueba.

B.11.2.2. Puntos

Esta *User Story* se decide que tenga 10 puntos ya la aplicación se encuentra en su fase final. Lo único que faltaría es corregir los errores que se produzcan durante las pruebas en esta fase.

B.11.2.3. Tareas

- **T01** - Conectar la aplicación con un entorno de prueba.
- **T02** - Generar el ejecutable *ad hoc* de la aplicación y distribuirlo.
- **T03** - Recoger el *feedback* de las personas que la han probado y de las herramientas de monitorización.
- **T04** - Corregir los posibles errores o sugerencias que se detecten.

B.11.2.4. Comentarios

Ya que se contaba con la colaboración de muchos usuarios potenciales de la aplicación se decidió que la mejor manera de probar tanto la funcionalidad como la sencillez de uso era instalando la aplicación en varios dispositivos y dándoselas a esos usuarios potenciales para que detectaran fallos o sugerencias y así después corregirlos para tener una primera versión funcional.

B.11.3. Riesgos

En este *Sprint* no se disparó ningún riesgo.

B.12. Sprint 10: 01 de junio - 7 de junio

B.12.1. *User Stories* a desarrollar

ID	Descripción
US14	Como <i>Product Owner</i> quiero tener el plan de desarrollo de software completo.

Tabla B.11: *User Stories* para el Sprint 10

B.12.2. US14 - Como *Product Owner* quiero tener el plan de desarrollo de software completo.

B.12.2.1. Descripción

En esta historia de usuario se obtendrá una versión casi final del plan de desarrollo de software completándolo con los datos actuales del proyecto y viendo como los riesgos han influido en las estimaciones temporales.

B.12.2.2. Puntos

Esta *User Story* se decide que tenga 8 puntos ya la aplicación se encuentra en su fase final.

B.12.2.3. Tareas

- **T01** - Actualizar el modelo de desarrollo de software.
- **T01** - Comprobar la gestión de riesgos y la documentación de nuevos riesgos si han llegado a aparecer.

B.12.3. Riesgos

En este *Sprint* no se disparó ningún riesgo.

B.13. Sprint 11: 08 de junio - 15 de junio

B.13.1. *User Stories* a desarrollar

ID	Descripción
US15	Como <i>Product Owner</i> quiero tener el modelo de dominio casi completo.
US16	Como <i>Product Owner</i> quiero tener el diseño de software del sistema casi completo.

Tabla B.12: *User Stories* para el Sprint 11

B.13.2. US15 - Como *Product Owner* quiero tener el modelo de dominio casi completo.

B.13.2.1. Descripción

En esta historia de usuario se obtendrá una versión casi final del modelo de dominio de la aplicación actualizándolo con las últimas modificaciones hechas en la aplicación.

B.13.2.2. Puntos

Esta *User Story* se decide que tenga 4 puntos ya la aplicación se encuentra en su fase final. Además este modelo se ha ido actualizando a medida que se añadía funcionalidad.

B.13.2.3. Tareas

- **T01** - Inspeccionar el modelo de dominio de la aplicación.
- **T02** - Comprobar que lo desarrollado en la aplicación se corresponde con el modelo.
- **T03** - Realizar cambios en la aplicación si fuera necesario para que se adecue al modelo.

B.13.2.4. Comentarios

Como los modelos son algo que se van actualizando a lo largo del desarrollo del proyecto, en esta última historia de usuario se propone una versión casi final que será discutida después para su aprobación.

B.13.3. US16 - Como *Product Owner* quiero tener el diseño de software del sistema casi completo.

B.13.3.1. Descripción

En esta historia de usuario se obtendrá una versión casi final del diseño de software actualizándolo con las últimas modificaciones hechas en la aplicación.

B.13.3.2. Puntos

Esta *User Story* se decide que tenga 6 puntos ya que se ha de revisar la arquitectura de la aplicación con los diseños creados previamente y completar tanto uno como otro para que sean equivalentes

B.13.3.3. Tareas

- **T01** - Inspeccionar los diagramas de diseño y el diseño real de la aplicación
- **T02** - Comprobar no difieran. En el caso de que así sea corregirlo.
- **T03** - Actualizar los diagramas de diseño a una versión casi final.

B.13.3.4. Comentarios

Como el diseño de la aplicación es algo que se va actualizando a lo largo del desarrollo del proyecto, en esta última historia de usuario se propone una versión casi final que será discutida después para su aprobación.

B.13.4. Riesgos

Para este *sprint* se disparará el riesgo R 4.5 a causa de la detección de un error en el modelo de dominio de la aplicación, por lo que hubo que modificar los diagramas de diseño para cumplir con el nuevo modelo de domino.

B.14. Sprint 12: 19 de junio - 05 de julio

B.14.1. *User Stories* a desarrollar

ID	Descripción
US15	Como <i>Product Owner</i> tener la memoria del proyecto acabada.

Tabla B.13: *User Stories* para el Sprint 12

B.14.2. US17 - Como *Product Owner* tener la memoria del proyecto acabada

B.14.2.1. Descripción

En esta historia de usuario se obtendrá una versión final de la memoria que recoge el desarrollo del proyecto *SGEmployee*.

B.14.2.2. Puntos

Esta *User Story* se decide que tenga 10 puntos ya que se cuenta con la memoria casi terminada.

B.14.2.3. Tareas

- T01 - Revisar el contenido de la memoria
- T02 - Acometer las correcciones o mejoras que el tutor proponga.
- T03 - Realizar el manual de usuario.
- T04 - Realizar el resumen.
- T04 - Revisar por ultima vez el documento.

B.14.2.4. Comentarios

Última historia de usuario del proyecto en la que se corrige y mejora la memoria del proyecto para su entrega final.

Anexo C

Manual de Usuario

C.1. Introducción

Este documento detalla la forma de utilizar la aplicación *SGEmployee* por parte de un Empleado.

Se detallará cada una de las funcionalidades de la aplicación junto con imágenes ilustrativas.

C.2. Funcionalidades

C.2.1. Identificarse

C.2.1.1. Paso 1. Introducir las credenciales y pulsar en el botón de identificarse.

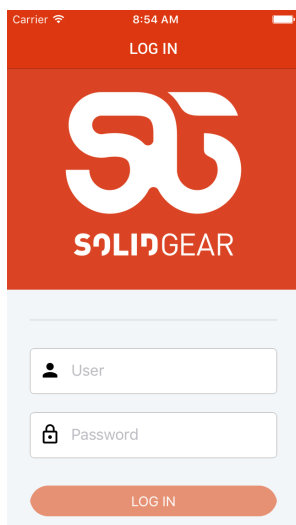


Figura C.1: Pantalla identificación.

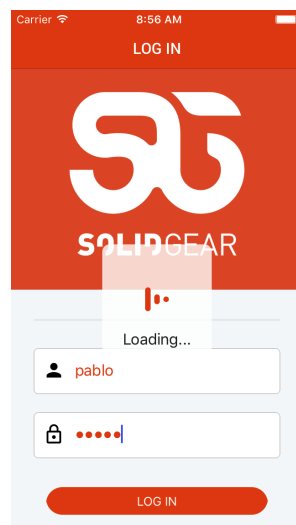


Figura C.2: Identificándose.

C.2.1.2. Paso 2. Dependiendo de si las credenciales son correctas, se procede a una pantalla u otra.

Cómo podemos ver en la Figura C4, si las credenciales son correctas, se muestra la pantalla principal es en la que se muestran los *tweets* de la compañía.

Si se pulsa sobre el botón *Olvidé la contraseña*, se abrirá una página web con el sistema que la empresa tenga habilitado para recuperación de contraseñas.

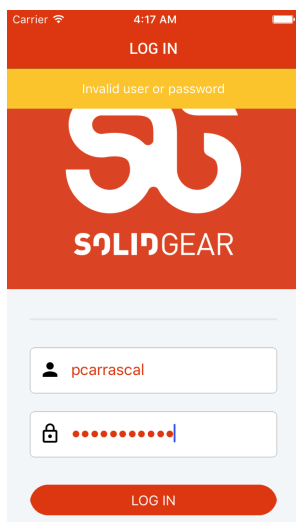


Figura C.3: Identificación fallida.



Figura C.4: Pantalla principal.

C.2.2. Abrir Y cerrar el menú

C.2.2.1. Paso 1. Pulsar sobre el botón menú para abrir o cerrar el menú



Figura C.5: Identificación fallida.

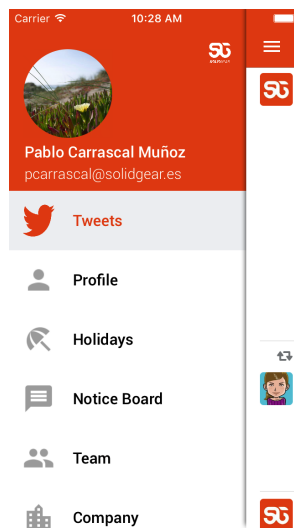


Figura C.6: Pantalla del perfil.

Si el menú está abierto y se pulsa sobre la imagen de perfil, se abrirá automáticamente la sección Perfil.

C.2.3. Ver Perfil

C.2.3.1. Paso 1. Pulsar sobre el botón perfil en el menú y se abrirá el perfil.

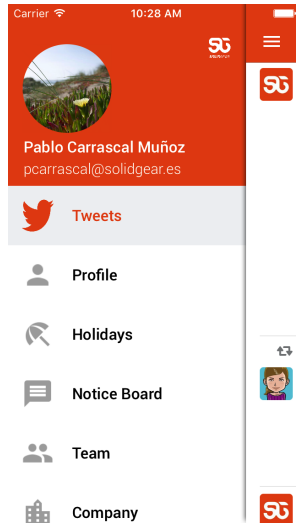


Figura C.7: Menú abierto.

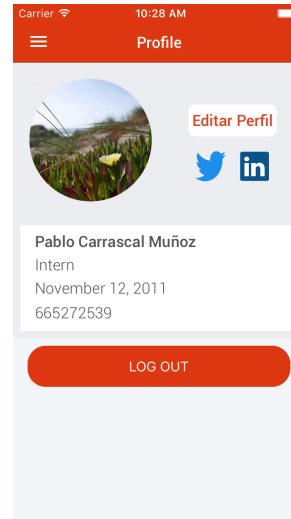


Figura C.8: Pantalla del perfil.

Si se pulsa en el botón *LOG OUT* se procederá a salir de la aplicación.

C.2.4. Editar Perfil

C.2.4.1. Paso 1. Se parte de la pantalla perfil y se pulsa en el botón Editar Perfil

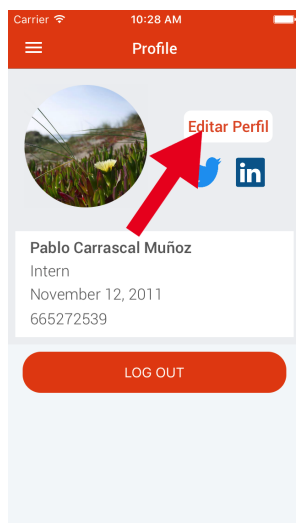


Figura C.9: Pantalla de perfil.

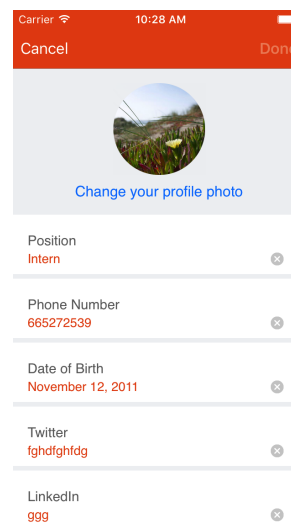


Figura C.10: Pantalla de editar perfil.

Para editar algún campo basta con pulsar sobre él. En el caso de la foto de perfil, basta con pulsar sobre el botón cambiar foto de perfil.

C.2.4.2. Paso 2. Si pulsas sobre el botón cancelar, los cambios no se guardan y se vuelve a la pantalla de perfil.

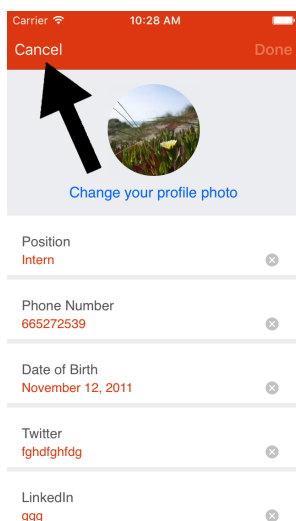


Figura C.11: Cancelar editar el perfil.

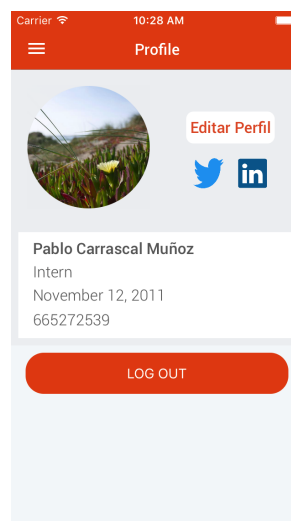


Figura C.12: Pantalla de perfil.

C.2.4.3. Paso 3. Si realizas algún cambio y pulsas sobre el botón hecho, los cambios se guardan y se vuelve a la pantalla perfil.

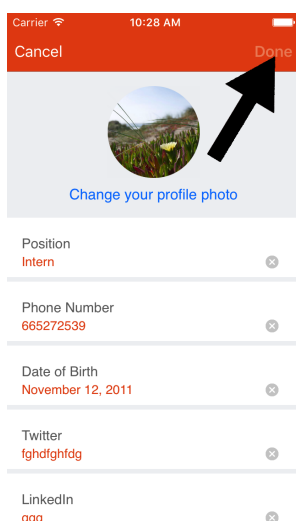


Figura C.13: Editar el perfil

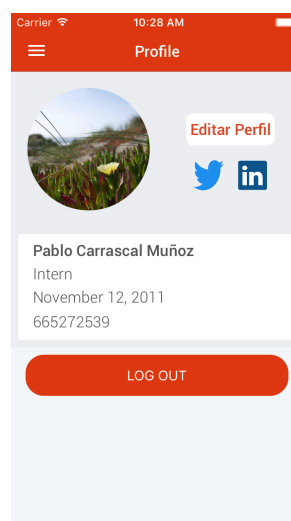


Figura C.14: Pantalla de perfil.

C.2.5. Cambiar ajustes

C.2.5.1. Paso 1. Pulsar sobre el botón ajustes en el menú y se abrirán los ajustes

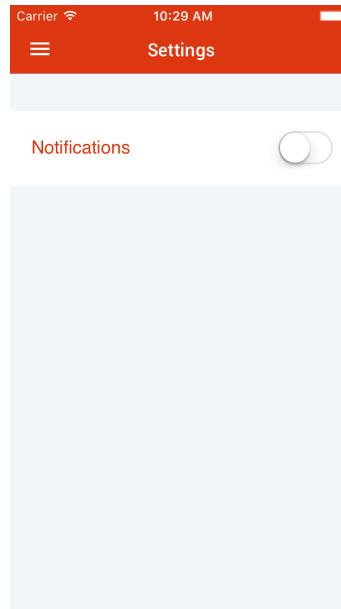


Figura C.15: Pantalla de ajustes.

Pulsando sobre el botón se activan o desactivan las notificaciones *Push*

C.2.6. Ver Compañeros de trabajo

C.2.6.1. Paso 1. Pulsar sobre el botón equipo en el menú y se abrirá la pantalla de equipo



Figura C.16: Lista de compañeros.



Figura C.17: Compañeros filtrados.

- Si se pulsa sobre el icono del teléfono, se llamará al compañero de trabajo.
- Si se pulsa sobre el icono de Twitter, se abrirá el perfil de Twitter del compañero.
- Si se pulsa sobre el icono de LinkedIn, se abrirá el perfil de LinkedIn del compañero.
- Si se pulsa sobre el email, se abrirá la aplicación por defecto para mandar un email a la dirección del compañero.

Todo esto claro está, si el compañero tiene actualizado su perfil, en caso contrario, el icono aparecerá como en las figuras C16 y C17 con un tono de color débil.

Como puede apreciarse en las figuras anteriores, si se pulsa sobre la barra de búsqueda, la interfaz se modifica a la Figura C17, pudiendo filtrar a los compañeros de trabajo por nombre o apellidos. En caso de no querer, bastará con pulsar cancelar.

C.2.7. Ver datos de la empresa

C.2.7.1. Paso 1. Pulsar sobre el botón compañía en el menú y se abrirá la pantalla de datos sobre la compañía.

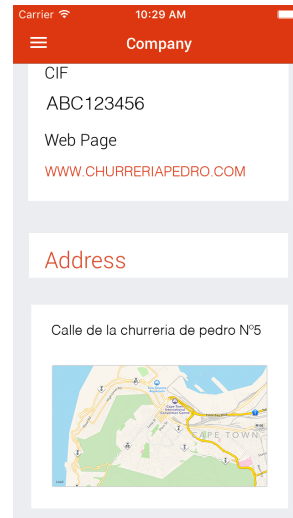
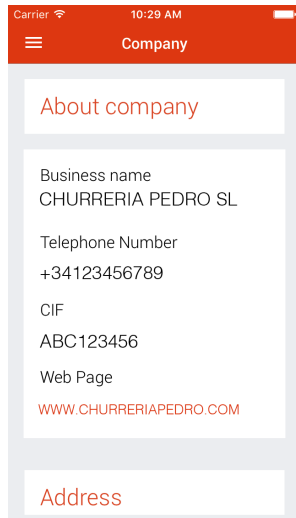


Figura C.18: Datos de la empresa **Figura C.19:** Dirección de la empresa

- Si se pulsa sobre el teléfono de la página web de la empresa, se abrirá la página web.
- Si se pulsa sobre el mapa se abrirá la aplicación *Maps* con la dirección de la empresa.

C.2.8. Ver vacaciones

C.2.8.1. Paso 1. Pulsar sobre el botón vacaciones en el menú y se abrirá la pantalla de ver vacaciones.

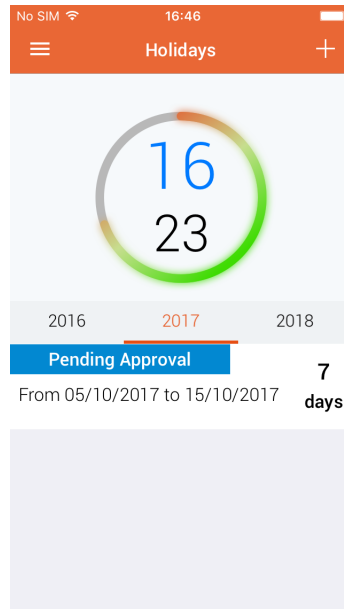


Figura C.20: Pantalla de ver vacaciones.

Se pueden ver dos zonas claramente diferenciadas:

- La zona en la que se muestran los días totales de vacaciones (23 en este caso) y los días restantes (16 en este caso).
- La zona en la que se muestran las vacaciones que han sido pedidas y sus detalles (En este caso 7 días pedidos y se encuentra en pendiente de aprobación). Aquí puede seleccionarse el año actual, el anterior y el siguiente.

C.2.9. Pedir vacaciones

C.2.9.1. Paso 1. Pulsar sobre el botón (+) pedir vacaciones.

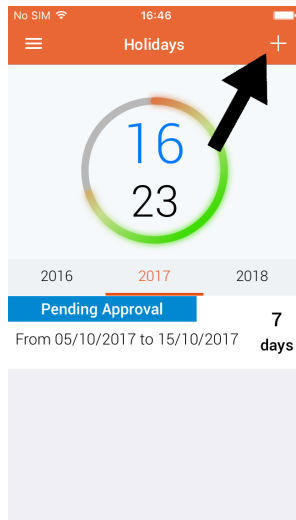


Figura C.21: Botón pedir vacaciones.

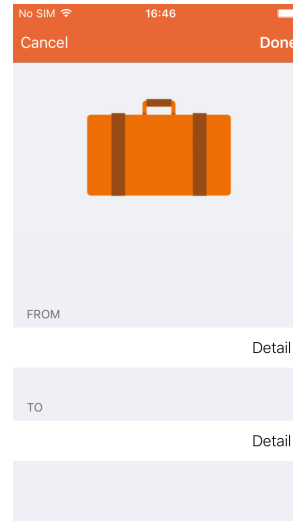


Figura C.22: Pantalla de pedir vacaciones.

C.2.9.2. Paso 2. Introducir fechas para las vacaciones y pulsar hecho.

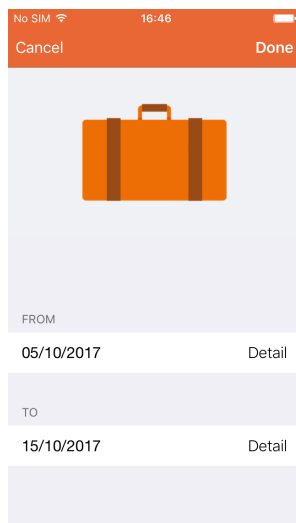


Figura C.23: Pantalla de pedir vacaciones.

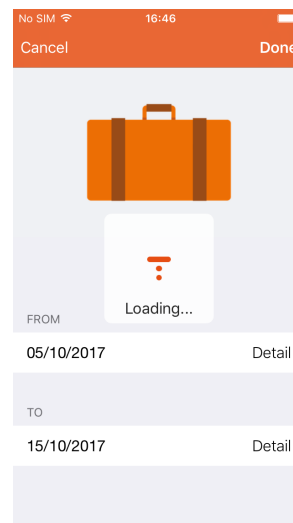


Figura C.24: Enviando datos al servidor.

- Si se pulsa en cancelar, se vuelve a la pantalla de ver vacaciones.
- Si se produce algún error, se avisa con un mensaje.

- Si se la petición se realiza correctamente, se vuelve a la pantalla de ver vacaciones.

C.2.10. Cancelar vacaciones

C.2.10.1. Paso 1. Deslizar a la derecha unas vacaciones. Estas han de estar en el estado pendiente de aprobación o aprobadas.

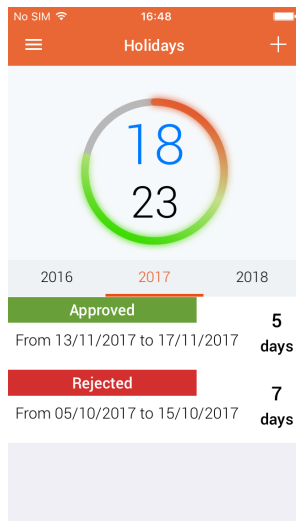


Figura C.25: Vacaciones.

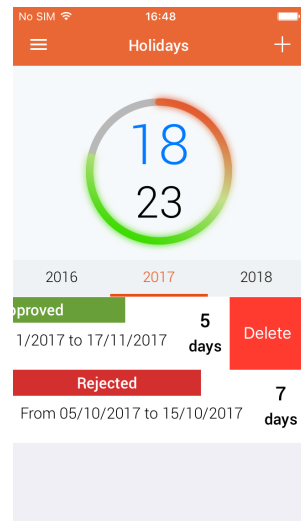


Figura C.26: Eliminar vacaciones.

C.2.10.2. Paso 2. Las vacaciones habrán cambiado de estado.

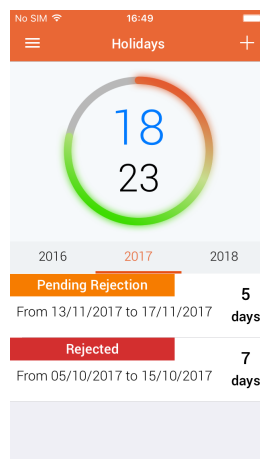


Figura C.27: Las vacaciones habrán cambiado de estado.

C.2.11. Ver avisos

C.2.11.1. Paso 1. Pulsar sobre el botón avisos en el menú y se abrirá la pantalla de avisos.

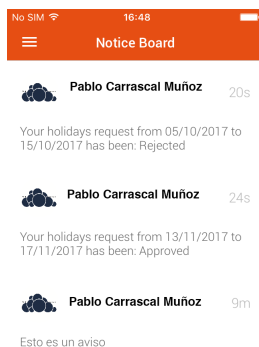


Figura C.28: Pantalla de avisos.

Anexo D

Manual de instalación

Precondiciones:

- Para poder compilar el proyecto es necesario contar con un ordenador con sistema operativo macOS y tener instalado Xcode en su versión 8 o superior.
- Es necesario tener *Ruby* instalado
- Es necesario tener instalada la gema *CocoaPods*

Instalación:

- Abrir la terminal e ir a la carpeta del proyecto.
- Ejecutar el comando
`pod install`

Esto instalará las dependencias del proyecto y creará un fichero llamado **SGEM.xcworkspace**

- Abrir Xcode y abrir el archivo de proyecto **SGEM.xcworkspace**
- Compilar y/o ejecutar el proyecto.