Universidad de Valladolid

Escuela de Ingeniería Informática

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática
(Mención Ingeniería del Software)

Aplicación para la gestión de tareas agrícolas

Autor:
D. Álvaro de Caso Morejón

Tutor:
Dr. Miguel A. Laguna

Resumen

En este Trabajo de Fin de Grado se presenta todo el proceso que se ha llevado a cabo para desarrollar un sistema que permita a agricultores gestionar las tareas que realizan en sus parcelas. Este proyecto surgió a partir de diferentes herramientas que se usan principalmente en el desarrollo de software y que permiten facilitar la organización de los proyectos.

La idea es la siguiente, estas herramientas permiten definir diferentes estados por los que tiene que pasar una tarea desde su creación hasta que se da por concluida. Además, en cada uno de estos estados hay ciertos atributos que hay que indicar para que la tarea pueda pasar al estado siguiente. Una vez que se ha creado la tarea, se le asignará un responsable que tendrá que desempeñarla, indicar los atributos propios de cada estado y actualizarla al estado siguiente hasta que esta tarea esté completada.

Además de gestión de las tareas, entre otras funcionalidades del sistema están la de gestión de aperos y parcelas de los agricultores, la gestión de los productos que ofertan las cooperativas y cuyos precios son necesarios para los agricultores para calcular los costes de desempeñar una tarea y la gestión de los usuarios que acceden al sistema de lo que se encargará un administrador.

Este proyecto se dará por completado cuando se haya generado toda la documentación necesaria para describir todo el proceso que ha tenido lugar desde la idea inicial del proyecto hasta el despliegue de un sistema que pueda ser utilizado por los diferentes usuarios y que reúna toda la funcionalidad que se describe en los primeros capítulos de esta memoria.

Índice general

Índice de figuras

Capítulo 1

Introducción

1.1. Motivación

En determinadas épocas del año los agricultores tienen gran carga de trabajo y que por determinadas circunstancias, como por ejemplo las condiciones meteorológicas, hace que este trabajo tenga que ser realizado de la manera más eficiente posible. Es por esto que una herramienta que permita a los agricultores conocer y organizar las tareas que tienen que realizar a corto plazo puede ayudarles a mejorar los tiempos que necesitan para completar las tareas. Otra de las funcionalidades que se creen interesantes sobre todo para agricultores más nóveles o cuando se trata de un cliente nuevo y del que no se conoce bien que parcelas trabaja es poder conocer la ubicación de la parcela donde hay que realizar las tareas. Esto es una alternativa a lo que se hace en la actualidad que consiste en mostrar las parcelas donde hay que desempeñar una tarea yendo a ver la parcela y recordando el camino recorrido. Para estos agricultores más nóveles que posiblemente no conozcan con precisión los caminos conocer la ubicación de la parcela y poder calcular la ruta con herramientas como Google Maps puede ser de gran utilidad además de reducir costes al no tener que recorrer las parcelas de las que no se conoce la ubicación por otros medios (principalmente coche).

1.2. Finalidad, alcance y objetivos

El objetivo de este proyecto es crear un sistema web que permita a agricultores que se hayan dado de alta en él gestionar sus tareas. La funcionalidad principal del sistema es la creación de tareas y su actualización a medida que estas van cambiando de estado. Como no todos los agricultores son capaces de completar todas sus tareas el sistema también permitirá que unos agricultores deleguen tareas en otros. Otras funcionalidades del sistema serán la gestión de parcelas, permitiendo visualizarlas en una lista y en un mapa, la gestión de los aperos de un agricultor y la gestión de precios de productos que ofertan las cooperativas. De las operaciones relacionadas con las cooperativas se encargará el encargado que también tendrá que estar dado de alta en el sistema. Además, el sistema permitirá al administrador del sistema gestionar cuentas bloqueándolas y desbloqueándolas. Para que los agricultores y los responsables de las cooperativas puedan darse de alta, tendrán que rellenar un formulario básico cuyos campos se especificarán más adelante.

1.3. Entregables del proyecto

Al final de proyecto se entregarán:

- Una memoria que contendrá el plan de proyecto, un documento de análisis, un documento de diseño, descripción de la implementación, descripción de las pruebas realizadas, manual de despliegue y manual de usuario.

- La última versión del código fuente del sistema.

1.4. Estructura de la memoria

La estructura de la memoria es la siguiente:

1. Un primer capítulo de introducción indicando la motivación para desarrollar este Trabajo de Fin de Grado y sus objetivos.
2. Un capítulo donde se presentará el plan de proyecto explicando la metodología que se va a seguir, la calendarización de este y el plan de control.
3. Un capítulo por cada una de las disciplinas del Proceso Unificado:
   - Elicitación de requisitos y análisis: se describirán los requisitos que debe de cumplir el sistema, los casos de uso de los diferentes actores, el modelo de dominio propuesto para este sistema y varios diagramas de secuencia de los casos de uso críticos de la aplicación.
   - Diseño del sistema: se especificará la estructura de los recursos de la API REST, el diseño de la base de datos, las arquitecturas de las diferentes aplicaciones, los diagramas de secuencia de los casos de uso críticos del sistema y por último los prototipos de las vistas de la aplicación.
   - Implementación: se expondrá algunas de las características más importantes de la implementación de las diferentes aplicaciones.
   - Pruebas: se describirán las diferentes pruebas que se han realizado a las diferentes aplicaciones.
4. Un capítulo donde se explique cómo y donde se realizará el despliegue de la aplicación.
5. Un capítulo donde se exponga cómo ha sido el desarrollo del proyecto respecto a las estimaciones iniciales.
6. Un último capítulo con las conclusiones obtenidas al finalizar el trabajo y posible trabajo futuro.
7. Sección de anexos:

   A Manual de usuario para que los diferentes usuarios aprendan cómo utilizar la aplicación.

Capítulo 2

Plan de proyecto

2.1. Organización del proceso

2.1.1. Modelo de proceso

La metodología que se va a emplear va a ser el *Proceso unificado (PU)*. Se ha elegido esta opción en lugar de una metodología ágil como por ejemplo *SCRUM* por las siguientes razones:

1. El sistema que se va a desarrollar está bien definido: no se prevén cambios que vayan a afectar de forma considerable al desarrollo del proyecto aunque siempre hay que tener en cuenta un posible cambio en alguna de las funcionalidades ya descritas pero si los hubiese, PU plantea mecanismos para la incorporación de estos.

2. Ausencia de algunos roles: en metodologías ágiles como *SCRUM* existen determinados roles como el *Scrum master* y el *Product owner* que en equipos pequeños, en este caso formado por una sola persona, no se podrían llevar a cabo sus funciones.

3. Familiarizarse con las tecnologías y herramientas: durante las fases iniciales del proyecto, antes de llegar a la *fase de Construcción*, se dedicará tiempo a la formación en determinados aspectos y así reducir riesgos relacionados con la falta de experiencia.

2.1.2. Proceso unificado

El *PU* se divide en cuatro fases. Estas fases se subdividen en iteraciones y cada iteración esta formada por disciplinas: *Requisitos, Análisis y Diseño, Codificación, Pruebas, Administración del Proyecto y Gestión de la configuración y cambios.*

Fases del Proceso unificado

1. **Fase de inicio:** en esta fase se define el alcance del proyecto teniendo en cuenta los casos de uso y estableciendo cuáles son los casos de uso críticos, identificando los riesgos, los recursos de los que se dispone y realizando un plan de fases. Cuando esta fase finalice, se tendrá al menos:

   - Un documento con los requisitos del proyecto y sus restricciones.
   - Un primer modelo de casos de uso.
   - Un plan de proyecto teniendo en cuenta fases e iteraciones.
   - Estudio inicial de los posibles riesgos del proyecto.

2. **Fase de elaboración:** en esta fase se establece una arquitectura para el proyecto ademas, se definen la mayor parte de los requisitos del sistema y toman las decisiones y medidas para reducir los riesgos con mayor impacto. Al finalizar esta fase se habrá obtenido al menos:

   - Un documento con las descripciónes de los casos de uso y descripción de requisitos no funcionales.
   - Un documento con la descripción de la arquitectura del sistema.

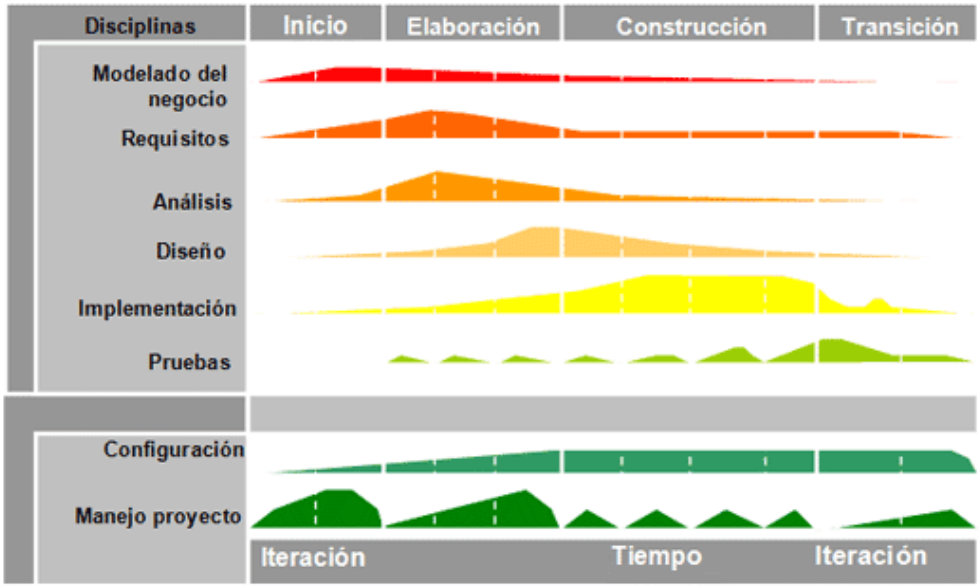

Figura 2.1: Proceso unificado: fases, iteraciones y disciplinas.

- Plan para el resto del proyecto.

3. **Fase de construcción:** durante esta fase se desarrollan diferentes versiones del producto buscando la mejor calidad posible y optimizando los costes del proyecto. Al finalizar esta fase se habrá obtenido una versión beta del producto.

4. **Fase de transición:** durante esta fase se busca la aceptación del cliente y que el producto sea lo más eficiente posible. También, durante esta fase se deberá buscar una alternativa para el despliegue del producto en el entorno donde se vaya a utilizar.

Roles y responsabilidades

Que no haya un equipo como tal sino que sólo haya una persona encargada del todo el proyecto implica que este asumirá todos los roles/responsabilidades a lo largo del del desarrollo del proyecto. Por la misma razón, que no haya varias personas formando el equipo, no habrá diferentes puntos de vista ante situaciones que puedan generar dudas o personas especializadas en determinadas tareas cuya opinión tenga más peso en el equipo por lo que ante cualquier duda será necesario buscar documentación de otras fuentes. Como personas externas al equipo se buscará posibles usuarios del sistema, en este caso agricultores y personas encargadas de la gestión de cooperativas, para aclarar posibles dudas y realizar pruebas del sistema.

2.2. Plan del proceso de gestión

2.2.1. Inicio del proyecto

Durante esta fase se prepararán el entorno, las herramientas y las tecnologías que se usarán. Estas se explican con más detalle en el punto *Plan de proceso técnico*. También, antes de comenzar el proyecto se dedicará tiempo a la formación en las tecnologías y herramientas para evitar posibles futuros problemas con estas.

2.2.2. Plan de trabajo

Como el *PU* es un desarrollo iterativo, en primer lugar, se hará un *Plan de fases* donde se dónde se explicará los objetivos y número de iteraciones y después se realizará un *Plan de iteración* donde se explicará con detalle las actividades que se realizarán en cada iteración. La calendarización se ha realizado considerando 6 horas de trabajo al día y 5 días a la semana.

Plan de fases

El número de iteraciones que se realizarán en cada fase son los siguientes:

Fase	**Nº de iteraciones**
Inicio	1
Elaboración	3
Construcción	2
Transición	1

Tabla 2.1: Número de iteraciones de cada fase.

A continuación, se describe lo que se va a realizar en cada iteración:

- Fase de Inicio - Iteración 1: durante esta iteración se definirán las funcionalidades básicas del sistema, se identificarán los riesgos con mayor impacto y se dedicará tiempo a familiarizarse con las tecnologías y las herramientas. Durante esta iteración se desarrollará el *Plan de proyecto*.
- Fase de Elaboración - Iteración 1: se especificarán los primeros requisitos del sistema y se describirán los casos de uso más críticos para el sistema. También se tomarán medidas para reducir los riesgos con un mayor impacto.
- Fase de Elaboración - Iteración 2: se revisará y modificará si es necesario los requisitos y casos de uso descritos en la primera iteración. Una vez se hayan realizado las correcciones necesarias,se continuará con el análisis del sistema.
- Fase de Elaboración - Iteración 3: se comenzará el diseño del sistema y se realizarán los prototipos de las vistas de la aplicación.
- Fase de Construcción - Iteración 1: se desarrollará el backend de la aplicación.
- Fase de Construcción - Iteración 2: se desarrollarán las vistas de las aplicaciones tanto la parte web como la parte móvil y sus funcionalidades.
- Fase de Transición - Iteración 1: se corregirán fallos de la aplicación que hayan surgido haciendo las pruebas y se estudiarán las solicitudes de cambios de los *stakeholders*. Cuando se hayan completado los cambios, se desarrollará el *Manual de usuario* y se estudiarán diferentes alternativas para desplegar la aplicación.

Plan de iteraciones

A continuación, se incluyen las actividades que se van a realizar en cada iteración incluyendo su identificador, su nombre, su duración estimada, su fecha de comienzo estimada, su fecha de fin estimada, sus actividades predecesoras y el diagrama de Gantt.

1	**Comienzo Fase Inicio - Iteración 1**	0 horas	lun 18/11/19	lun 18/11/19	
2	Visión general proyecto	12 horas	lun 18/11/19	mar 19/11/19	1
3	Familiarizarse herramientas/tecnol	12 horas	lun 18/11/19	mar 19/11/19	1
4	Elicitación primeros requisitos	5 horas	mar 19/11/19	mié 20/11/19	2
5	Plan de fases	5 horas	mar 19/11/19	mié 20/11/19	2
6	Plan de iteraciones	9 horas	mié 20/11/19	jue 21/11/19	5
7	Plan de riesgos	5 horas	mar 19/11/19	mié 20/11/19	2;3
8	Plan de iteracion siguiente	3 horas	jue 21/11/19	jue 21/11/19	4;5;7;6
9	**Fin Fase Inicio - Iteración 1**	0 horas	jue 21/11/19	jue 21/11/19	8

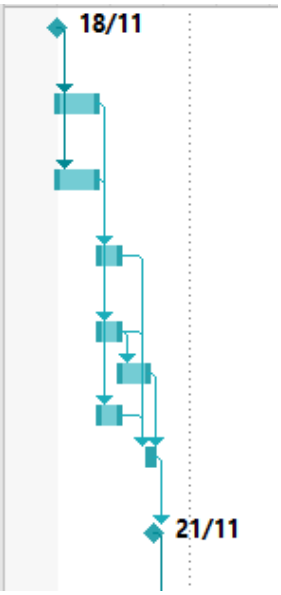

Figura 2.2: Calendarización Fase Inicio - Iteración 1.

10	**Comienzo Fase Elaboración - Iteración 1**	0 días	jue 21/11/19	jue 21/11/19	9
11	Elicitación de requisitos	12 horas	jue 21/11/19	lun 25/11/19	10
12	Diagrama de casos de uso	5 horas	lun 25/11/19	lun 25/11/19	11
13	Descripción de los casos de uso	15 horas	lun 25/11/19	mié 27/11/19	12
14	Plan de iteración siguiente	3 horas	mié 27/11/19	mié 27/11/19	12;13
15	**Fin Fase Elaboración - Iteración 1**	0 horas	jue 28/11/19	jue 28/11/19	14

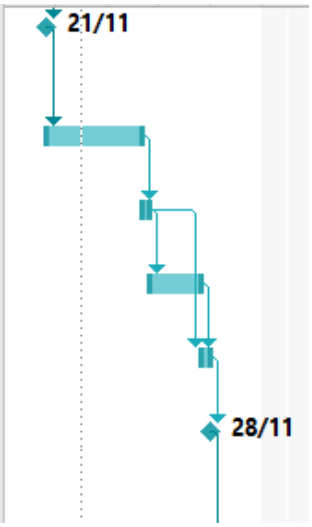

Figura 2.3: Calendarización Fase Elaboración - Iteración 1.

16	**Comienzo Fase Elaboración - Iteración 2**	0 horas	jue 28/11/19	jue 28/11/19	15
17	Revisión requisitos	5 horas	jue 28/11/19	jue 28/11/19	16
18	Revisión de los casos de uso	5 horas	jue 28/11/19	vie 29/11/19	17
19	Modelo de dominio en análisis	7 horas	vie 29/11/19	lun 02/12/19	18
20	Modelo de datos	3 horas	lun 02/12/19	lun 02/12/19	19
21	Diagramas de secuencia de los casos de uso críticos	15 horas	vie 29/11/19	mar 03/12/19	18
22	Plan de iteración siguiente	3 horas	mar 03/12/19	mar 03/12/19	21;20
23	**Fin Fase Elaboración - Iteración 2**	0 horas	mar 03/12/19	mar 03/12/19	22

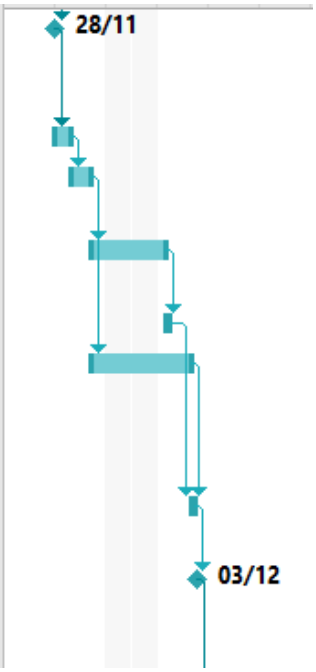

Figura 2.4: Calendarización Fase Elaboración - Iteración 2.

24	**Comienzo Fase Elaboración - Iteración 3**	0 horas	mar 03/12/19	mar 03/12/19	23
25	Diseño de la arquitectura front-end	5 horas	mar 03/12/19	mié 04/12/19	24
26	Diseño de la arquitectura del back-end	15 horas	mar 03/12/19	jue 05/12/19	24
27	Diseño de las vistas	10 horas	mié 04/12/19	jue 05/12/19	25
28	Diseño de la API REST	10 horas	jue 05/12/19	vie 06/12/19	26
29	Modelo de dominio en diseño	7 horas	jue 05/12/19	vie 06/12/19	26
30	Diagramas de secuencia de los casos de uso críticos en diseño	15 horas	vie 06/12/19	mar 10/12/19	28;29
31	Plan de iteración siguiente	3 horas	mar 10/12/19	mar 10/12/19	27;30
32	**Fin Fase Elaboración - Iteración 3**	0 horas	mar 10/12/19	mar 10/12/19	31

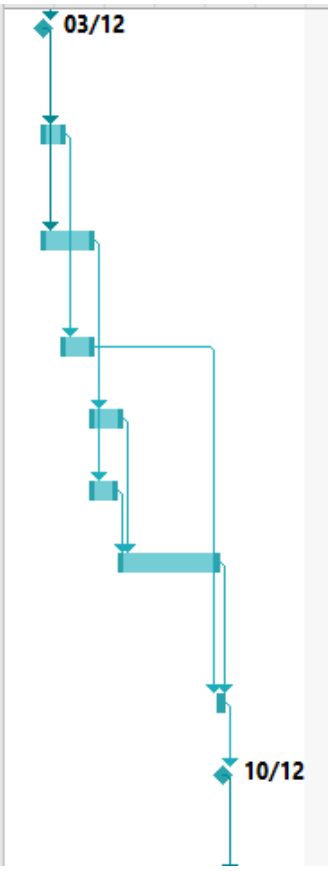

Figura 2.5: Calendarización Fase Elaboración - Iteración 3.

33	**Comienzo Fase Construcción - Iteracción 1**	0 horas	mar 10/12/19	mar 10/12/19	32
34	Implementación de la lógica de la aplicación	60 horas	mar 10/12/19	vie 20/12/19	33
35	Implementación del acceso a persistencia	30 horas	vie 20/12/19	jue 26/12/19	34
36	Implementación de la API REST	30 horas	jue 26/12/19	mar 31/12/19	35
37	Pruebas de los elementos desarollados en esta fase	10 horas	mar 31/12/19	jue 02/01/20	36
38	Plan iteración siguiente	3 horas	jue 02/01/20	jue 02/01/20	37
39	**Fin Fase Construcción - Iteracción 1**	0 horas	jue 02/01/20	jue 02/01/20	38

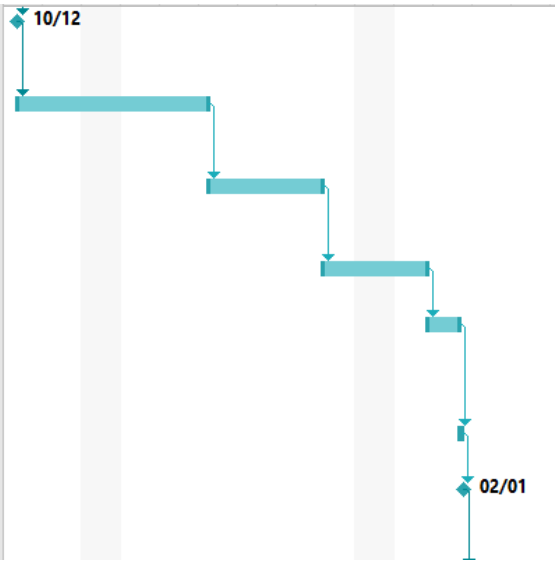

Figura 2.6: Calendarización Fase Construcción - Iteración 1.

40	**Cominezo Fase Contrucción - Iteracción 2**	0 horas	jue 02/01/20	jue 02/01/20	39
41	Implementación de las vistas de la parte web	20 horas	jue 02/01/20	lun 06/01/20	40
42	Implementación de las vistas de la parte móvil	10 horas	jue 02/01/20	vie 03/01/20	40
43	Implementación de la lógica de la parte web	70 horas	mar 07/01/20	vie 17/01/20	41
44	Implementación de la lógica de la parte móvil	30 horas	vie 03/01/20	jue 09/01/20	42
45	Plan iteración siguiente	3 horas	jue 23/01/20	jue 23/01/20	43;44
46	**Fin Fase Construcción - Iteracción 2**	0 horas	jue 23/01/20	jue 23/01/20	45

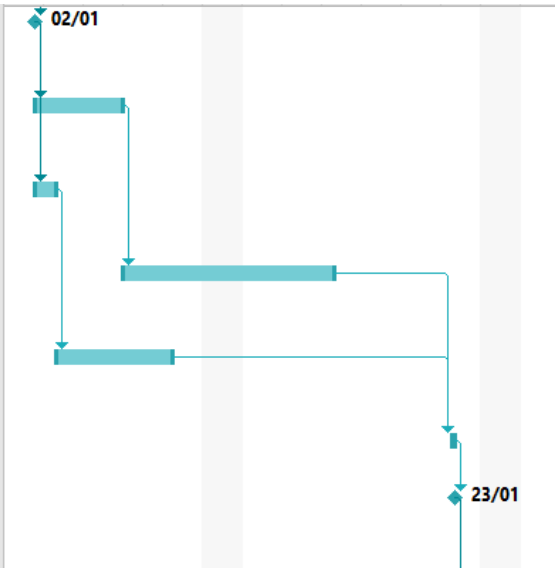

Figura 2.7: Calendarización Fase Construcción - Iteración 2.

47	**Comienzo Fase Transición - Iteración 1**	0 horas	jue 23/01/20	jue 23/01/20	46
48	Prueba de los stakeholder	10 horas	jue 23/01/20	lun 27/01/20	47
49	Correcciones y mejoras	10 horas	lun 27/01/20	mar 28/01/20	48
50	Manual de usuario	5 horas	mar 28/01/20	mar 28/01/20	49
51	despliegue	5 horas	mié 29/01/20	mié 29/01/20	50
52	**Fin fase Transición - Iteración 1**	0 horas	mié 29/01/20	mié 29/01/20	51

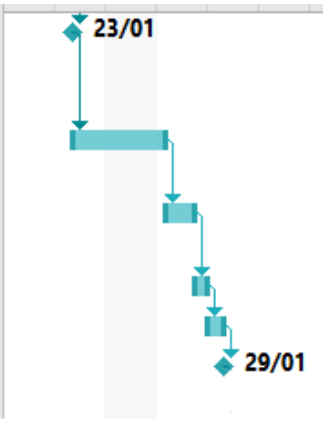

Figura 2.8: Calendarización Fase Transición - Iteración 1.

2.2.3. Plan de control del proyecto

Gestión de requisitos

Se comenzará a trabajar con los requisitos que se han definido al comienzo del proyecto, aunque puede darse el caso que en fases avanzadas del proyecto se requiera un cambio o una nueva funcionalidad. Si se diese este caso, antes de continuar con el proyecto, habría que estimar el cambio o la nueva funcionalidad, estudiar que partes del trabajo ya realizado se verían afectadas y realizar los cambios pertinentes sin afectar a otras partes del sistema.

Gestión de gastos y presupuestos

A continuación se exponen los costes estimados:

- Coste de personal: considerando que un programador tiene un sueldo medio de 18€por hora y que el proyecto está estimado en 485 horas el coste sería de 18*485 = 4365€.
- Coste hardware: la máquina que se va a emplear tiene un coste aproximado de 1200€y se le estima una vida útil de 5 años por lo tanto el coste del hardware es: 3/(5*12)*1200 = 60€.
- Coste software: el software utilizado es de código abierto o se dispone de licencia gratuita por lo tanto 0€.

Por lo tanto, el gasto total estimado es de 8730 + 60 + 0 = 8790 €.

Gestión de riesgos del proyecto

Para describir los posibles riesgos que se pueden dar a lo largo del proyecto se seguirá una plantilla con los siguientes campos:

- Identificador del riesgo.
- Nombre del riesgo.
- Descripción del riesgo.
- Probabilidad del riesgo. Este campo podrá tener uno de los siguientes valores: muy bajo, bajo, medio, alto, muy alto.
- Impacto del riesgo.
- Exposición: su valor dependerá de la siguiente tabla:

Impacto/Probabilidad	**Muy alto**	**Alto**	**Medio**	**Bajo**	**Muy bajo**
Catastrófico	Alto	Alto	Moderado	Moderado	Bajo
Crítico	Alto	Alto	Moderado	Bajo	Ninguno
Marginal	Moderado	Moderado	Bajo	Ninguno	Ninguno
Depreciable	Moderado	Bajo	Bajo	Ninguno	Niniguno

Tabla 2.2: Matriz Impacto/Probabilidad

- Proteccion del riesgo.
- Plan de acción.

Los riesgos que se han tenido en cuenta en este proyecto son:

Identificador del riesgo	R1
Nombre del riesgo	Enfermedad.
Descripción del riesgo	La ausencia de un equipo y de que una persona sea la encargada de todo el proyecto puede suponer que ante una baja de este el proyecto se alargue más de lo estimado en un principio.
Probabilidad del riesgo	Baja.
Impacto del riesgo	Crítico.
Exposición al riesgo	Bajo.
Protección del riesgo	Ninguno.
Plan de acción	Reestimar la duración del proyecto.

Tabla 2.3: Riesgo R1: Enfermedad

Identificador del riesgo	R2
Nombre del riesgo	Perdida parcial o total de documentación o código.
Descripción del riesgo	Pérdida del trabajo realizado hasta un instante determinado.
Probabilidad del riesgo	Baja.
Impacto del riesgo	Catastrófico.
Exposición al riesgo	Moderado.
Protección del riesgo	Utilizar herramientas online y control de versiones así como clonar el trabajo en varios lugares.
Plan de acción	Evaluar el alcance de la perdida y reestimar la duración del proyecto teniendo en cuenta el tiempo que llevó realizar ese trabajo.

Tabla 2.4: Riesgo R2: Perdida parcial o total de documentación o código.

Identificador del riesgo	R3
Nombre del riesgo	Ausencia de diferentes puntos de vista o especialidades.
Descripción del riesgo	La ausencia de un equipo formado por varias personas hace que no se puedan proponer diferentes formas de realizar el trabajo o que no haya alguien especializado en determinadas tareas.
Probabilidad del riesgo	Medio.
Impacto del riesgo	Marginal.
Exposición al riesgo	Bajo.
Protección del riesgo	Buscar variantes a medida que se va realizando el trabajo y buscar documentación en cuanto surja alguna duda.
Plan de acción	Aplicar con criterio cambios siguiendo las fuentes consultadas.

Tabla 2.5: Riesgo R3: Ausencia de diferentes puntos de vista o especialidades.

Identificador del riesgo	R4
Nombre del riesgo	Cambio en los requisitos.
Descripción del riesgo	Con el proyecto en un estado medianamente avanzado, se requiere una nueva funcionalidad o que se modifique algo sobre lo que ya se había trabajado.
Probabilidad del riesgo	Medio.
Impacto del riesgo	Marginal.
Exposición al riesgo	Bajo.
Protección del riesgo	Ceñirse a la idea inicial sin proponer nuevas funcionalidades o modificaciones de las iniciales en fases avanzadas del proyecto.
Plan de acción	Reestimar las tareas que se ven afectadas por estos cambios e incorporarlos sin modificar nada que no sea estrictamente necesario.

Tabla 2.6: Riesgo R4: Cambio en los requisitos.

Identificador del riesgo	R5
Nombre del riesgo	Caducidad de licencias o cambios en las condiciones de uso.
Descripción del riesgo	A lo largo del proyecto se van a utilizar herramientas y tecnologías que dependen de una subscripción. Estas subscripciones pueden caducar o modificar sus condiciones de uso.
Probabilidad del riesgo	Medio.
Impacto del riesgo	Crítico.
Exposición al riesgo	Moderado.
Protección del riesgo	Comprobar antes del proyecto las condiciones de uso reales y hacer un estudio de diferentes alternativas a estas herramientas y tecnologías.
Plan de acción	Cambiar de herramienta o tecnología o emplear versiones gratuitas de estas subscripciones.

Tabla 2.7: Riesgo R5: Caducidad de licencias o cambios en las condiciones de uso.

Identificador del riesgo	R6
Nombre del riesgo	Uso de tecnologías y herramientas nuevas.
Descripción del riesgo	Se pretenden utilizar herramientas y tecnologías actuales con las que no se está familiarizado al cien por cien.
Probabilidad del riesgo	Medio.
Impacto del riesgo	Marginal.
Exposición al riesgo	Bajo.
Protección del riesgo	Documentarse sobre las herramientas más adecuadas para las tecnologías que se pretenden utilizar y comparar las diferentes herramientas.
Plan de acción	Ante cualquier duda, consultar la documentación de las herramientas o tecnologías o buscar información en otras fuentes.

Tabla 2.8: Riesgo R6: Uso de tecnologías y herramientas nuevas.

Identificador del riesgo	R7
Nombre del riesgo	Centrarse en la apariencia en lugar de en la funcionalidad.
Descripción del riesgo	Centrarse más en la apariencia de la parte front end de la aplicación dejando de lado la una correcta implementación del back end y de sus funcionalidades.
Probabilidad del riesgo	Alto.
Impacto del riesgo	Crítico.
Exposición al riesgo	Alto.
Protección del riesgo	Ajustarse a las estimaciones de tiempo sin dedicar tiempo a mayores al menos que no se haya completado el objetivo de la tarea.
Plan de acción	Optar por una apariencia más simple de la aplicación para no invertir más tiempo del estimado en un principio.

Tabla 2.9: Riesgo R8: Centrarse en la apariencia en lugar de en la funcionalidad.

Identificador del riesgo	R8
Nombre del riesgo	Diferentes alternativas para realizar el despliegue.
Descripción del riesgo	En su punto correspondiente se hará un estudio de las diferentes alternativas para realizar el despliegue de la aplicación exponiendo sus puntos a favor y en contra.
Probabilidad del riesgo	Alto.
Impacto del riesgo	Crítico.
Exposición al riesgo	Alto.
Protección del riesgo	Hacer un estudio de las diferentes formas de realizar el despliegue estudiando sus puntos a favor y en contra.
Plan de acción	Aplicar la alternativa que se identifique como la más adecuada.

Tabla 2.10: Riesgo R8: Uso de tecnologías y herramientas nuevas.

2.2.4. Plan de proceso técnico

Herramientas y tecnologías

A lo largo de proyecto se utilizarán las siguientes tecnologías:

- Para el desarrollo del frontend se usará *Angular*, un framework desarrollado en *TypeScript* por *Google*. La versión mínima de este framework que se usará será la 8. También será necesario emplear *HTML* y *CSS* para desarrollar las vistas de la aplicación.
- Para el desarrollo del backend se usará *Spring Boot* un framework de *Java*. Como sistema gestor de bases de datos se usará *PostgreSQL*.
- Java para el desarrollo de la aplicación móvil.
- *UML* para realizar los diagramas en las fases *Elicitación de requisitos y análisis* y *Diseño del sistema*.

Respecto a las herramientas se usarán:

- *Visual Studio Code*, un entorno de desarrollo integrado que con su sistema de extensiones resulta muy útil para desarrollar las diferentes tecnologías que engloba este proyecto.
- *Android Studio* para el desarrollo de la aplicación móvil.
- *Postman* para probar las respuestas de la *API REST*.
- *pgadmin3* para administrar la base de datos.
- *Astah UML* para realizar los diagramas.
- *VirtualBox* para crear la máquina virtual donde se desarrollará el proyecto.
- *Google Chrome* y *Mozilla Firefox* para comprobar el código desarrollado.
- *Project* para realizar la calendarización del proyecto.
- También se usará draw.io, una herramienta online para el diseño de las vistas.

Infraestructura

El proyecto se desarrollará en una máquina virtual creada con *VirtualBox* que estará hospedada en una máquina con sistema operativo *Windows 10*. La máquina virtual tiene las siguientes características:

- Sistema operativo: *Ubuntu 16.04 LTS*.
- Almacenamiento: 100 GB.
- Memoria: 8 GB.
- Procesador: 2 procesadores lógicos de un *Intel i7-9750H*.

Si fuese necesario a lo largo del proyecto, se podrían ampliar tanto la memoria como el número de procesadores.Para el desarrollo de la aplicación Android se usará el siguiente dispositivo:

- Marca: OnePlus.
- Modelo: 6T.
- Memoria: 8GB.
- Almacenamiento: 128GB.
- Tamaño: 6.41".
- Versión de Android: 9.

Respecto al entorno de producción, habrá que realizar un estudio sobre la manera en la que se va a desplegar el sistema.

Capítulo 3

Elicitación de requisitos y análisis

3.1. Requisitos del sistema

3.1.1. Requisitos funcionales

RF-1 El sistema deberá diferenciar usuarios de tipo *Agricultor*, usuarios de tipo *Administrador* y usuarios de tipo *Responsable de cooperativa*.

RF-2 El sistema deberá permitir a los usuarios de tipo *Agricultor* y *Responsable de cooperativa* registrarse.

RF-3 El sistema deberá permitir al usuario *Agricultor* crear una tarea indicando la parcela donde la quiere realizar y el tipo de labor que quiere hacer.

RF-4 El sistema deberá permitir al usuario *Agricultor* escoger entre realizar el mismo la tarea o delegarla en otro agricultor.

RF-5 El sistema deberá permitir al usuario *Agricultor* en tareas en las que sea necesario emplear algún producto, indicar el nombre/s de este/estos.

RF-6 El sistema deberá permitir al usuario *Agricultor* encargado de realizar una tarea con estado pendiente de estimación indicar el día en el que se pretende empezar a realizar la tarea, el tiempo que estima que va a ser necesario para completarla y el apero que va a emplear.

RF-7 El sistema deberá permitir al usuario *Agricultor* encargado de realizar una tarea en estado pendiente de estimación y que requiera algún tipo de producto para que esta pueda ser realizada indicar las unidades de producto que estima necesarias por hectárea y el precio por unidad.

RF-8 El sistema deberá permitir al usuario *Agricultor* encargado de una tarea con estado estimada indicar cuántas horas y qué días ha trabajado en esa tarea.

RF-9 El sistema deberá permitir al usuario *Agricultor* encargado de una tarea con estado estimada indicar, si ha sido necesario emplear algún producto, la cantidad por hectárea que se ha necesitado para realizar la tarea.

RF-10 El sistema deberá permitir al usuario *Agricultor* indicar la cantidad de producto y el tipo recogido por hectárea si al completar la tarea se ha recolectado algún tipo de producto.

RF-11 El sistema deberá permitir al usuario *Agricultor* encargado de una tarea que ha sido delegada generar un documento en el que se tenga en cuenta el coste por hectárea del apero que ha empleado y la cantidad de producto que haya empleado si este fuera necesario.

RF-12 El sistema deberá permitir al usuario *Agricultor* ver las tareas que tiene planeadas para realizar ese mismo día.

RF-13 El sistema deberá permitir al usuario *Agricultor* que ha creado una tarea desecharla si esta está en los estados pendiente de estimación o estimada.

RF-14 El sistema deberá permitir al usuario *Agricultor* registrar nuevas parcelas.

RF-15 El sistema deberá permitir al usuario *Agricultor* ver un listado de todas las parcelas que ha registrado en el sistema.

RF-16 El sistema deberá permitir al usuario *Agricultor* ver en un mapa las parcelas que ha registrado y que tienen una ubicación.

RF-17 El sistema deberá permitir al usuario *Agricultor* ver los detalles de una parcela.

RF-18 El sistema deberá permitir al usuario *Agricultor* ver el historial de las tareas que se han desarrollado en una parcela.

RF-19 El sistema deberá permitir al usuario *Agricultor* registrar nuevos aperos.

RF-20 El sistema deberá permitir al usuario *Agricultor* ver un listado de todos los aperos que tiene registrados.

RF-21 El sistema deberá permitir al usuario *Agricultor* ver los detalles de un apero.

RF-22 El sistema deberá permitir al usuario *Agricultor* añadir otro agricultor a su agenda.

RF-23 El sistema deberá permitir al usuario *Agricultor* eliminar un usuario de su agenda.

RF-24 El sistema deberá permitir al usuario *Administrador* ver una lista de todos los usuarios que hay en el sistema.

RF-25 El sistema deberá permitir usuario *Administrador* bloquear cuentas de otros usuarios.

RF-26 El sistema deberá permitir usuario *Administrador* desbloquear cuentas de otros usuarios bloqueados.

RF-27 El sistema deberá permitir al usuario *Responsable de cooperativa* generar un código único para cada para que los usuarios de tipo *Agricultor* puedan unirse a la cooperativa.

RF-28 El sistema deberá permitir al usuario *Responsable de cooperativa* ver un listado de los productos.

RF-29 El sistema deberá permitir al usuario *Responsable de cooperativa* añadir nuevos productos.

RF-30 El sistema deberá permitir al usuario *Responsable de cooperativa* eliminar productos.

RF-31 El sistema deberá permitir al usuario *Responsable de cooperativa* indicar precios de productos.

RF-32 El sistema deberá permitir al usuario *Agricultor* ver los precios que se han propuesto en la cooperativa de la que forma parte.

3.1.2. Requisitos no funcionales

RNF-1 El sistema deberá funcionar en los navegadores *Google Chrome* y *Mozilla Firefox*.

RNF-2 El sistema utilizará *PostgreSQL* como sistema de gestión de bases de datos.

RNF-3 El sistema sólo permitirá a las peticiones autenticadas acceder a los datos de la aplicación.

RNF-4 La unidad de superficie por defecto que utilizará el sistema será la hectárea.

RNF-5 La moneda por defecto que utilizará el sistema será el euro.

RNF-6 La parte móvil de la aplicación estará disponible para dispositivos *Android*

3.1.3. Reglas de negocio

RN-1 Los estados de una tarea son pendiente de estimación, estimada y completada.

RN-2 Para que una tarea pueda ser creada, el agricultor encargado de completarla tiene que tener un apero capaz de realizar esta tarea.

RN-3 Para que un agricultor pueda delegar una tarea en otro, el segundo tiene que estar en la agenda del primero.

RN-4 Para que una tarea en estado pendiente de estimación pase al estado estimada, el apero que se indica en el formulario tiene que ser apto para completar esa tarea.

RN-5 Para que una tarea con estado estimada pueda ser desechada, el usuario *Agricultor* tiene que hacerlo con al menos dos días de antelación a la fecha que se tenía planeado realizar la tarea.

3.1.4. Requisitos de información

RI-1 De los agricultores se guardarán los siguientes campos:

a) Nombre completo.

b) Correo electrónico.

c) Contraseña.

d) Municipio.

e) Si la cuenta ha sido bloqueada.

RI-2 De las parcelas se guardarán los siguientes campos:

a) Un identificador.

b) Nombre.

c) Descripción.

d) Localidad en la que se encuentra.

e) Tamaño de la parcela en hectáreas.

f) Las coordenadas de la parcela.

g) El propietario.

RI-3 De un apero se guardará:

a) Un identificador.

b) Marca.

c) Modelo.

d) Matrícula.

e) Descripción.

f) Coste por hectárea trabajada.

g) Los tipos de labores que puede realizar.

h) El propietario.

RI-4 De una tarea se guardará:

a) Identificador de la tarea.

b) Estado de la tarea.

c) Nombre/s de los producto/s que habría que emplear si fuese necesario.

d) Fecha en la que se tiene pensado realizar la tarea.

e) Tiempo que se estima que va a ser necesario para realizar la tarea.

f) Estimación de las unidades de producto por hectárea necesarias para realizar la tarea.

g) Precio del producto que se quiere emplear.

h) Registros horas-días en los que se ha trabajado en la tarea.

i) Cantidad de producto/s real/es por hectárea que se utilizó para realizar la tarea.

j) Cantidad de producto/s que se recogió como resultado de realizar determinados tipos de labores.

k) La parcela.

l) El apero con el que se va a realizar la tarea.

m) El usuario *Agricultor* encargado de la tarea si esta ha sido delegada.

RI-5 De los tipos de labores se guardará:

a) Un identificador.

b) Nombre.

c) Descripción.

d) Si requiere algún producto adicional para que pueda ser realizada.

e) Si una vez que se ha completado se recolecta algún producto.

RI-6 De las cooperativas se guardarán:

a) Nombre de la cooperativa.

b) Dirección de la sede de la cooperativa.

c) Número de teléfono.

d) Correo electrónico.

e) Contraseña.

f) Nombre completo del responsable de la cooperativa.

g) Si la cuenta ha sido bloqueada.

h) Los usuarios de tipo *Agricultor* que sean socios.

RI-7 De los productos de una cooperativa se guardarán los siguientes campos:

a) Un identificador del producto.

b) Un nombre del producto.

c) Una descripción del producto.

d) El historial de precios del producto.

e) La fecha de la ultima modificación.

f) ELa cooperativa que oferta el producto.

3.2. Casos de uso

3.2.1. Diagrama de Casos de Uso

Para mejorar la legibilidad del diagrama, se ha separado el diagrama que abarcaba todos los casos de uso en subdiagramas teniendo en cuenta los actores que intervienen en el sistema.

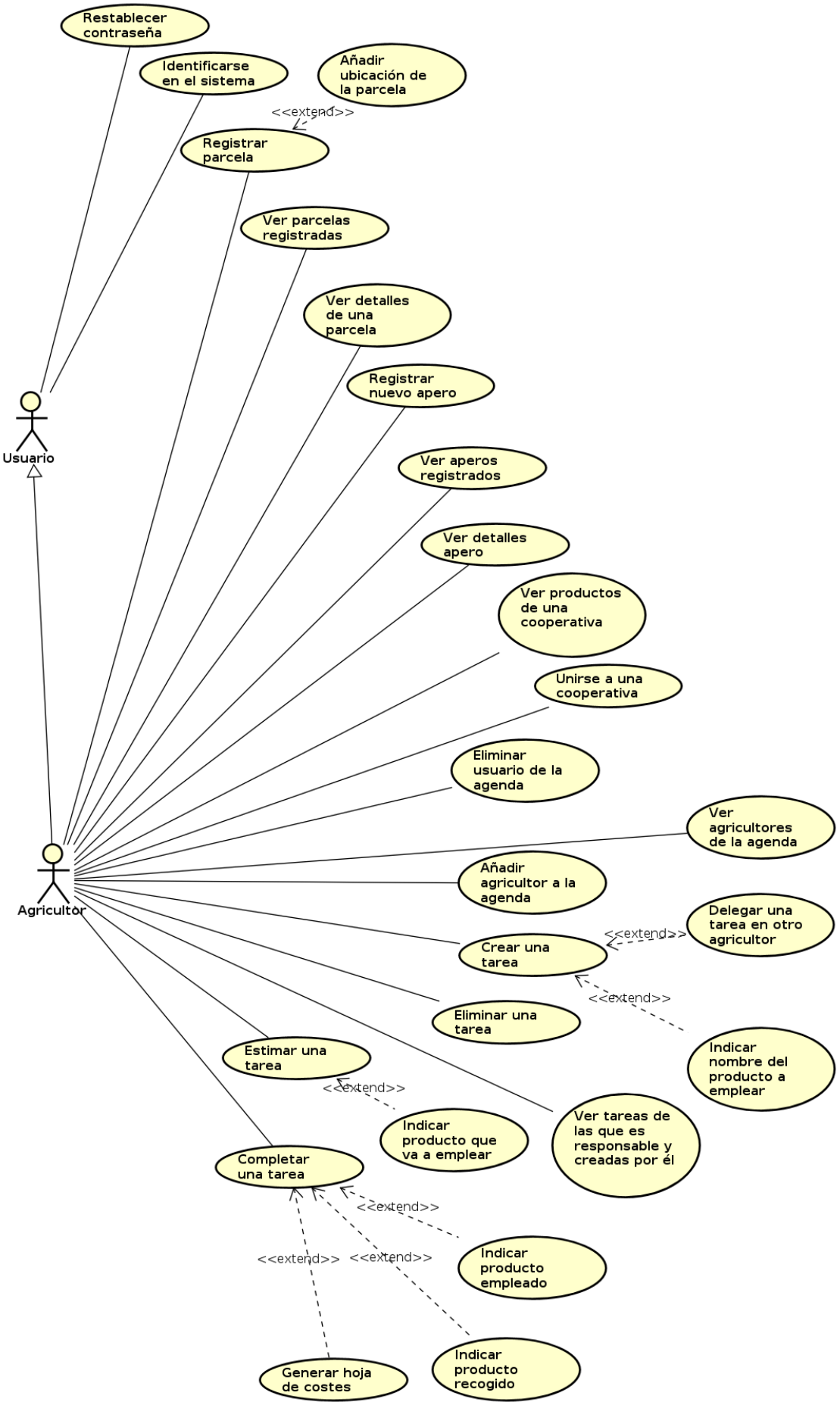

Figura 3.1: Casos de uso del actor Agricultor.

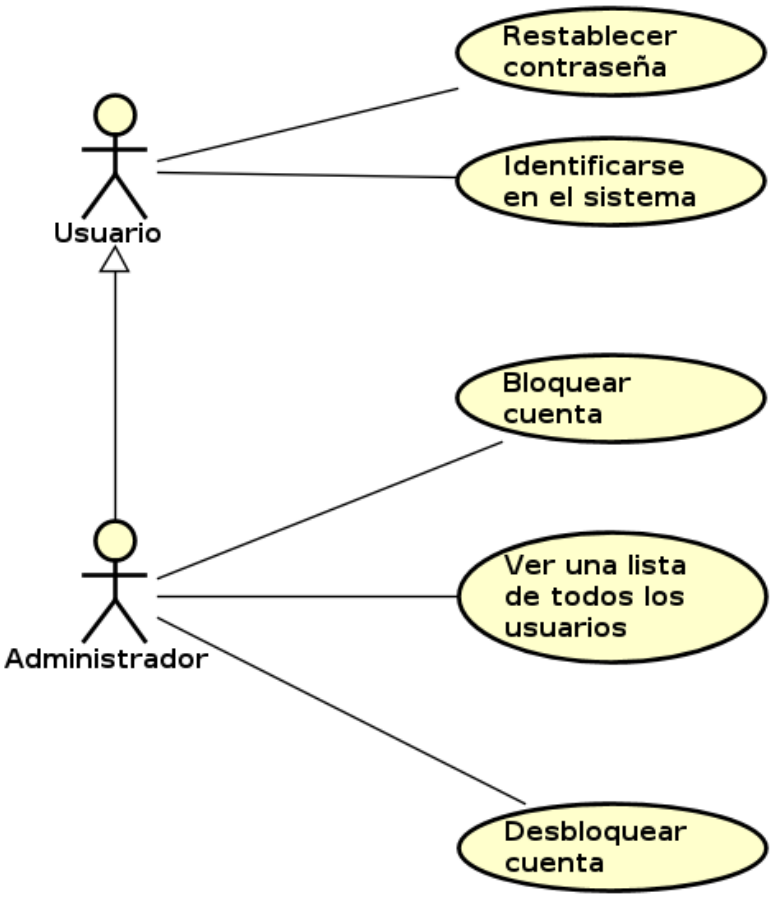

Figura 3.2: Casos de uso del actor Administrador.

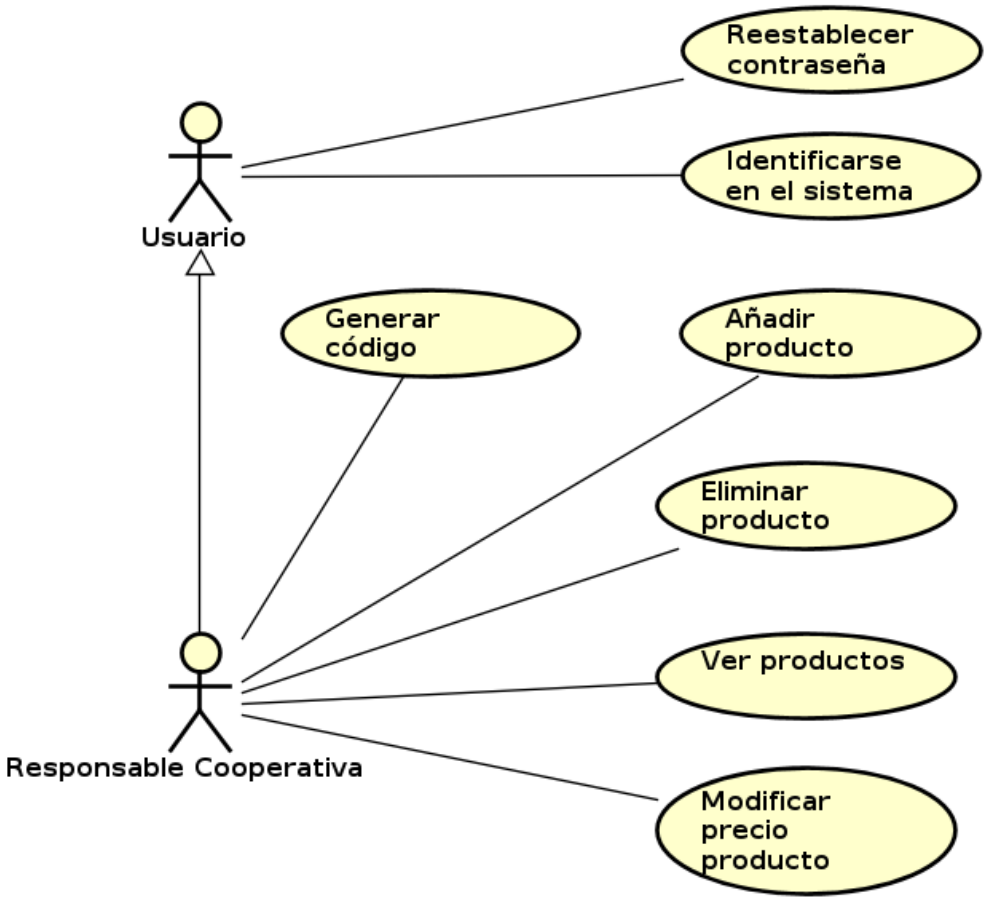

Figura 3.3: Casos de uso del actor Responsable Cooperativa.

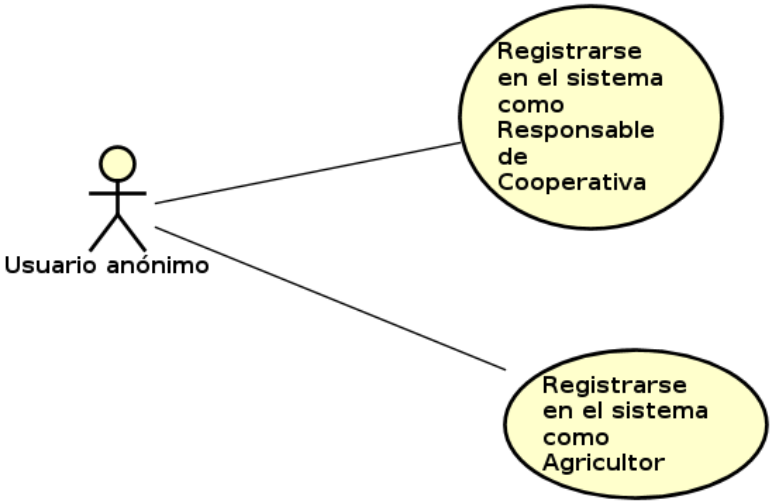

Figura 3.4: Casos de uso del actor Usuario anónimo.

3.2.2. Descripción de los Casos de Uso

Título	Registrarse en el sistema como Agricultor.
Resumen	El actor Usuario anónimo se da de alta en el sistema.
Precondición	Ninguna.
Postcondición	El actor Agricultor se ha dado de alta en el sistema.
Secuencia Principal	1- El actor Agricultor escoge la opción registrarse como Agricultor. 2- El sistema solicita al actor Agricultor que introduzca su nombre, un correo electrónico, una contraseña y el municipio en el que vive y selecciona *Aceptar*. 3- El sistema comprueba que los datos son válidos. 4- El sistema da de alta al usuario Agricultor.
Secuencia Excepcional	2.1- Si el actor Agricultor cancela, el caso de uso queda sin efecto 3.1- Si los datos introducidos no son válidos, el CU vuelve al paso 1

Tabla 3.1: Caso de Uso: *Registrarse en el sistema como Agricultor*

Título	Registrarse en el sistema como Responsable de Cooperativa.
Resumen	El actor Usuario anónimo se da de alta en el sistema.
Precondición	Ninguna.
Postcondición	El actor Responsable de Cooperativa se ha dado de alta en el sistema.
Secuencia Principal	1- El actor Responsable de Cooperativa escoge la opción registrarse como Responsable de Cooperativa. 2- El sistema le solicita el nombre de la cooperativa, la dirección de esta, un correo electrónico, una contraseña y el nombre completo del responsable y selecciona *Aceptar*. 3- El sistema comprueba que los datos son válidos. 4- El sistema registra al actor Responsable de Cooperativa.
Secuencia Excepcional	2.1 Si el actor Responsable de Cooperativa cancela, el caso de uso queda sin efecto. 3.1 Si los datos introducidos no son válidos, el caso de uso vuelve al paso 2.

Tabla 3.2: Caso de Uso: *Registrarse en el sistema como Responsable de Cooperativa.*

Título	Identificarse en el sistema
Resumen	El actor Usuario introduce usuario y contraseña para identificarse en el sistema
Precondición	Ninguna
Postcondición	El actor Usuario está identificado en el sistema
Secuencia Principal	1- El actor Usuario selecciona iniciar sesión 2- El sistema solicita usuario y contraseña. 3- El actor Usuario introduce el correo electrónico y la contraseña y selecciona *Aceptar*. 4- El sistema comprueba que los datos introducidos son correctos.
Secuencia Excepcional	3.1- Si el actor Usuario cancela, el caso de uso queda sin efecto 4.1- Si los datos introducidos no son correctos, el caso de uso vuelve al paso 1

Tabla 3.3: Caso de Uso: *Identificarse en el sistema*

Título	Restablecer contraseña.
Resumen	El actor Usuario cambia la contraseña.
Precondición	Ninguna.
Postcondición	La contraseña del actor Usuario se ha actualizado.
Secuencia Principal	1- El actor Usuario selecciona *Restablece contraseña*. 2 - El sistema le solicita que indique su correo electrónico. 3 - El actor Usuario introduce el correo electrónico y selecciona Aceptar. 4- El sistema comprueba que el correo electrónico está en el sistema. 5- El sistema solicita que introduzca la nueva contraseña. 6- El actor Usuario introduce la nueva contraseña y selecciona Aceptar. 7- El sistema actualiza la contraseña.
Secuencia Excepcional	3.1- El actor Usuario cancela, el caso de uso queda sin efecto. 4.1- El correo electrónico no está en el sistema, el caso de uso vuelve al paso 2. 6.1 - El actor Usuario cancela, el caso de uso queda sin efecto.

Tabla 3.4: Caso de Uso: *Restablece contraseña*

Título	Ver parcelas registradas.
Resumen	El actor Agricultor quiere obtener un listado de las parcelas que ha registrado.
Precondición	El actor Agricultor está identificado en el sistema.
Postcondición	El sistema le muestra al actor Agricultor un listado de sus parcelas.
Secuencia Principal	1- El actor Agricultor selecciona la opción *Parcelas*. 2- El sistema le muestra las parcelas que el actor Agricultor ha registrado.
Secuencia Excepcional	1.1- Si el actor Agricultor escoge la opción *Ver parcelas en el mapa*, el sistema le muestra en un mapa las parcelas en las que se ha indicado su ubicación.

Tabla 3.5: Caso de Uso: *Ver parcelas registradas.*

Título	Registrar parcela.
Resumen	El actor Agricultor añade una nueva parcela en el sistema.
Precondición	El actor Agricultor se encuentra identificado en el sistema.
Postcondición	Se ha registrado una nueva parcela en el sistema.
Secuencia Principal	1- El actor Agricultor escoge la opción *Registrar parcela.* 2- El sistema le solicita un nombre, una descripción, el nombre de la localidad en la que se encuentra y el tamaño en hectáreas. 3- El actor Agricultor introduce los datos solicitados y selecciona *Aceptar.* 4- El sistema registra la parcela.
Secuencia Excepcional	3.1- Si el actor Agricultor cancela, el caso de uso queda sin efecto. 3.2- Si el actor Agricultor escoge la opción indicar ubicación, se ejecuta el caso de uso Añadir ubicación

Tabla 3.6: Caso de Uso: *Registrar parcela.*

Título	Añadir ubicación de la parcela.
Resumen	El actor Agricultor indica donde se encuentra la parcela.
Precondición	En el caso de uso Registrar parcela, el actor Agricultor se encuentra identificado en el sistema.
Postcondición	Se añade la ubicación a una parcela.
Secuencia Principal	1- El sistema solicita al actor Agricultor que indique la ubicación de la parcela. 2- El actor Agricultor indica la ubicación de la parcela. 3- El sistema añade la ubicación a la parcela.
Secuencia Excepcional	2.1- Si el actor Agricultor cancela, el caso de uso queda sin efecto.

Tabla 3.7: Caso de Uso: *Añadir ubicación de la parcela.*

Título	Ver detalles de una parcela.
Resumen	El actor Agricultor consulta los detalles de una parcela.
Precondición	El actor Agricultor está registrado en el sistema.
Postcondición	Ninguna.
Secuencia Principal	1- El actor Agricultor selecciona una parcela de la lista. 2- El sistema muestra los detalles de esa parcela.
Secuencia Excepcional	Ninguna.

Tabla 3.8: Caso de Uso: *Ver detalles de una parcela.*

Título	Ver aperos registrados.
Resumen	El actor Agricultor consulta los aperos que tiene registrados en el sistema.
Precondición	El actor Agricultor está identificado en el sistema.
Postcondición	Ninguna.
Secuencia Principal	1- El actor Agricultor selecciona *Aperos*. 2- El sistema muestra una lista con los aperos que el actor Agricultor tiene registrados.
Secuencia Excepcional	Ninguna.

Tabla 3.9: Caso de Uso: *Ver aperos registrados.*

Título	Registrar nuevo apero.
Resumen	El actor Agricultor registra un nuevo apero en el sistema.
Precondición	El actor Agricultor está identificado en el sistema.
Postcondición	Se añade un nuevo apero al sistema.
	1- El actor Agricultor selecciona *Nuevo apero*. 2- El sistema solicita una marca del apero, el modelo, la matrícula si la tuviese, una descripción, el coste por hectárea trabajada y los tipos de labores que puede desempeñar. 3- El actor Agricultor introduce los datos y selecciona aceptar. 4- El sistema añade el nuevo apero.
Secuencia Excepcional	3.1- Si el actor Agricultor selecciona *Cancelar*, el caso de uso queda sin efecto.

Tabla 3.10: Caso de Uso: *Registrar nuevo apero.*

Título	Ver detalles apero.
Resumen	El actor Agricultor consulta los datos de uno de sus aperos.
Precondición	El actor Agricultor se encuentra identificado en el sistema.
Postcondición	Ninguna.
Secuencia Principal	1- El actor Agricultor selecciona uno de los aperos. 2- El sistema muestra los detalles del apero.
Secuencia Excepcional	Ninguna.

Tabla 3.11: Caso de Uso: *Ver detalles apero.*

Título	Unirse a una cooperativa.
Resumen	El actor Agricultor se une a una cooperativa.
Precondición	El actor Agricultor está identificado en el sistema.
Postcondición	Se añade el actor Agricultor a la cooperativa.
Secuencia Principal	1- El actor Agricultor selecciona *Cooperativa*. 2- El sistema solicita que se introduzca un código. 3- El actor Agricultor introduce el identificador y selecciona *Aceptar*. 4- El sistema comprueba que el código es válido. 5- El sistema añade el actor Agricultor a la cooperativa.
Secuencia Excepcional	3.1- Si el actor Agricultor cancela, el caso de uso queda sin efecto. 4.1- Si el código no es válido, el caso de uso vuelve al paso 2.

Tabla 3.12: Caso de Uso: *Unirse a una cooperativa.*

Título	Ver productos de la cooperativa.
Resumen	El actor Agricultor consulta los precios de la cooperativa.
Precondición	1- El actor Agricultor está identificado en el sistema. 2- El actor Agricultor es miembro de alguna cooperativa.
Postcondición	Ninguna.
Secuencia Principal	1- El actor Agricultor selecciona *Cooperativa*. 2- El sistema muestra los productos de la cooperativa.
Secuencia Excepcional	Ninguna.

Tabla 3.13: Caso de Uso: *Ver productos de la cooperativa.*

Título	Ver agricultores de la agenda.
Resumen	El actor Agricultor consulta los agricultores que tiene registrados en su agenda.
Precondición	El actor Agricultor está identificado en el sistema.
Postcondición	Ninguna.
Secuencia Principal	1- El actor Agricultor selecciona *Agenda*. 2- El sistema muestra los agricultores de la agenda.
Secuencia Excepcional	Ninguna

Tabla 3.14: Caso de Uso: *Ver agricultores de la agenda.*

Título	Añadir agricultor a la agenda.
Resumen	El actor Agricultor añade un nuevo agricultor a su agenda.
Precondición	El actor Agricultor está identificado en el sistema.
Postcondición	Se ha añadido un nuevo agricultor a la agenda.
Secuencia Principal	1- El actor Agricultor selecciona *Añadir nuevo agricultor*. 2- El sistema solicita que introduzca el nombre del agricultor. 3- El actor Agricultor introduce el nombre del agricultor que quiere añadir y selecciona *Aceptar*. 4- El sistema añade el agricultor a la agenda.
Secuencia Excepcional	3.1- Si el actor Agricultor cancela, el caso de uso queda sin efecto. 3.2- Si no hay ningún agricultor registrado en el sistema con ese nombre, el caso de uso queda sin efecto.

Tabla 3.15: Caso de Uso: *Añadir agricultor a la agenda.*

Título	Eliminar usuario de la agenda.
Resumen	El actor Agricultor quiere eliminar un usuario de la agenda.
Precondición	El actor Agricultor está identificado en el sistema.
Postcondición	Se ha eliminado el actor Agricultor de la agenda.
Secuencia Principal	1- El actor Agricultor selecciona los usuarios que quiere eliminar y selecciona *Eliminar*. 2- El sistema elimina los usuarios de la agenda.
Secuencia Excepcional	Ninguna.

Tabla 3.16: Caso de Uso: *Eliminar agricultor de la agenda.*

Título	Ver tareas de las que es responsable y creadas por él.
Resumen	El actor Agricultor quiere consultar sus tareas.
Precondición	El actor Agricultor está identificado en el sistema.
Postcondición	Ninguna.
Secuencia Principal	1- El actor Agricultor selecciona *Tareas*. 2- El sistema le muestra las tareas de las que es responsable y creadas por él agrupadas por el estado.
Secuencia Excepcional	Ninguna.

Tabla 3.17: Caso de Uso: *Ver tareas de las que es responsables y creadas por él.*

Título	Eliminar una tarea.
Resumen	El actor Agricultor quiere eliminar una de sus tareas. Para que pueda ser eliminada, tiene que estar en estado pendiente de estimación o estimada y en el segundo caso que la fecha de inicio sea al menos dos días posterior a la fecha actual.
Precondición	1- El actor Agricultor está identificado en el sistema. 2- El actor Agricultor ha creado esa tarea.
Postcondición	Se ha eliminado la tarea.
Secuencia Principal	1- El actor Agricultor selecciona una tarea y selecciona *Eliminar*. 2- El sistema comprueba que se puede eliminar la tarea. 3- El sistema elimina la tarea.
Secuencia Excepcional	2.1 Si la tarea no puede ser eliminada, el sistema muestra un mensaje y el caso de uso vuelve al paso 1.

Tabla 3.18: Caso de Uso: *Eliminar una tarea.*

Título	Crear una tarea.
Resumen	El actor Agricultor crea una tarea.
Precondición	El actor Agricultor está identificado en el sistema.
Postcondición	Se ha creado una tarea.
Secuencia Principal	1- El actor Agricultor selecciona *Crear tarea*. 2- El sistema solicita al actor Agricultor que introduzca la parcela donde la quiere realizar y el tipo de labor. 3- El actor Agricultor introduce los datos y selecciona aceptar. 4- El sistema crea la tarea con estado pendiente de estimación.
Secuencia Excepcional	3.1- Si el actor Agricultor selecciona cancelar, el caso de uso queda sin efecto. 3.2- Si el actor Agricultor quiere delegar la tarea, se ejecuta el caso de uso Delegar una tarea en otro agricultor. 3.3 - Si la tarea seleccionada requiere algún producto adicional, se ejecuta el caso de uso Indicar nombre del producto a emplear.

Tabla 3.19: Caso de Uso: *Crear una tarea.*

Título	Delegar una tarea en otro agricultor.
Resumen	El actor Agricultor selecciona delegar una tarea en otro agricultor.
Precondición	En el caso de uso Crear una tarea, el actor Agricultor está identificado en el sistema.
Postcondición	Se asigna la tarea al agricultor en el que se ha delegado la tarea.
Secuencia Principal	1- El actor Agricultor selecciona *Delegar*. 2- El sistema solicita el nombre del agricultor. 3- El actor Agricultor introduce el nombre del agricultor y selecciona *Aceptar*. 4- El sistema asigna como responsable de la tarea a ese agricultor.
Secuencia Excepcional	3.1 Si el actor Agricultor selecciona cancelar, el caso de uso queda sin efecto.

Tabla 3.20: Caso de Uso: *Delegar una tarea en otro agricultor.*

Título	Indicar nombre del producto a emplear.
Resumen	El actor Agricultor indica el producto que quiere emplear.
Precondición	1- En el caso de uso Crear una tarea, el actor Agricultor está identificado en el sistema. 2- En el caso de uso Crear una tarea, la tarea requiere que se indique un producto.
Postcondición	Se ha guardado el producto que se quiere emplear.
Secuencia Principal	1- El actor Agricultor selecciona añadir producto. 2- El sistema le solicita el nombre del producto que quiere emplear. 3- El actor Agricultor indica el nombre del producto que quiere emplear y selecciona *Aceptar*. 4- El sistema añade el producto a emplear a la tarea.
Secuencia Excepcional	3.1- Si el actor Agricultor cancela, el caso de uso queda sin efecto.

Tabla 3.21: Caso de Uso: *Indicar nombre del producto a emplear.*

Título	Estimar una tarea.
Resumen	El actor Agricultor estima una tarea en estado pendiente de estimación.
Precondición	1- El actor Agricultor está identificado en el sistema.
Postcondición	Se cambia el estado de la tarea a estimado.
Secuencia Principal	1- El actor Agricultor selecciona una tarea con estado pendiente de estimación. 2- El sistema solicita al actor Agricultor que introduzca el día que pretende realizar la tarea, el tiempo que estima que va a necesitar y el apero que va a emplear. 3- El actor Agricultor introduce los datos y selecciona aceptar. 4- El sistema guarda los datos y cambia el estado de la tarea a estimada.
Secuencia Excepcional	3.1- Si el actor Agricultor cancela, el caso de uso queda sin efecto. 3.2- Si la tarea que va a realizar necesita algún producto a mayores, se ejecuta el caso de uso Indicar el producto que va a emplear.

Tabla 3.22: Caso de Uso: *Estimar una tarea.*

Título	Indicar producto que va a emplear.
Resumen	El actor Agricultor indica el producto que va a emplear en una tarea con estado pendiente de estimación.
Precondición	1- El actor Agricultor está identificado en el sistema. 2- En el caso de uso Estimar una tarea, la tarea requiere que se indique un producto. 3- La tarea está en pendiente de estimación.
Postcondición	Se ha añadido un producto y la cantidad a la tarea.
Secuencia Principal	1- El actor Agricultor selecciona *Estimar producto* 2- El sistema solicita que se indique la cantidad por hectárea que estima que va a necesitar y el precio de este. 3- El actor Agricultor introduce los valores y selecciona *Aceptar*. 4- El sistema registra la solicitud.
Secuencia Excepcional	3.1- Si el actor Agricultor selecciona cancelar, el caso de uso queda sin efecto.

Tabla 3.23: Caso de Uso: *Indicar producto que va a emplear.*

Título	Completar una tarea.
Resumen	El actor Agricultor completa los datos de una tarea que ya ha realizado.
Precondición	1- El actor Agricultor se encuentra identificado en el sistema. 2- La tarea se encuentra en estado estimada.
Postcondición	Se cambia el estado de la tarea a completada.
Secuencia Principal	1- El actor Agricultor selecciona una tarea con estado estimada. 2- El sistema solicita al actor Agricultor que introduzca los días y las horas que ha trabajado en ella. 3- El actor Agricultor introduce los datos y selecciona *Aceptar*. 4- El sistema registra las horas que se han dedicado a la tarea.
Secuencia Excepcional	3.1- Si el actor Agricultor cancela, el caso de uso queda sin efecto. 3.2- Si la tarea que se realizó ha requerido algún producto a mayores, se ejecuta el caso de uso Indicar producto empleado. 3.3- Si cuando se ha terminado la labor se ha recogido algún producto, se ejecuta el caso de uso Indicar producto recogido. 3.4- Si el actor Agricultor selecciona generar hoja de costes, el sistema genera un archivo indicando el coste del apero y los costes de los productos que se hayan empleado.

Tabla 3.24: Caso de Uso: *Completar una tarea.*

Título	Indicar producto empleado.
Resumen	El actor Agricultor indica el producto que ha empleado para realizar tarea.
Precondición	1- En el caso de uso Completar una tarea, el actor Agricultor está identificado en el sistema. 2-En el caso de uso Completar una tarea, la tarea está en estado estimada. 3- En el caso de uso Completar una tarea, la tarea necesita algún tipo de producto para ser realizada.
Postcondición	Se ha registrado el producto empleado.
Secuencia Principal	1- El actor Agricultor selecciona *Indicar producto empleado.* 2- El sistema solicita que indique la cantidad de producto empleado por hectárea. 3- El actor Agricultor introduce la cantidad por hectárea de producto que ha empleado y selecciona *Aceptar*. 4- El sistema registra la cantidad de producto empleada.
Secuencia Excepcional	3.1 Si el actor Agricultor cancela, el caso de uso queda sin efecto.

Tabla 3.25: Caso de Uso: *Indicar producto empleado.*

Título	Indicar producto recogido.
Resumen	El actor Agricultor indica el producto que ha recogido después de haber realizado una tarea.
Precondición	1- En el caso de uso Completar una tarea, el actor Agricultor está identificado en el sistema. 2-En el caso de uso Completar una tarea, la tarea está en estado estimada. 3- En el caso de uso Completar una tarea, la tarea genera algún tipo de producto una vez que ha sido realizada.
Postcondición	Se ha registrado el producto recogido en la tarea.
Secuencia Principal	1- El actor Agricultor selecciona *Indicar producto recogido.* 2- El sistema solicita que indique el nombre y la cantidad de producto empleado por hectárea. 3- El actor Agricultor introduce el nombre y la cantidad por hectárea de producto que ha empleado y selecciona *Aceptar.* 4- El sistema registra la cantidad de producto recogida.
Secuencia Excepcional	3.1 Si el actor Agricultor cancela, el caso de uso queda sin efecto.

Tabla 3.26: Caso de Uso: *Indicar producto recogido.*

Título	Generar código.
Resumen	El actor Responsable de Cooperativa genera un código para que los actores Agricultores puedan unirse a la cooperativa.
Precondición	El actor Administrador está identificado en el sistema.
Postcondición	Ninguna.
Secuencia Principal	1- El actor Agricultor selecciona *Socios* y *Generar código.* 2- El sistema genera un código.
Secuencia Excepcional	Ninguna.

Tabla 3.27: Caso de Uso: *Generar código.*

Título	Ver productos.
Resumen	El actor Responsable de Cooperativa consulta una lista de los productos de la cooperativa.
Precondición	El actor Responsable de Cooperativa está identificado en el sistema.
Postcondición	Ninguna.
Secuencia Principal	1- El actor Responsable de Cooperativa selecciona *Productos.* 2- El sistema le muestra un listado de los productos de la cooperativa.
Secuencia Excepcional	Ninguna.

Tabla 3.28: Caso de Uso: *Ver productos.*

Título	Añadir producto.
Resumen	El actor Responsable de Cooperativa añade un nuevo producto a la cooperativa.
Precondición	El actor Responsable de Cooperativa está identificado en el sistema.
Postcondición	Se ha añadido un nuevo producto a la cooperativa.
Secuencia Principal	1- El actor Responsable de Cooperativa selecciona *Añadir*. 2- El sistema solicita que indique el nombre y el precio del producto. 3- El actor Responsable de Cooperativa indica los datos y selecciona *Aceptar*. 4- El sistema registra el nuevo producto.
Secuencia Excepcional	3.1- Si el actor Responsable de Cooperativa cancela, el caso de uso queda sin efecto.

Tabla 3.29: Caso de Uso: *Añadir producto.*

Título	Eliminar producto.
Resumen	El actor Responsable de Cooperativa elimina un producto de la cooperativa.
Precondición	El actor Responsable de Cooperativa está identificado en el sistema.
Postcondición	Se ha eliminado el producto.
Secuencia Principal	1- El actor Responsable de Cooperativa selecciona el producto que quiere eliminar y selecciona *Eliminar*. 2- El sistema elimina el producto.
Secuencia Excepcional	Ninguna.

Tabla 3.30: Caso de Uso: *Eliminar producto.*

Título	Modificar precio producto.
Resumen	El actor Responsable de Cooperativa modifica el precio un producto de la cooperativa.
Precondición	El actor Responsable de Cooperativa está identificado en el sistema.
Postcondición	Se ha modificado el producto.
Secuencia Principal	1- El actor Responsable de Cooperativa selecciona el producto que quiere modificar. 2- El sistema solicita que introduzca el nuevo precio. 3- El actor Responsable de Cooperativa introduce los datos y selecciona *Aceptar*. 34- El sistema modifica el precio.
Secuencia Excepcional	3.1- Si el actor Responsable de Cooperativa cancela, el caso de uso queda sin efecto.

Tabla 3.31: Caso de Uso: *Modificar precio producto.*

Título	Ver listado usuarios.
Resumen	El actor Administrador obtiene un listado de todos los usuarios del sistema.
Precondición	El actor Administrador está identificado en sistema.
Postcondición	Ninguna.
Secuencia Principal	1- El actor Administrador selecciona *Usuarios*. 2- El sistema le devuelve un listado de todos los usuarios.
Secuencia Excepcional	Ninguna.

Tabla 3.32: Caso de Uso: *Ver listado usuarios.*

Título	Bloquear cuenta.
Resumen	El actor Administrador bloquea una cuenta de un usuario del sistema.
Precondición	El actor Administrador está identificado en el sistema.
Postcondición	La cuenta del usuario ha sido bloqueada.
Secuencia Principal	1- El actor Administrador selecciona *Bloquear* del usuario. 2- El sistema bloquea la cuenta.
Secuencia Excepcional	Ninguna.

Tabla 3.33: Caso de Uso: *Bloquear cuenta.*

Título	Desbloquear cuenta.
Resumen	El actor Administrador desbloquea una cuenta de un usuario del sistema.
Precondición	1- El actor Administrador está identificado en el sistema. 2- La cuenta del usuario está bloqueada.
Postcondición	La cuenta del usuario ha sido desbloqueada.
Secuencia Principal	1- El actor Administrador selecciona *Desbloquear* del usuario. 2- El sistema desbloquea la cuenta.
Secuencia Excepcional	Ninguna.

Tabla 3.34: Caso de Uso: *Desbloquear cuenta.*

3.3. Modelo de dominio

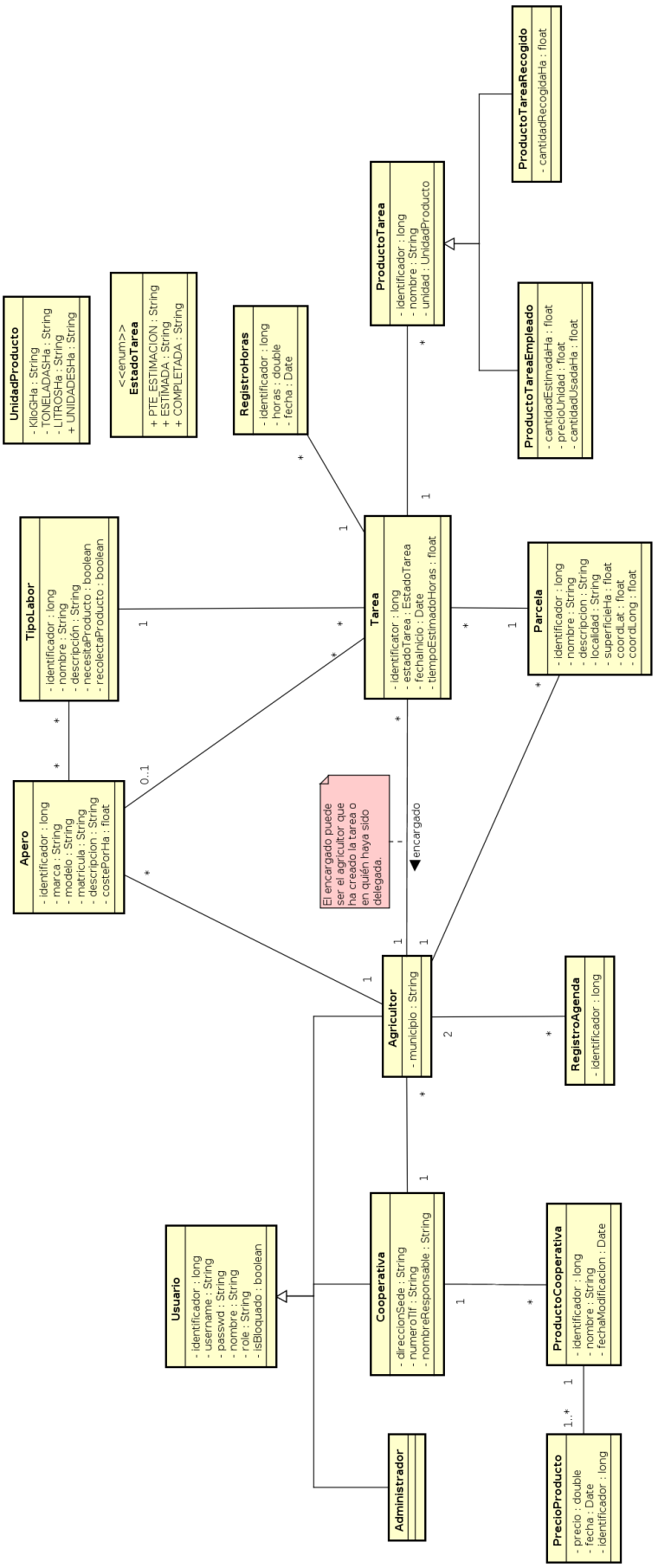

Figura 3.5: Modelo de dominio.

3.4. Diagramas de secuencia en análisis

A continuación, se describen en análisis los mensajes que se necesitan para llevar a cabo los principales casos de uso de la aplicación.

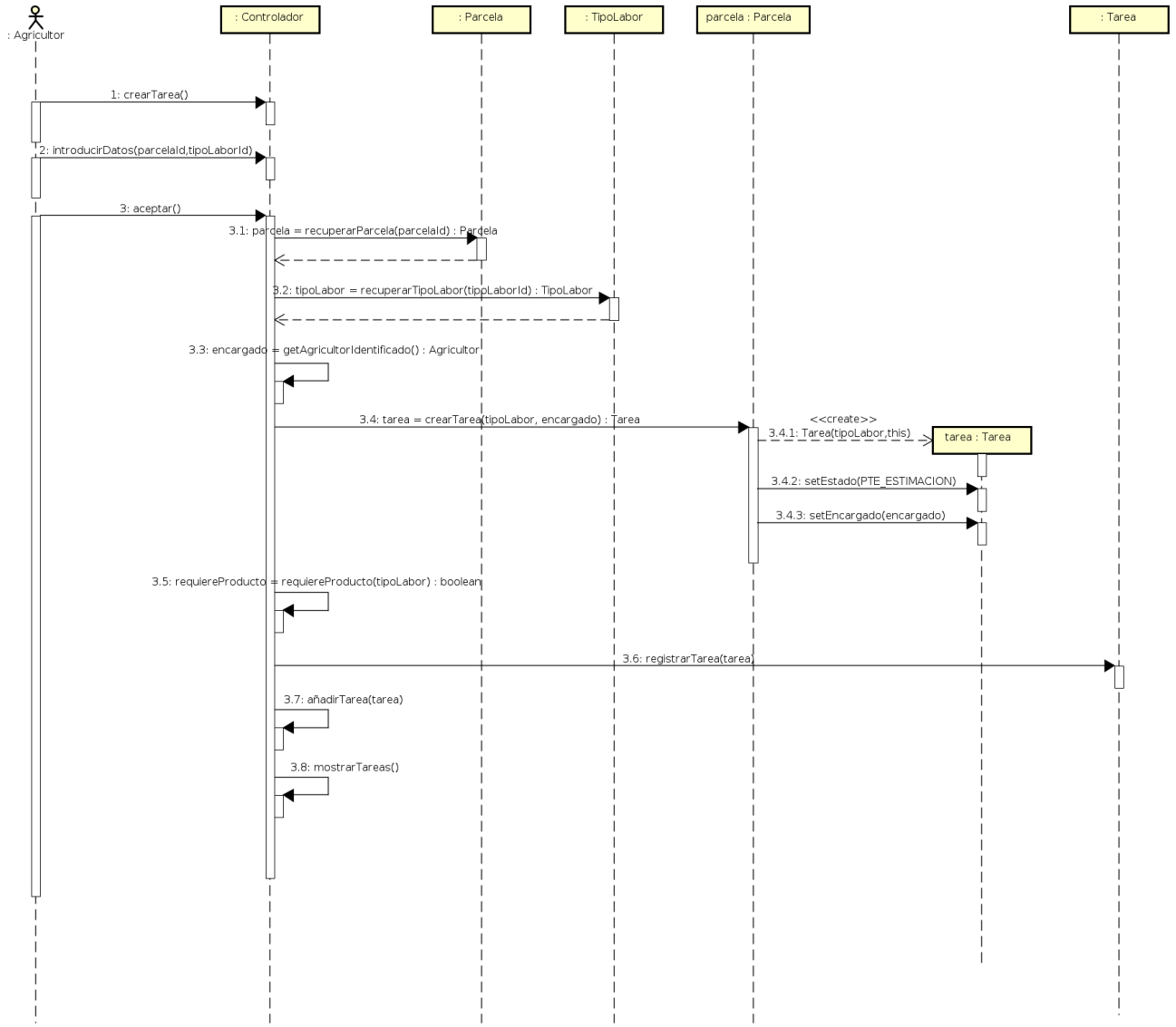

Figura 3.6: Diagrama secuencia análisis: Crear tarea.

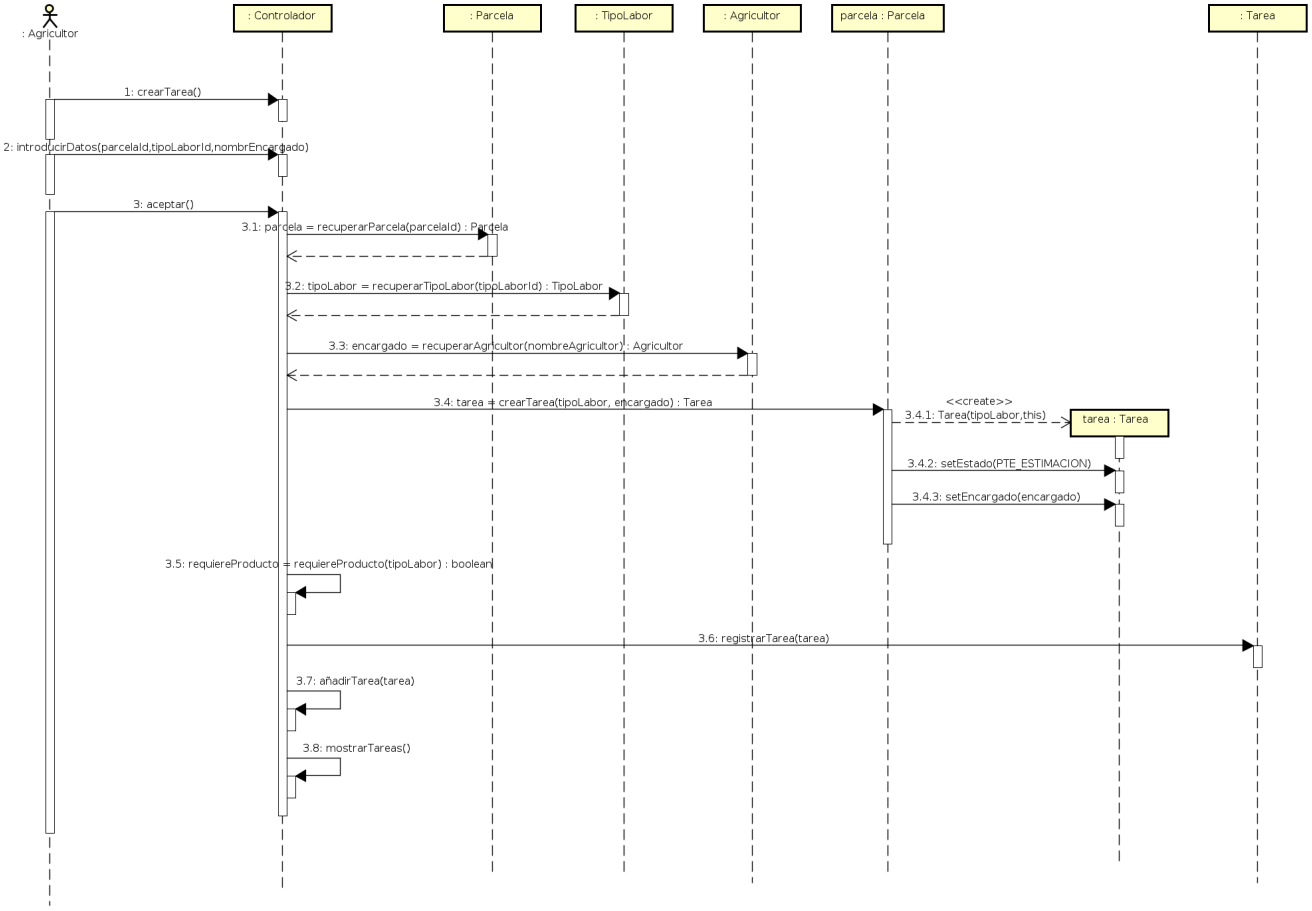

Figura 3.7: Diagrama secuencia análisis: Crear tarea delegada.

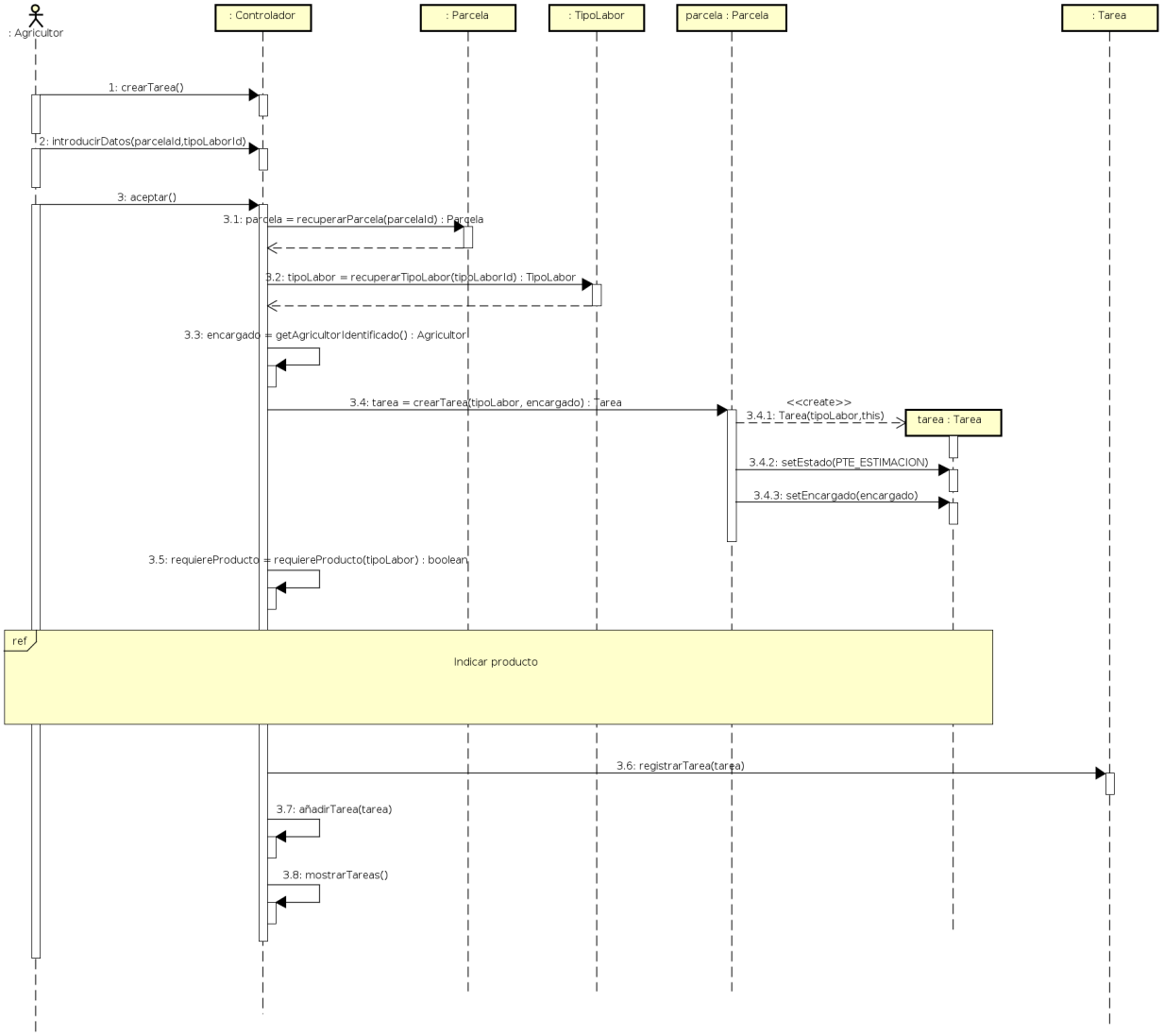

Figura 3.8: Diagrama secuencia análisis: Crear tarea requiere algún producto.

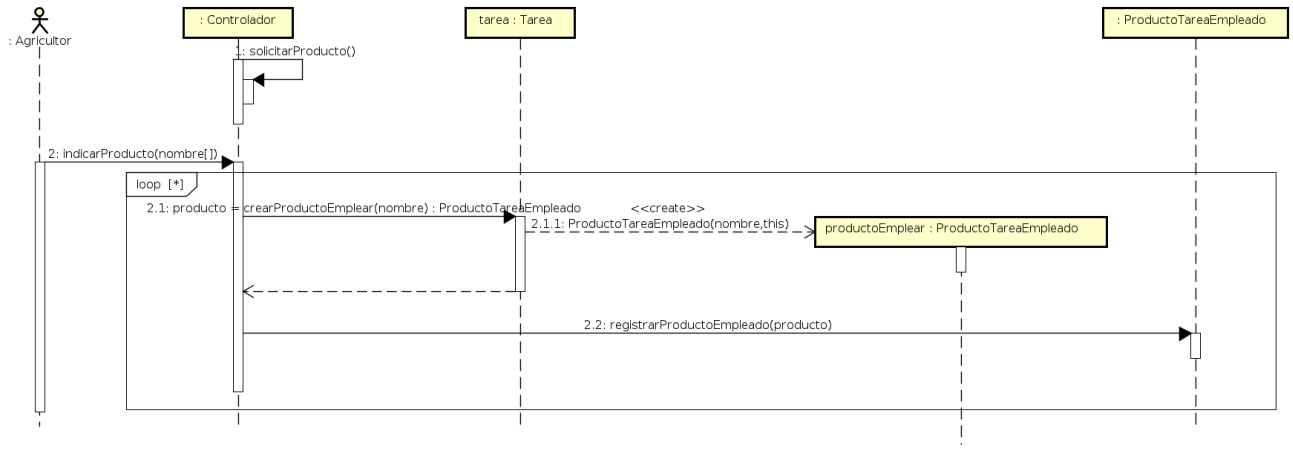

Figura 3.9: Diagrama secuencia análisis: Indicar producto a emplear.

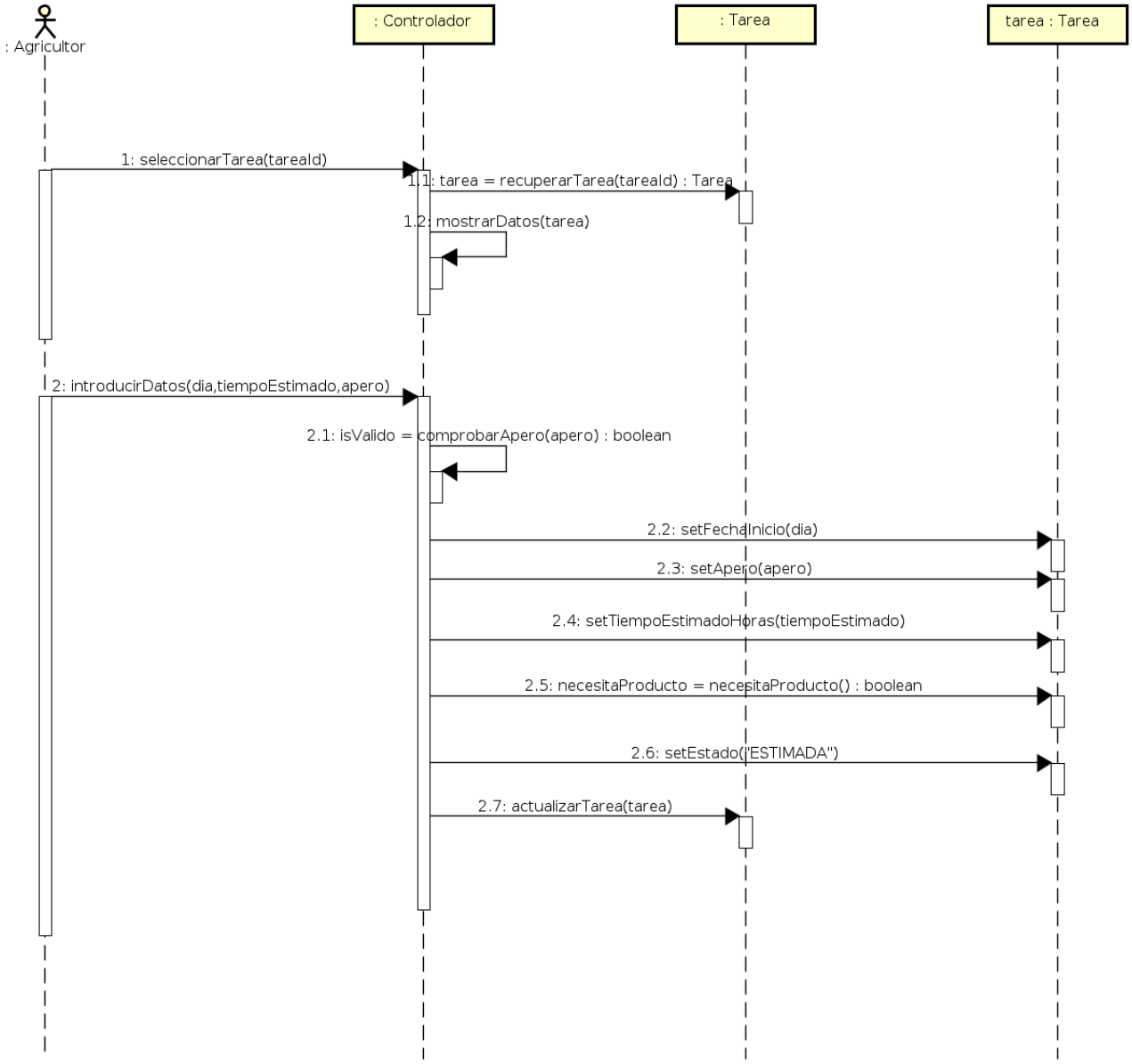

Figura 3.10: Diagrama secuencia análisis: Estimar tarea.

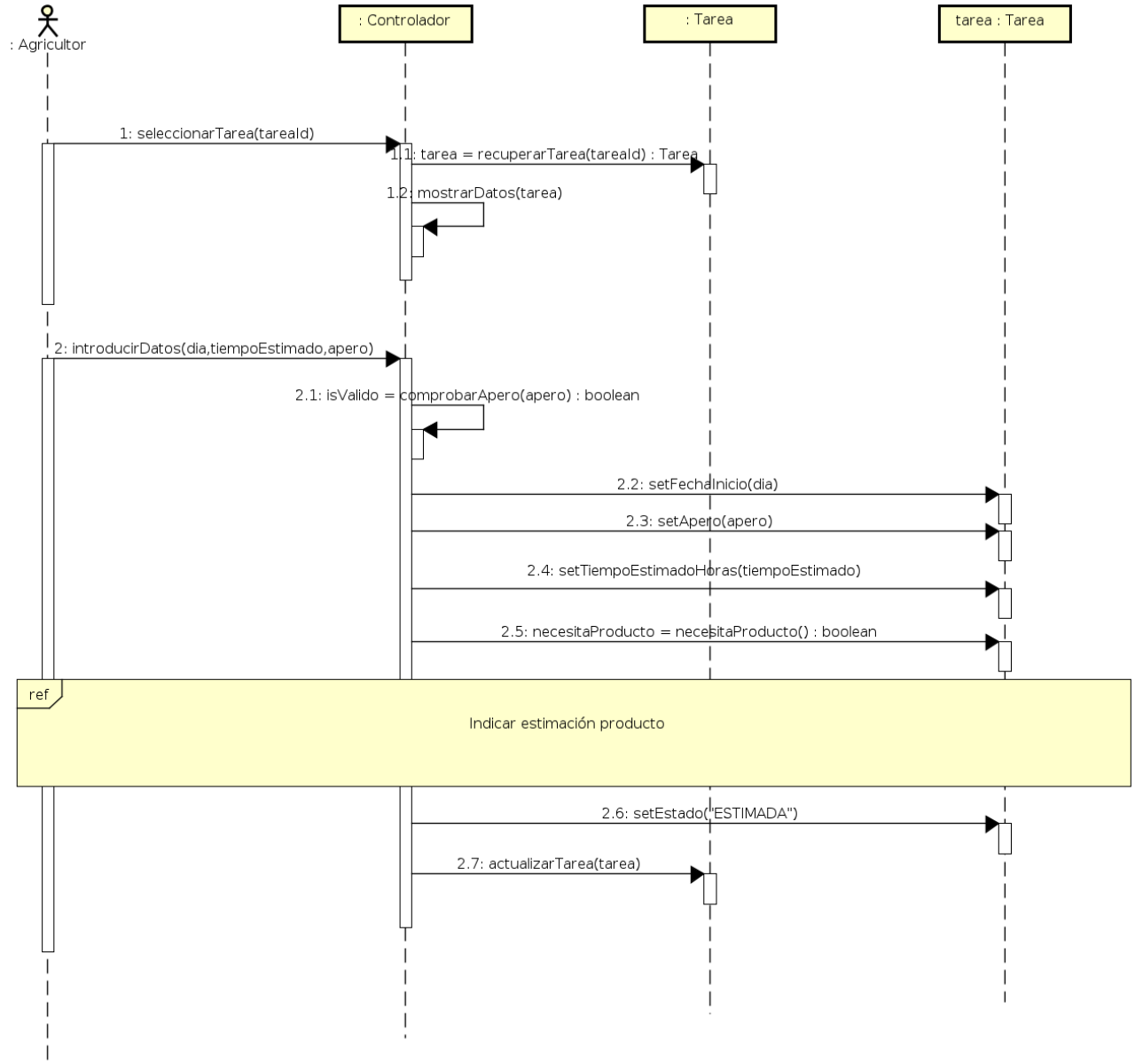

Figura 3.11: Diagrama secuencia análisis: Estimar tarea que requiere algún producto.

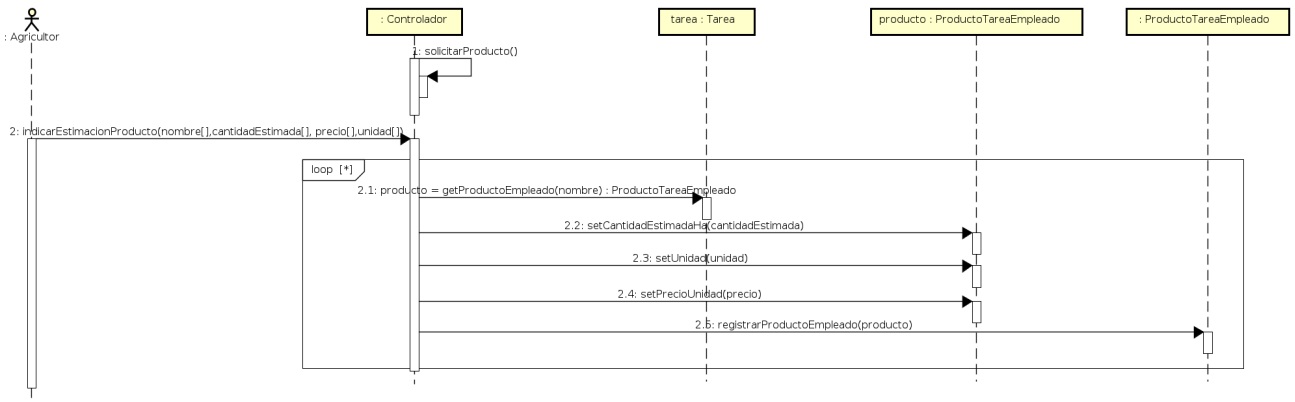

Figura 3.12: Diagrama secuencia análisis: Indicar estimación de producto a emplear.

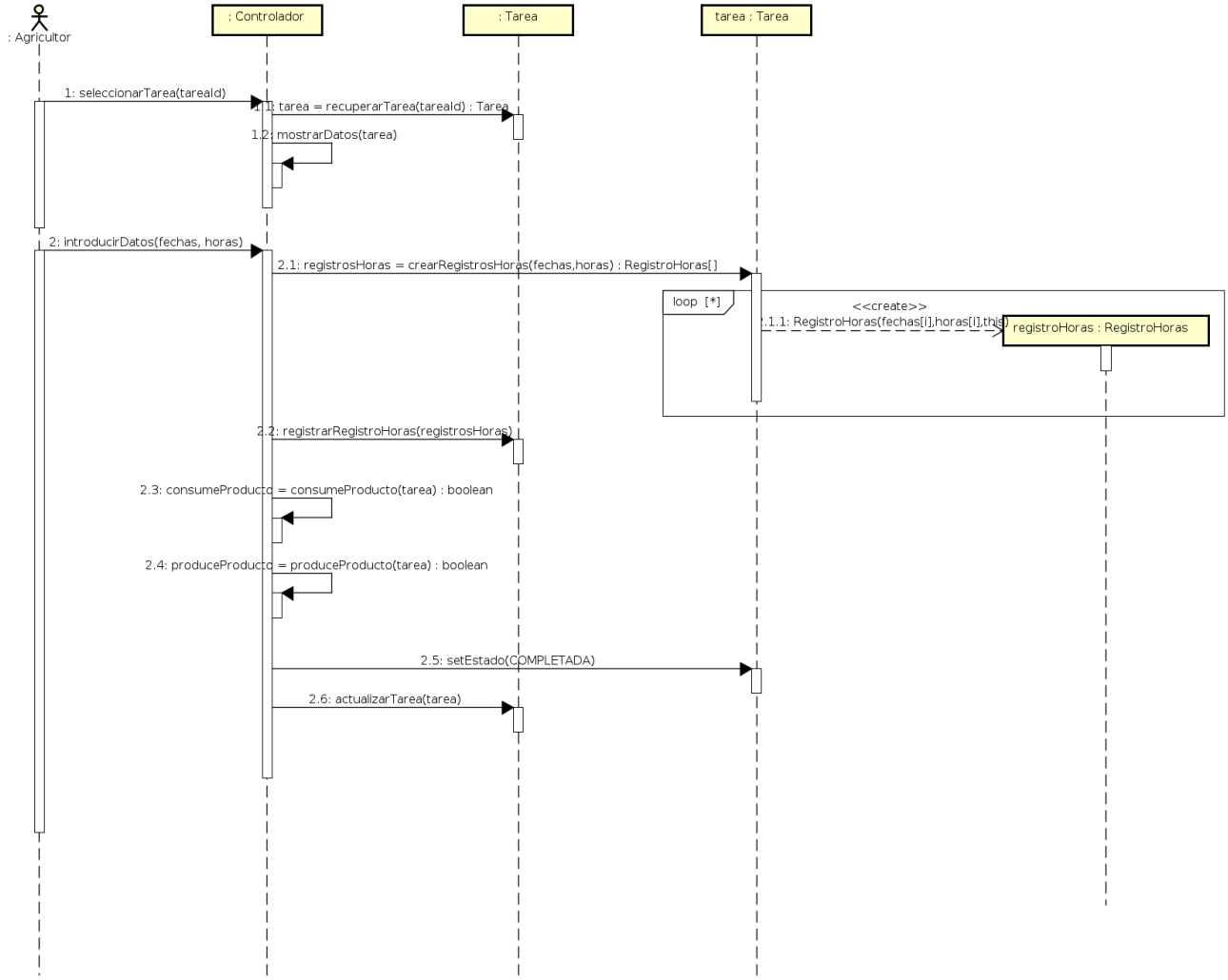

Figura 3.13: Diagrama secuencia análisis: Completar tarea.

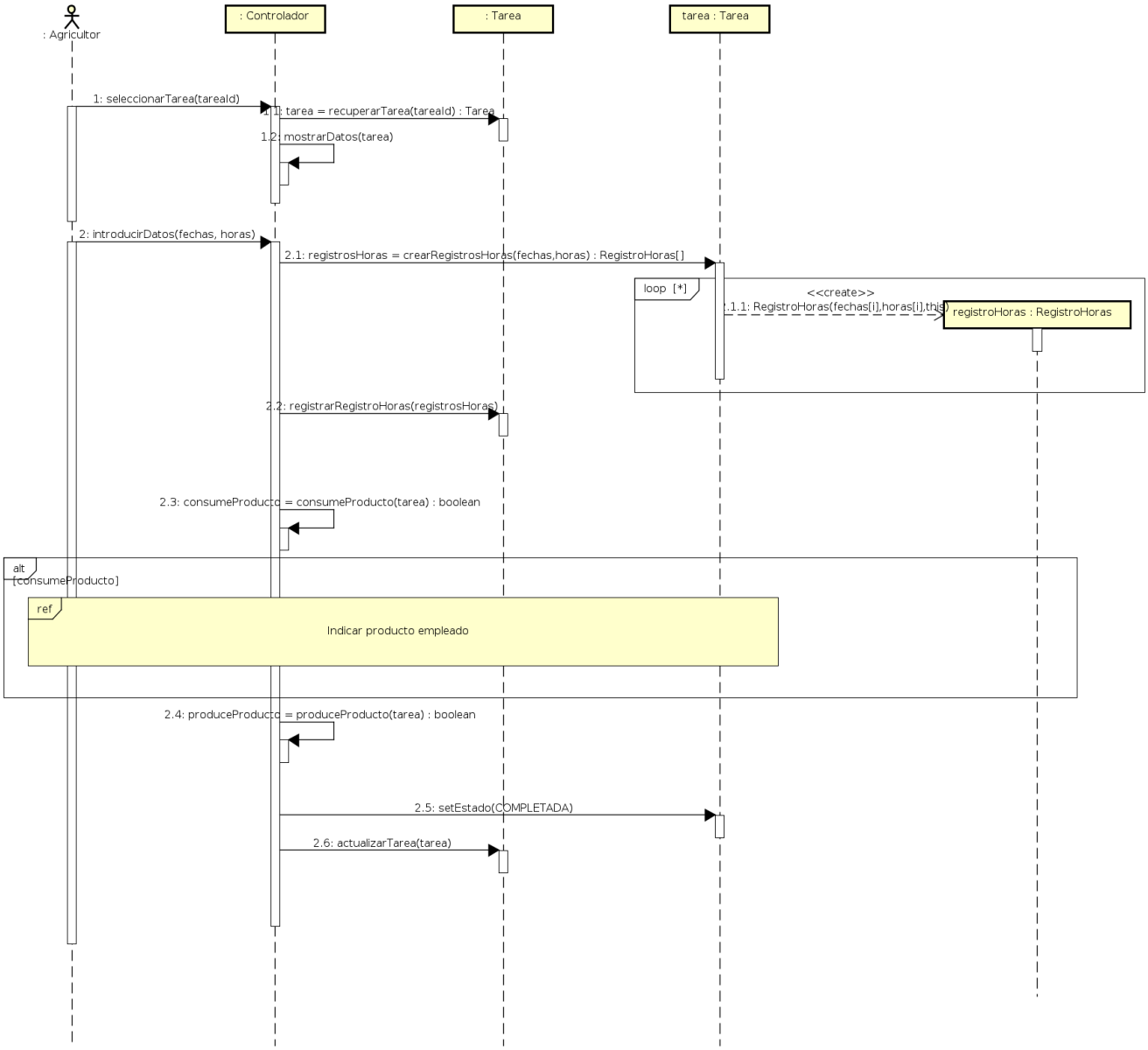

Figura 3.14: Diagrama secuencia análisis: Completar tarea requiere producto.

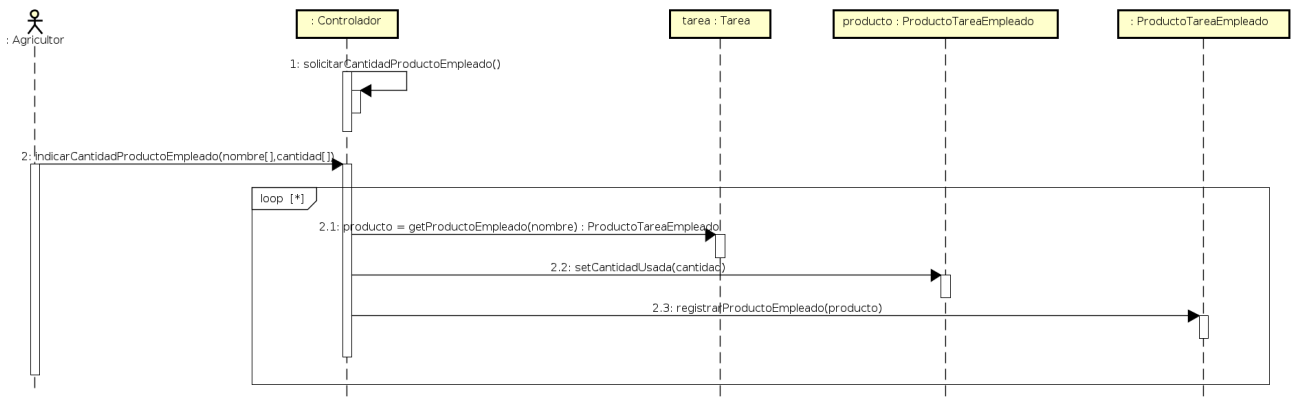

Figura 3.15: Diagrama secuencia análisis: Indicar producto empleado.

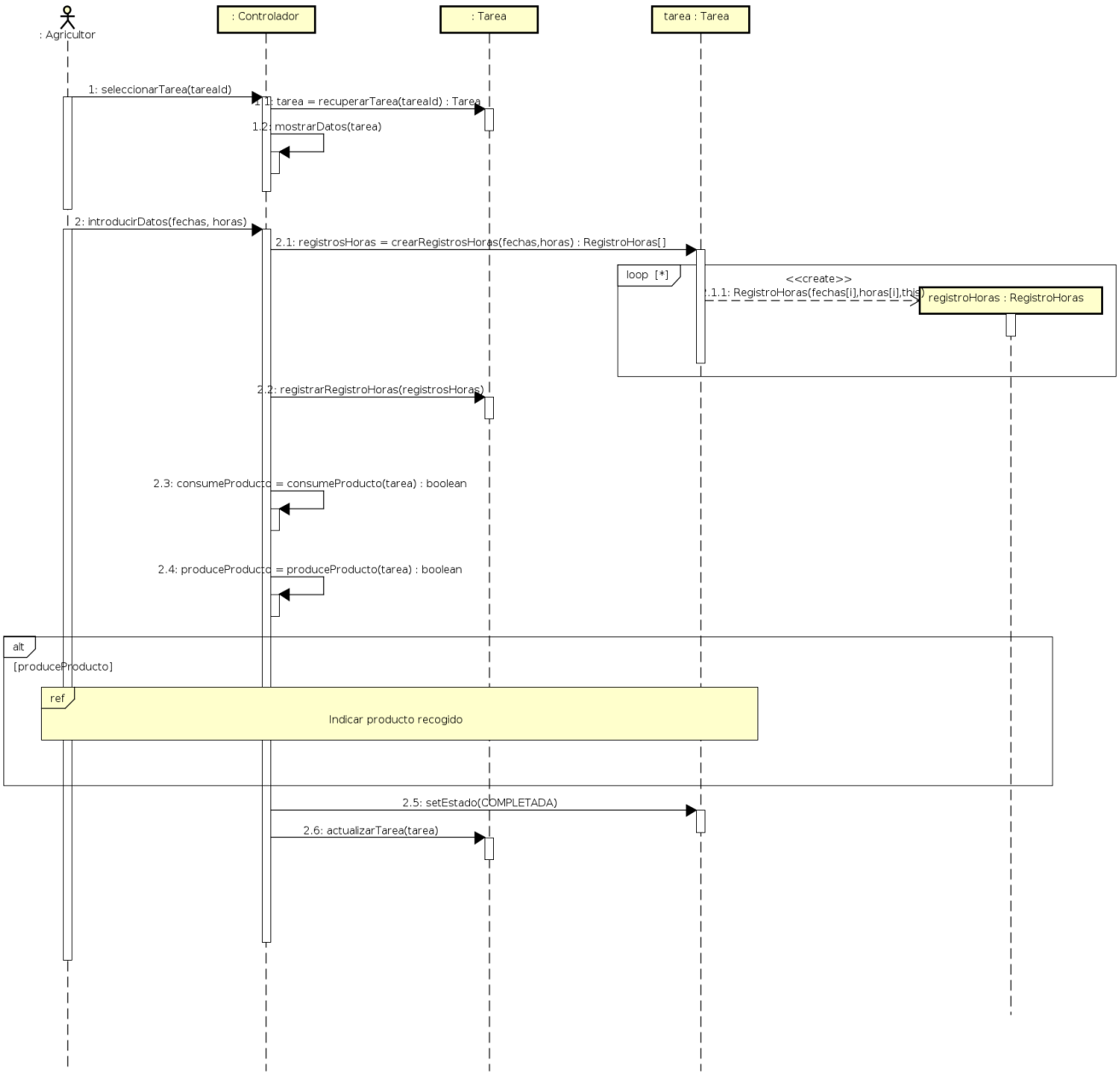

Figura 3.16: Diagrama secuencia análisis: Completar tarea recolecta producto.

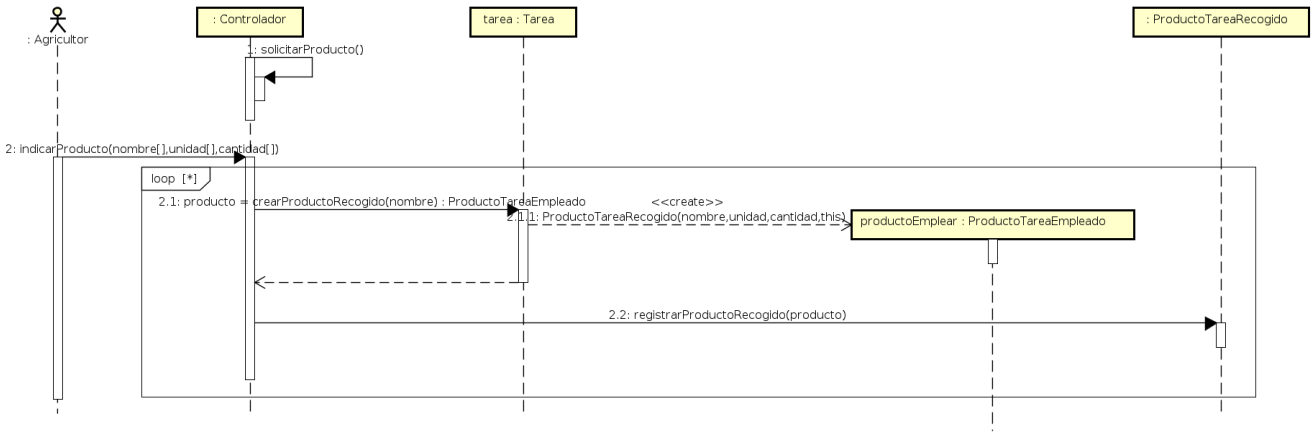

Figura 3.17: Diagrama secuencia análisis: Indicar producto recogido.

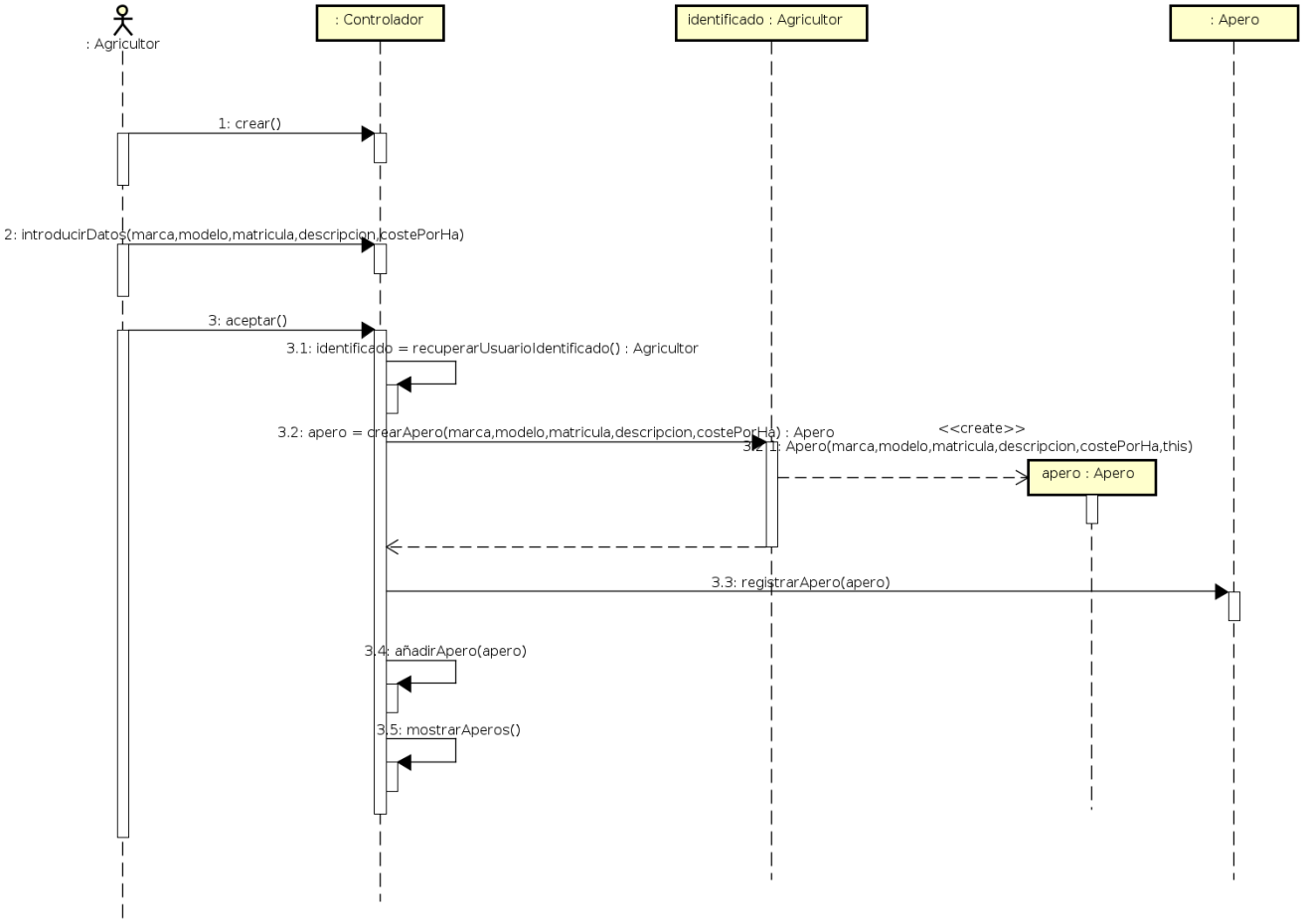

Figura 3.18: Diagrama secuencia análisis: Registrar apero.

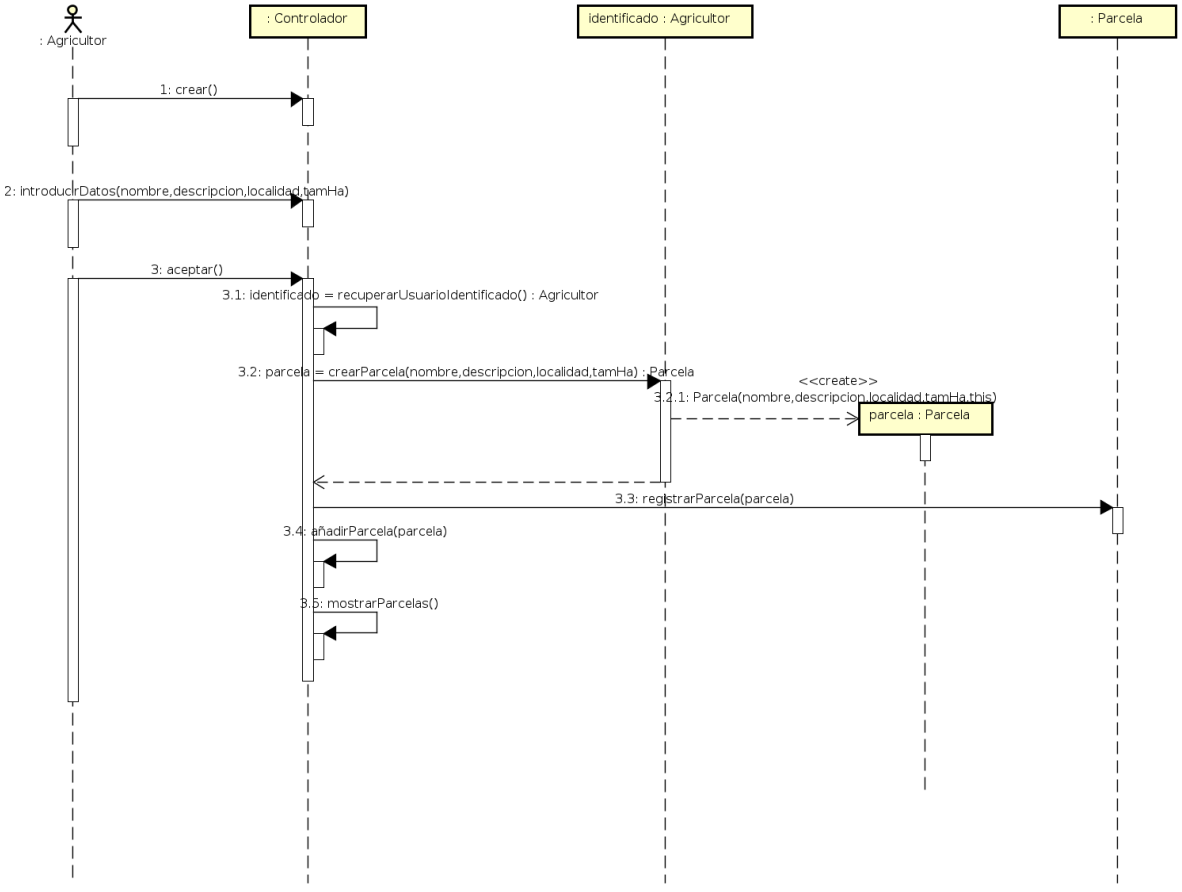

Figura 3.19: Diagrama secuencia análisis: Registrar parcela.

Capítulo 4

Diseño del sistema.

En este capítulo se presenta el diseño del sistema que se ha desarrollado teniendo en cuenta los resultados de la fase de análisis. En primer lugar, se describirán los diagramas de la estructura de la API REST y a continuación el diagrama de donde se representa la estructura de la base de datos. Después, se describirá la estructura de los paquetes de las diferentes aplicaciones que forman el sistema y por último, los diagramas de secuencia en diseño de los principales casos de uso y los prototipos de las vistas de la aplicación.

4.1. Diseño de la API REST.

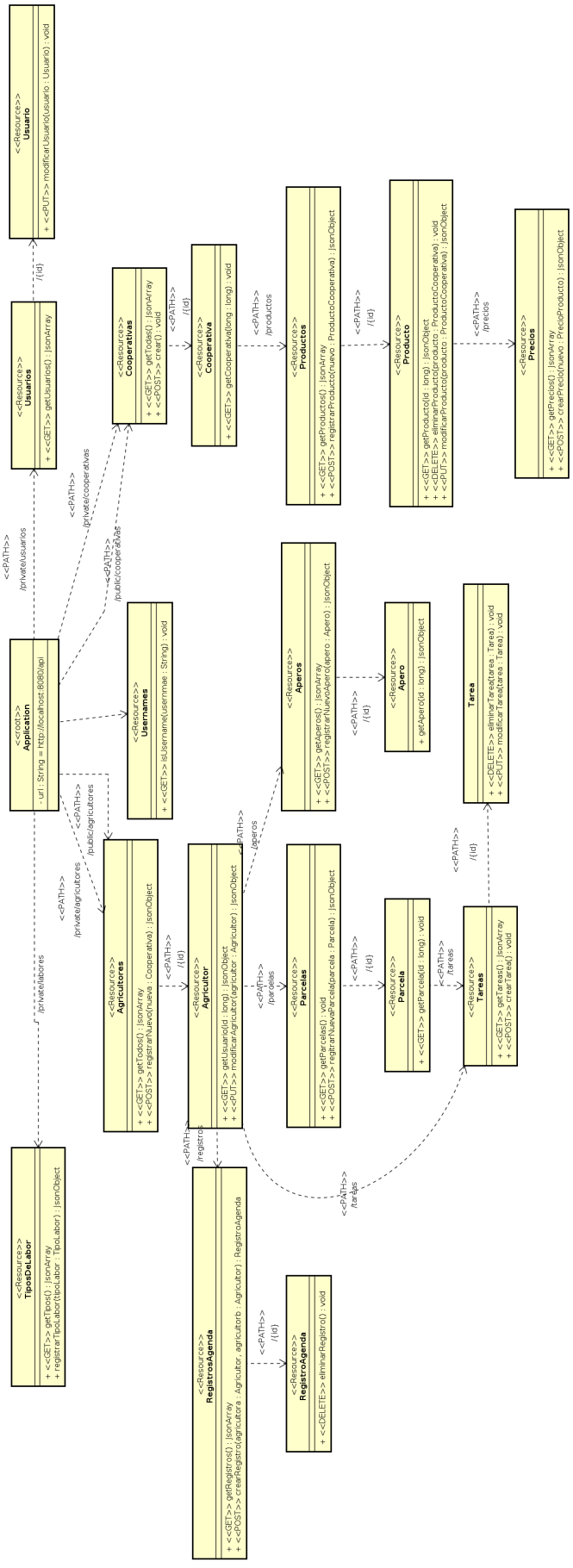

Figura 4.1: Diseño API REST.

4.2. Diseño de base de datos

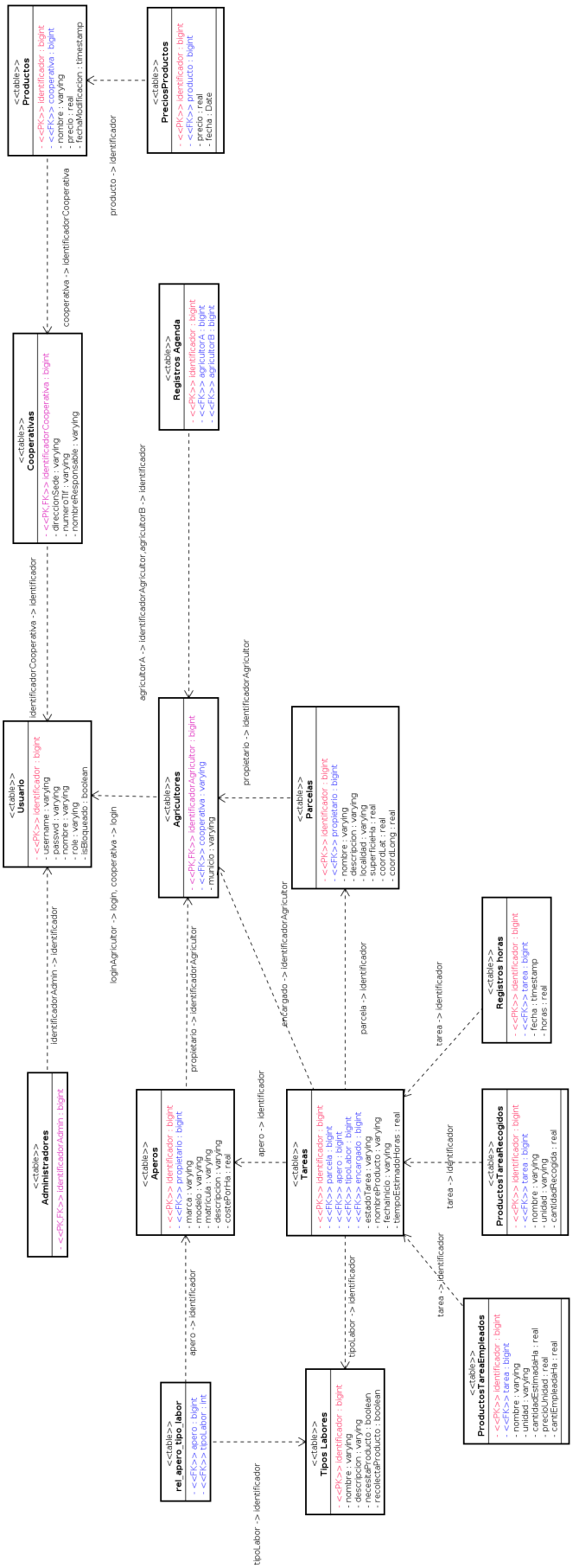

Figura 4.2: Modelo de base de datos.

4.3. Arquitectura general del sistema

Ha sido diseñada basándose en el patrón *cliente-servidor* por lo tanto se han definido dos partes denominadas *front-end* y *back-end*. La parte *front-end* estará formada por una aplicación web orientada a ser utilizada en dispositivos no móviles implementada en *Angular* y una aplicación móvil desarrollada para dispositivos *Android* que permita al agricultor hojear las tareas que debería realizar ese día.Ambas contendrán las vistas que permitirán interactuar a los usuarios con el sistema y los mecanismos para comunicarse con el *back-end*. El *back-end* contendrá el acceso a persistencia, control de acceso a los datos por parte de los usuarios y un servicio que permita la comunicación entre el *front-end* y el *back-end*. Esta parte será implementada en *Spring Boot*. La principal ventaja de este diseño es que permite reducir la carga del servidor porque no tiene que hacerse cargo de las vistas de la aplicación puesto que de esto se encarga el navegador del cliente o el dispositivo móvil en el caso de la aplicación.

4.3.1. *Front-end.*

Android

Para el desarrollo de la aplicación *Android* se pueden tomar como referencia dos patrones arquitectónicos: el MVC (*Model-View-Controller*) o el MVVM (*Model-View-ViewModel*). La principal diferencia entre estos dos patrones es que en el MVC es el controlador quien actualiza la vista de forma "manual"mientras que en el MVVM existe un componente encargado de sincronizar la vista automáticamente cuando se produce algún cambio en modelo. La documentación de *Android* recomienda usar la arquitectura MVVM para el desarrollo de las aplicaciones, sin embargo, para este proyecto se utilizará el patrón MVC por el hecho de estar más familiarizado con él y porque, según los requisitos, las funcionalidades de la aplicación no va a ser necesario realizar cambios en el modelo.

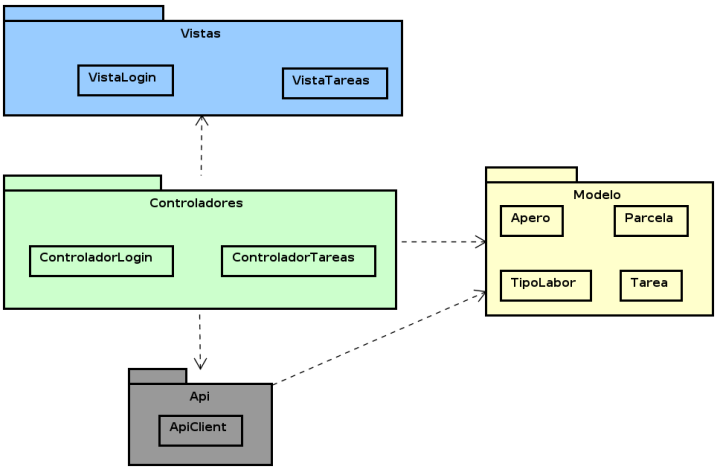

Figura 4.3: Arquitectura del aplicación *Android*.

Angular

Angular sigue una arquitectura basada en módulos. Por defecto, al iniciar un proyecto, se genera un módulo raíz (*root module*) que sirve como punto de inicio para compilar la aplicación sin embargo, es conveniente agrupar funcionalidades similares en módulos adicionales. Además, para conseguir una aplicación funcional en *Angular* es necesario definir otros elementos como son los componentes y los servicios:

- Los componentes que permiten generar las vistas con *HTML* y las directivas propias de *Angular* y sus controladores que se codifican utilizando *TypeScript*.
- Los servicios se utilizan para generar funcionalidades externas a los componentes como puede ser la de compartir datos entre estos. Su principal ventaja es que se pueden reutilizar.

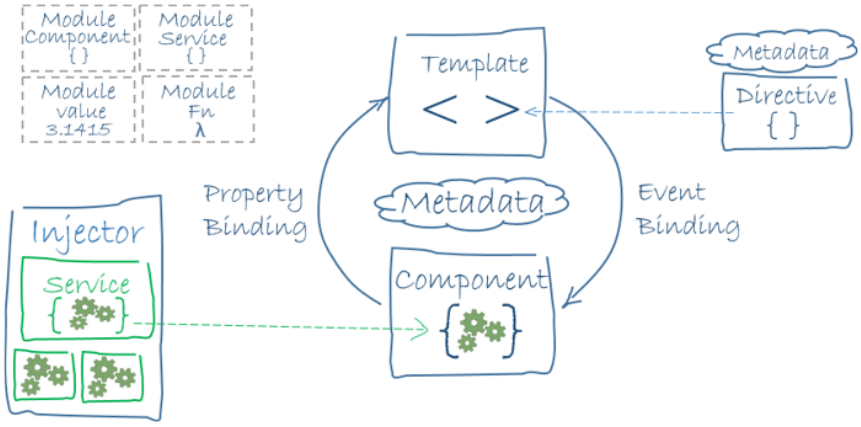

Figura 4.4: Arquitectura del aplicación web.

4.3.2. *Back-end.*

Arquitectura general del *back-end*

Es esta sección se describe el contenido de los paquetes del *back-end* comenzando por su arquitectura general y en las siguientes figuras detallando el contenido de cada uno de los paquetes.

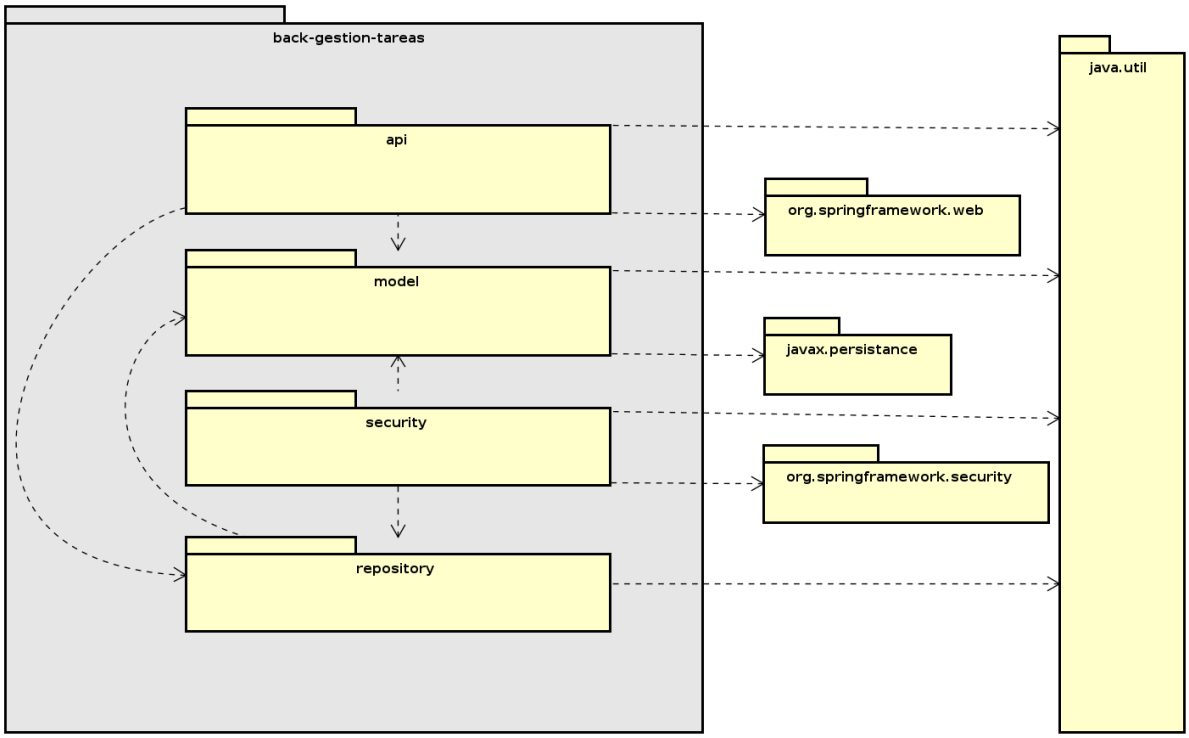

Figura 4.5: Arquitectura general del *back-end.*

Como se puede observar en la figura anterior, la aplicación del *back-end* ha sido dividida en cuatro paquetes:

- Paquete *api*: este paquete contendrá los controladores de la API REST y que serán los encargados de gestionar las peticiones que vengan por parte de las aplicaciones que forman el *front-end.* Se definirán seis controladores dos para las operaciones relacionadas con los agricultores, uno para las operaciones relacionadas con las operaciones del administrador, otro para las operaciones relacionadas con los responsables de las cooperativas, uno para la gestión del registro de nuevos usuarios y el último encargado de las operaciones relacionadas con los tipos de labores. Respecto a la organización de los recursos, estos se han agrupado en los diferentes controladores.

    - En el controlador *PrivateAgricultoresController* se han agrupado *Agricultor/es*, *Parcela/s*, *Apero/s* y *RegistrosAgenda.*

    - En el controlador *PrivateCooperativaController* se han agrupado *Cooperativa/s*, *Producto/s* y *Precios*.
    - *PrivateTareasController* se encarga de las operaciones relacionadas con el recurso *Tarea/s*.
    - En el controlador *PublicApiController* agrupa las operaciones relacionadas con el registro de *Cooperativas* y *Agricultores*.
    - *PrivateAgricultoresController* se encarga de las operaciones relacionadas con el recurso *Usuario/s*.
    - *PrivateTiposLaborController* se encarga de las operaciones relacionadas con el recurso *TipoLabores*.
- Paquete *model*: contendrá las *entities* del sistema.
- Paquete *security*: contendrá todas las clases cuya funcionalidad está relacionada con el control de acceso y seguridad del sistema. Entre alguna de las funcionalidades de esta capa está la de la configuración de aplicaciones que pueden acceder a los datos, generación de *tokens* para los usuarios que se hayan identificado correctamente o control de acceso a las URLs de la API REST según el rol del usuario.
- Paquete *repository*: es el encargado de hacer las consultas a la base de datos.

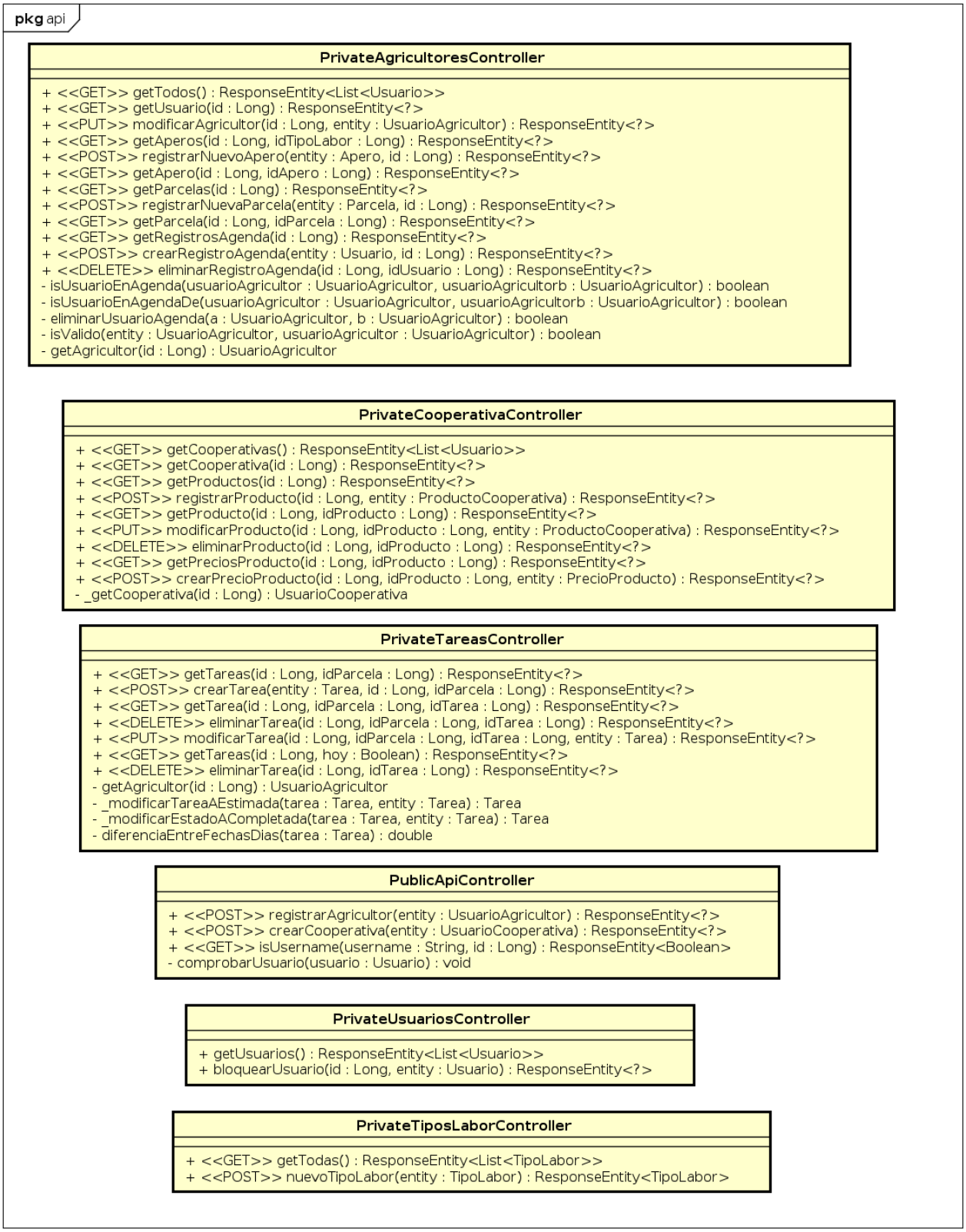

Figura 4.6: Paquete *api*.

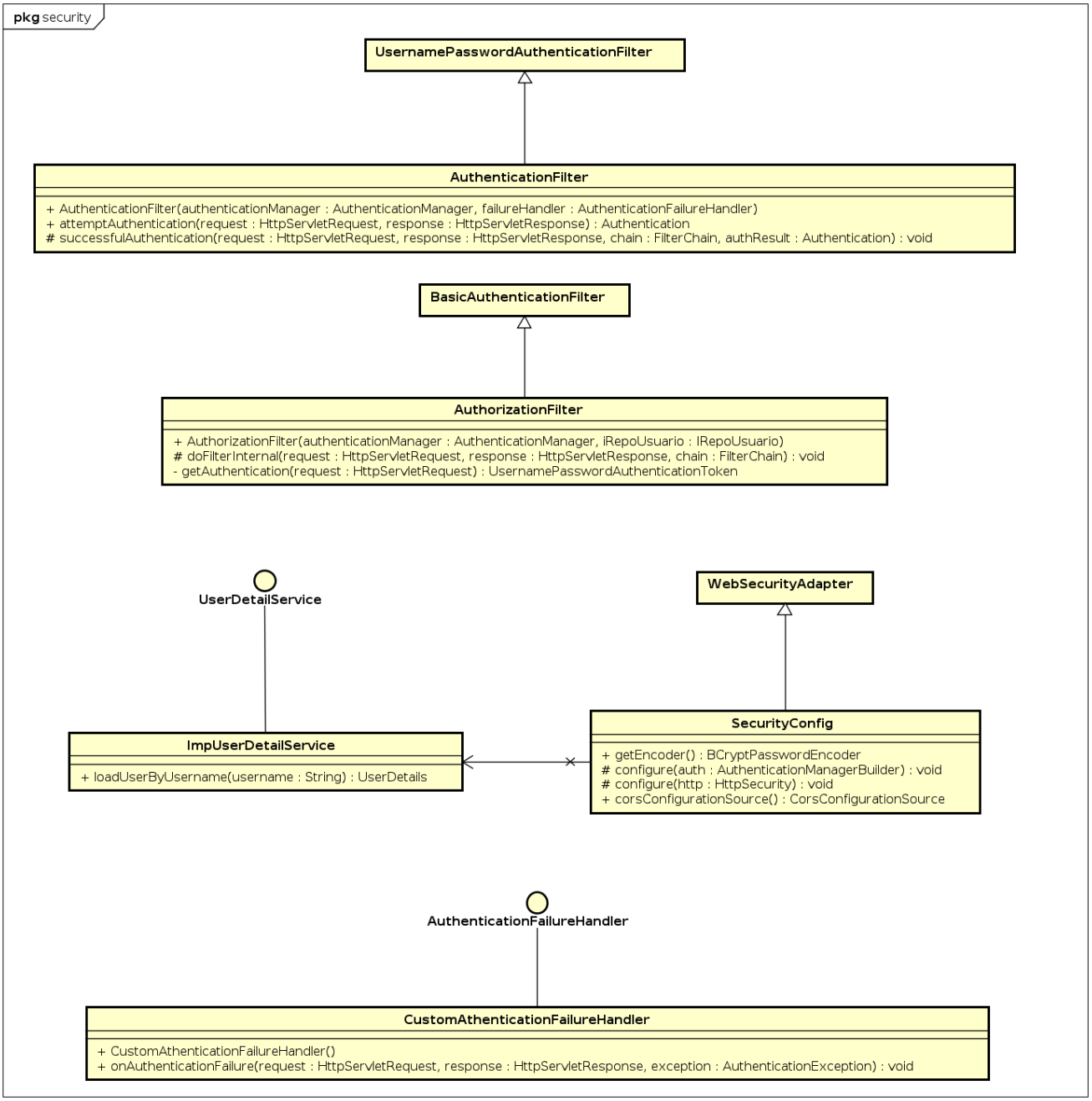

Figura 4.7: Paquete *security*.

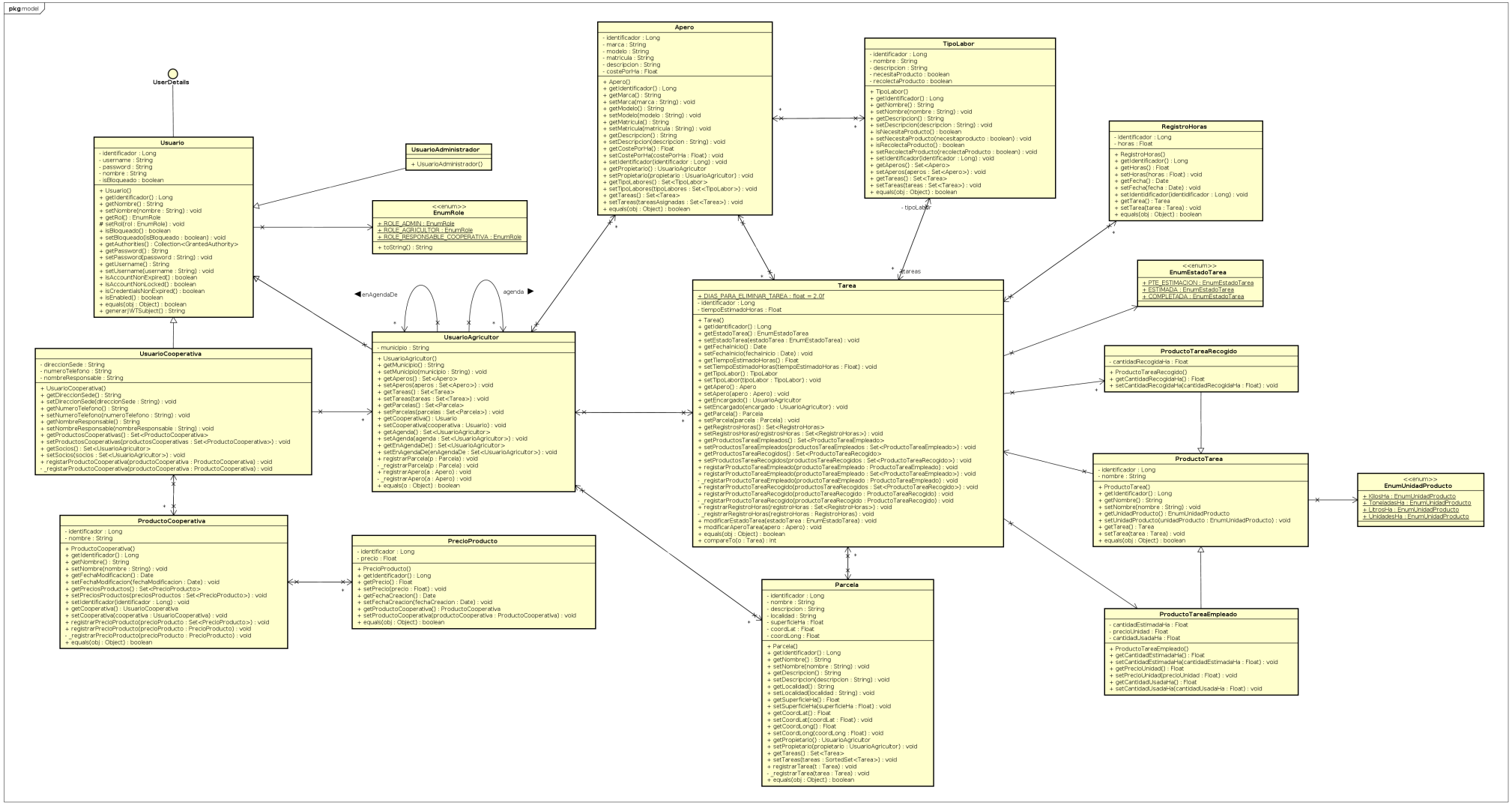

Figura 4.8: Paquete *model*.

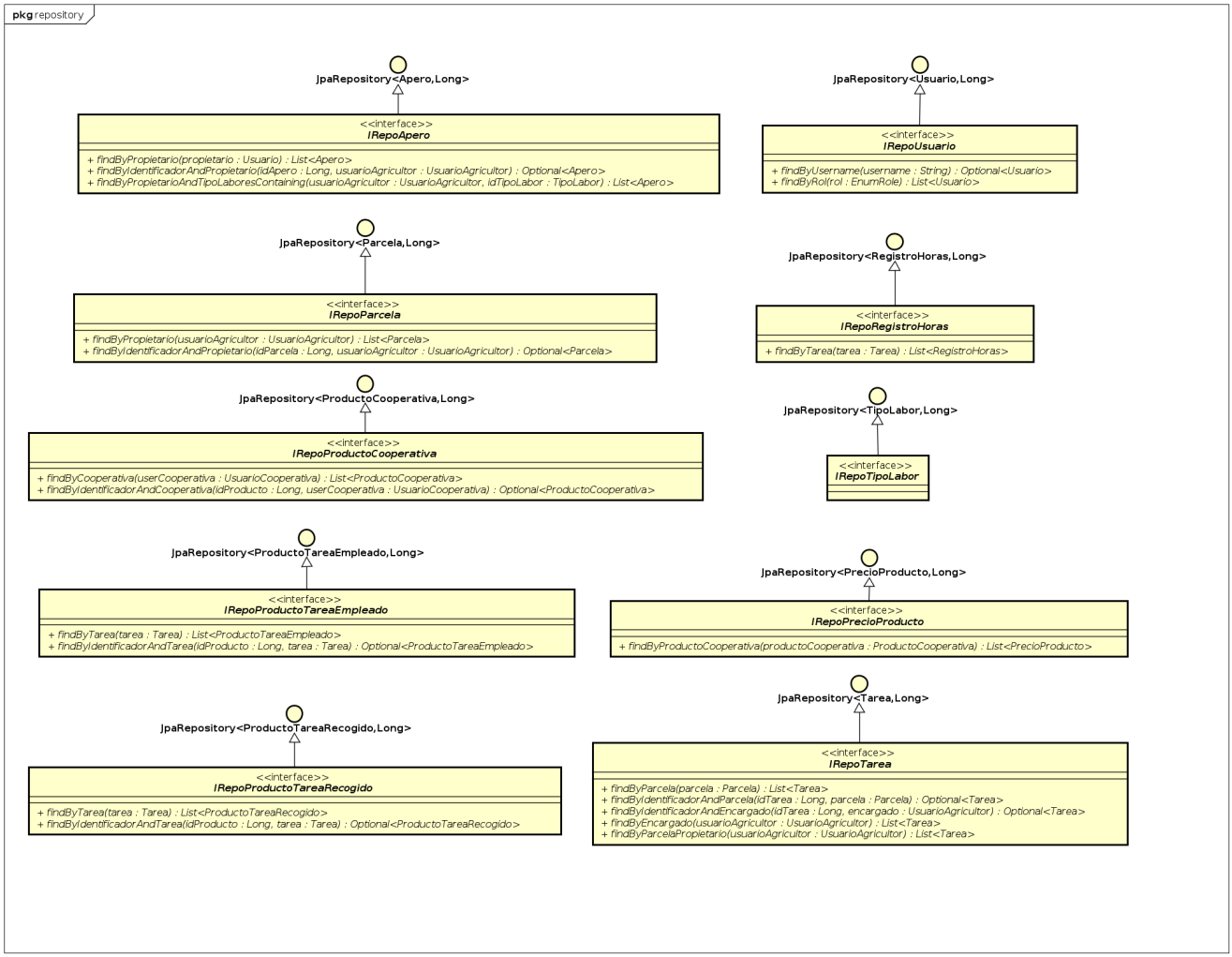

Figura 4.9: Paquete *repository*.

4.4. Diagramas de secuencia en diseño

A continuación se describen los diagramas de secuencia de los principales casos de uso del sistema.

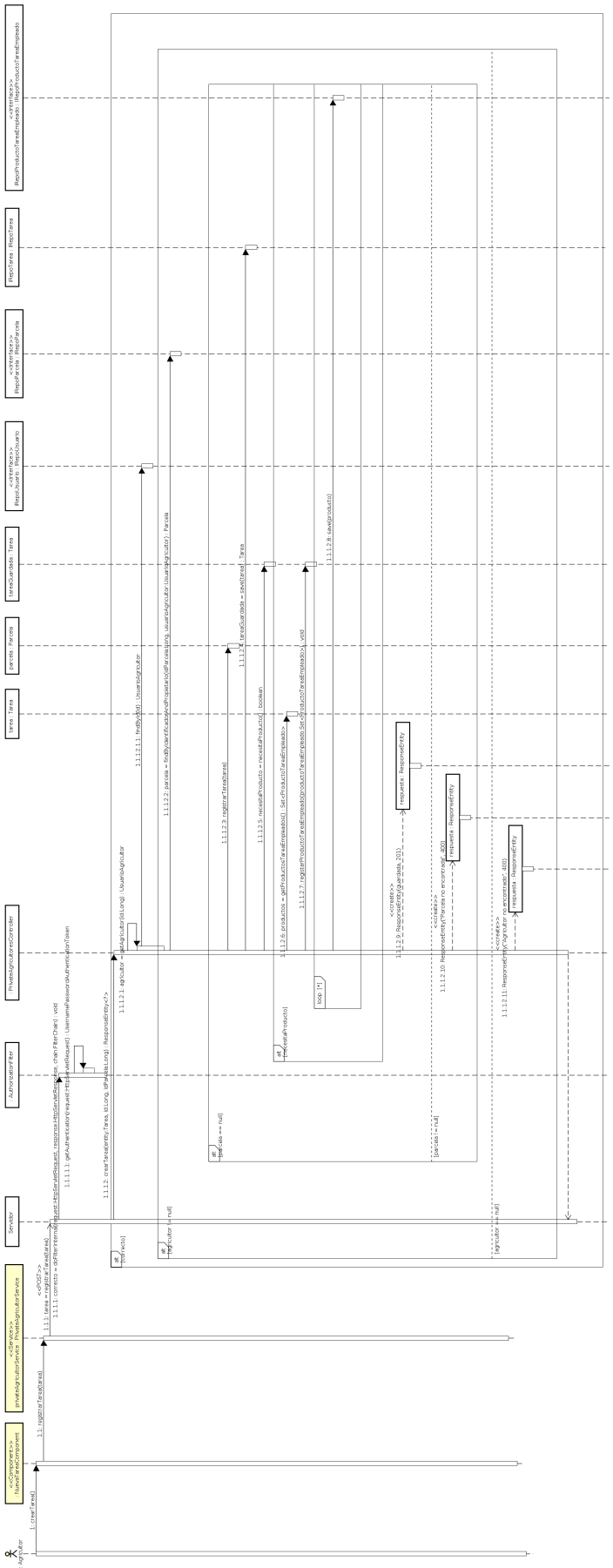

Figura 4.10: Diagrama de secuencia en diseño: Crear tarea.

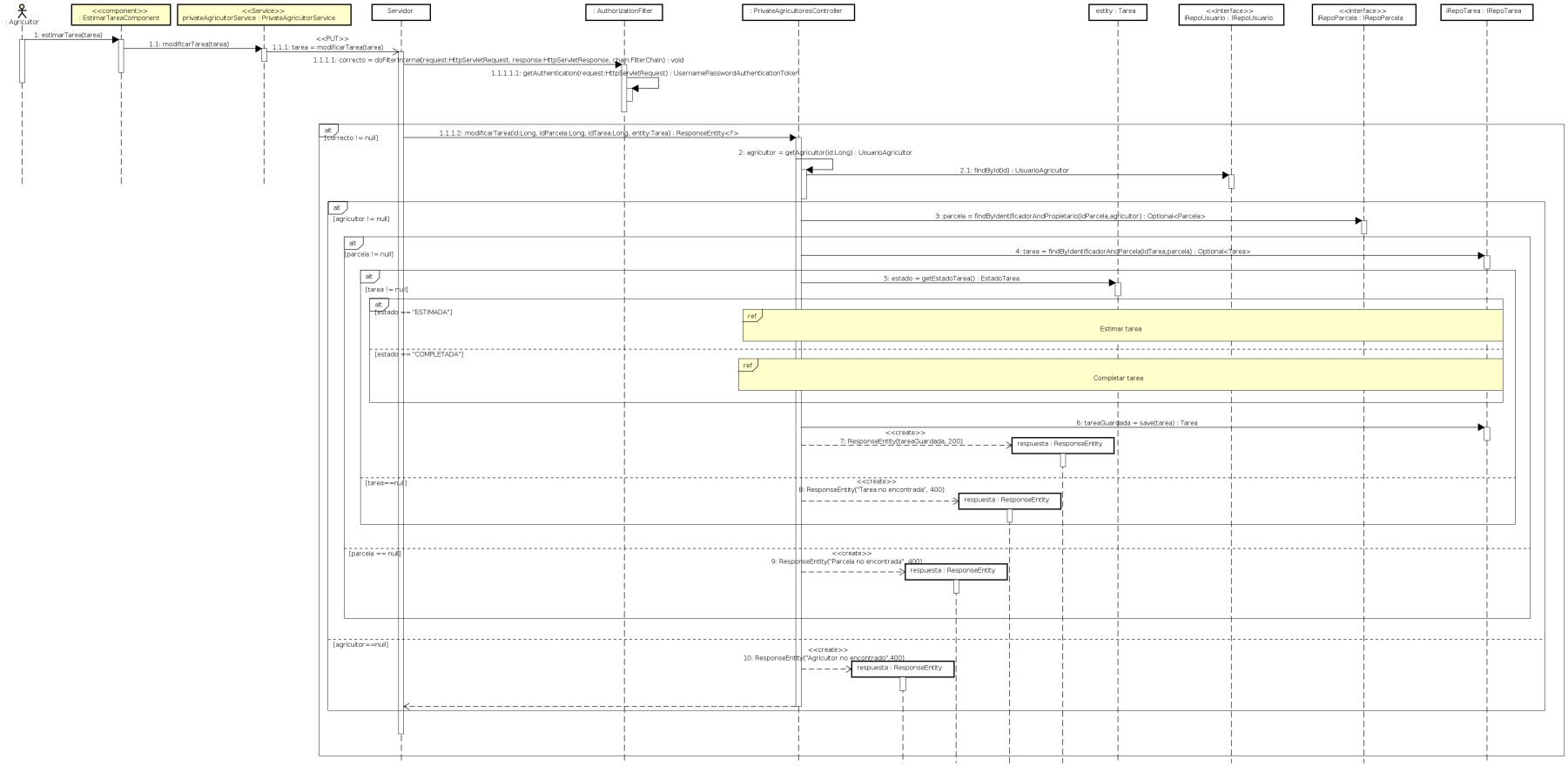

Figura 4.11: Diagrama de secuencia en diseño: Modificar tarea.

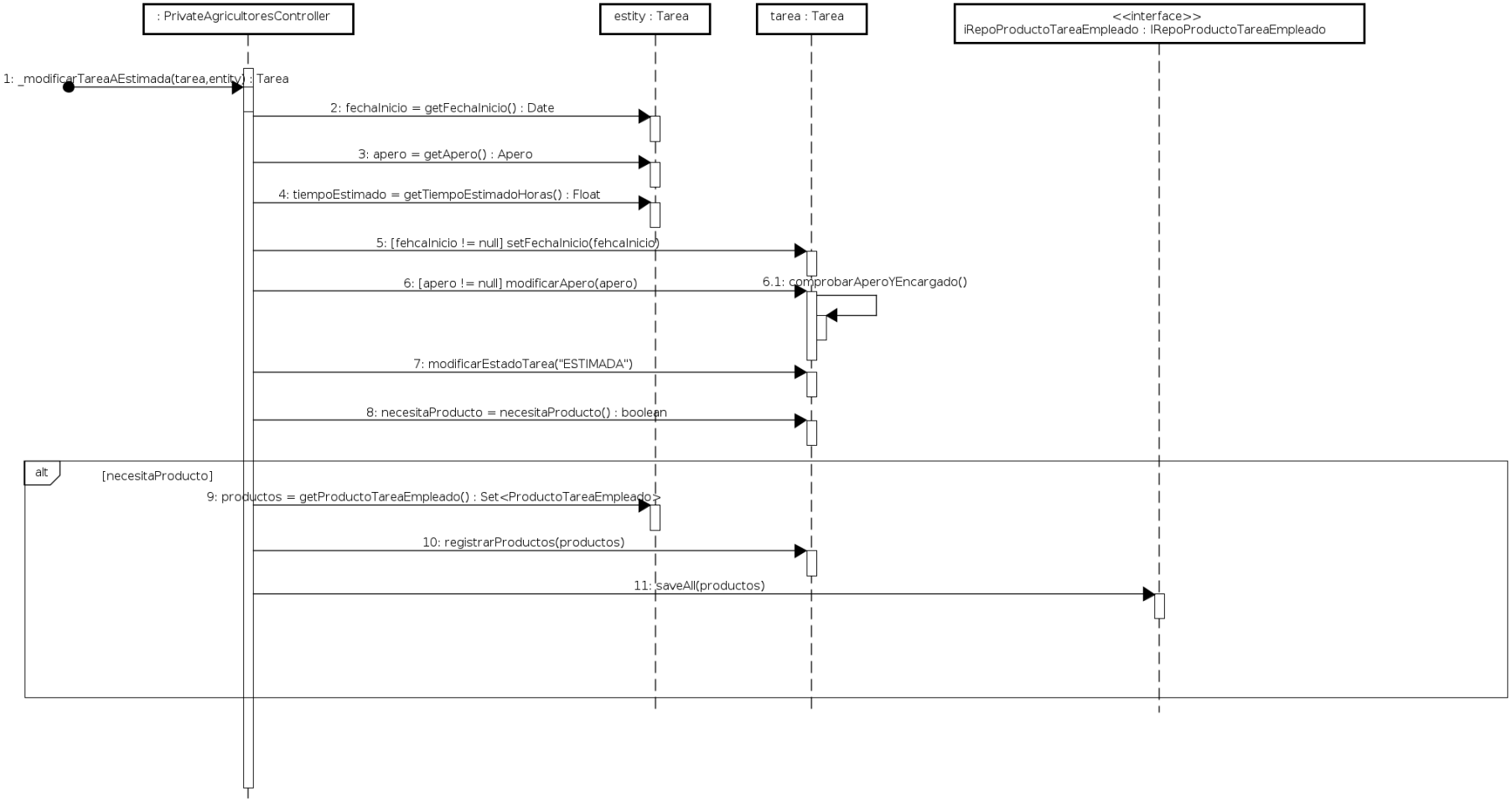

Figura 4.12: Diagrama de secuencia en diseño: Estimar tarea.

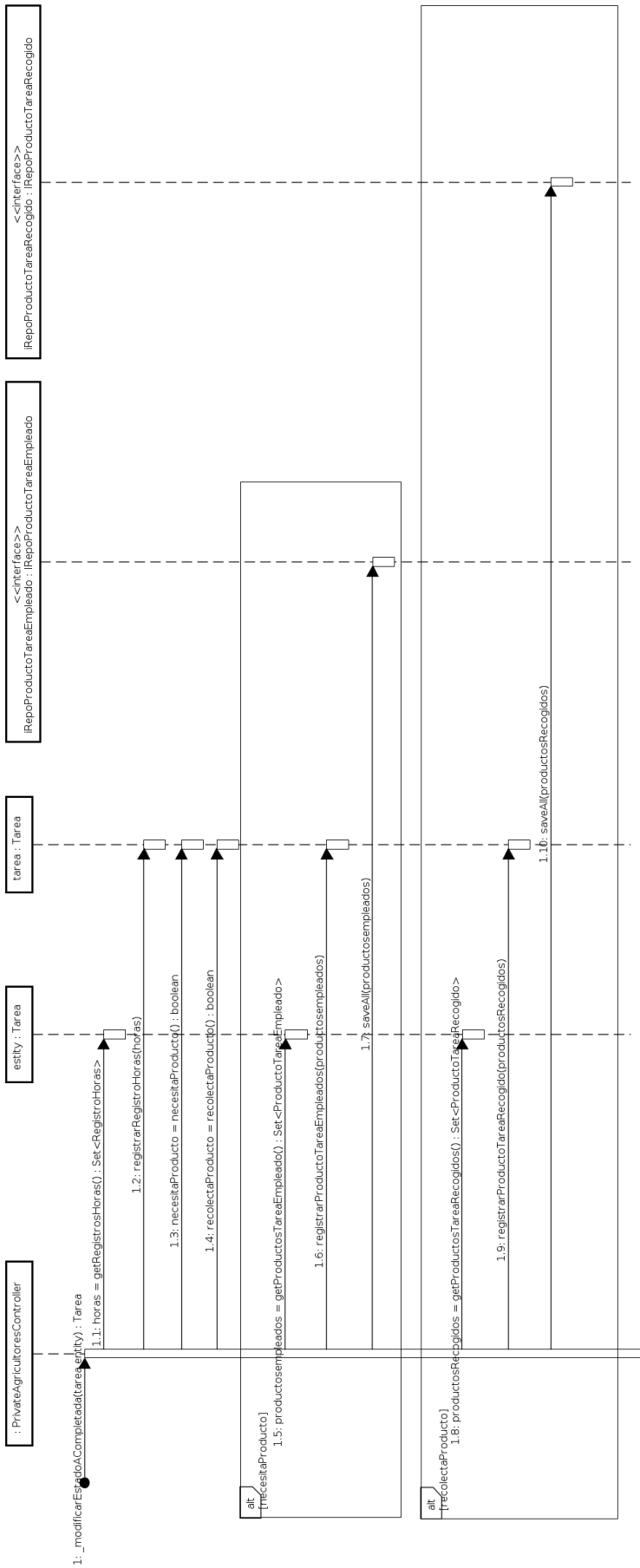

Figura 4.13: Diagrama de secuencia en diseño: Completar tarea.

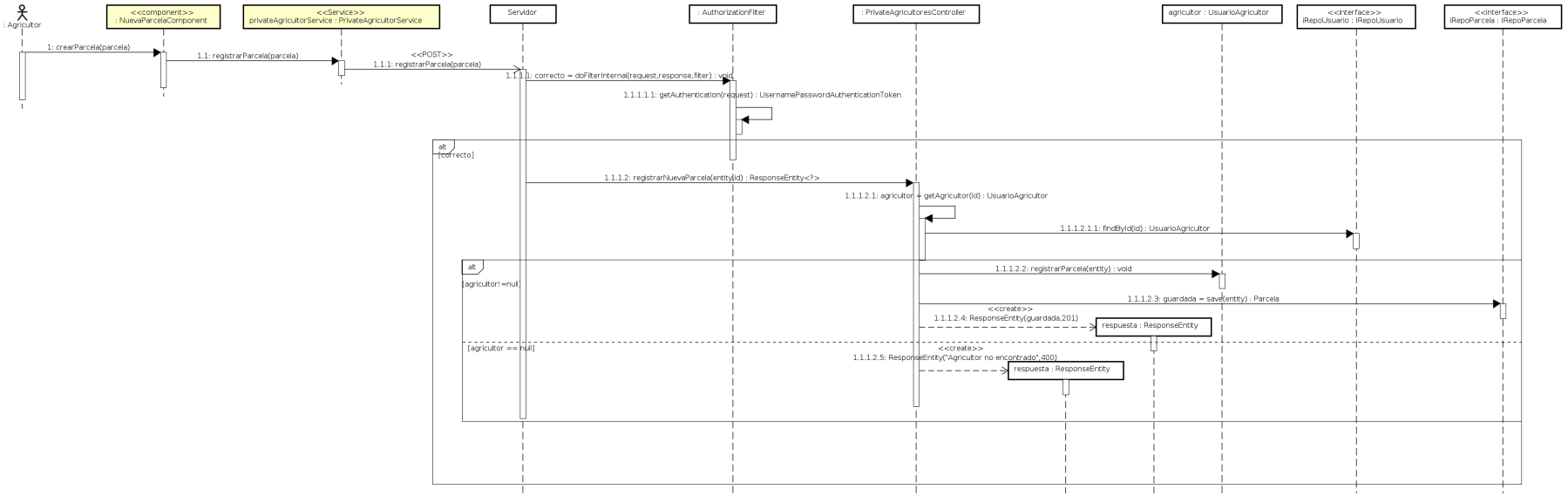

Figura 4.14: Diagrama de secuencia en diseño: Registrar parcela.

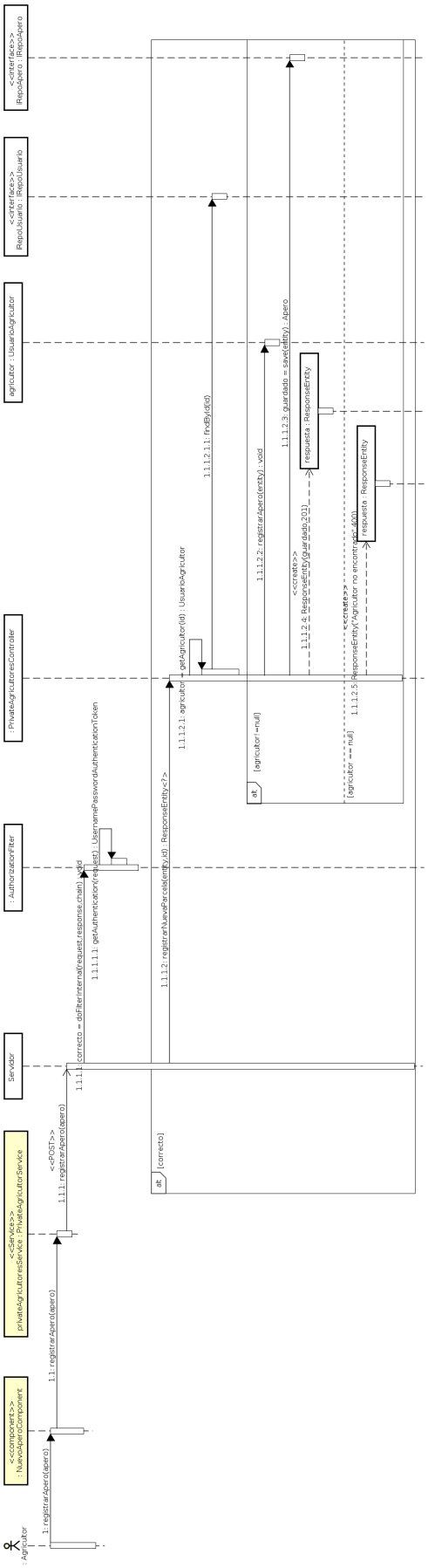

Figura 4.15: Diagrama de secuencia en diseño: Registrar parcela.

4.5. Diseño de las vistas.

En esta sección se presentan los prototipos de las pantallas de la aplicación. Estos prototipos se usarán como referencia para la implementación de las vistas reales de la aplicación.

4.5.1. Aplicación web

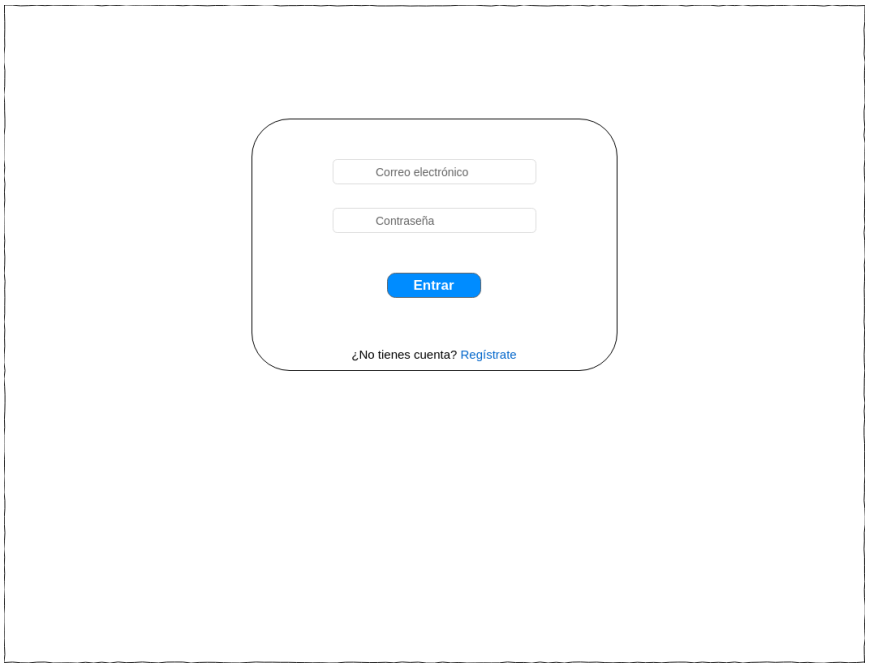

Figura 4.16: Prototipo pantalla login.

Actor Agricultor

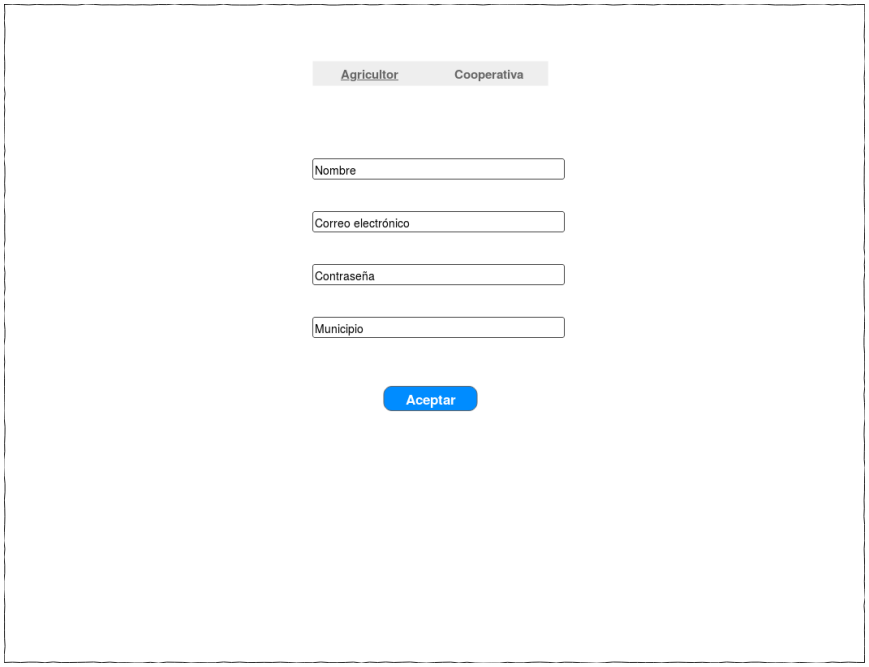

Figura 4.17: Prototipo pantalla registro agricultor.

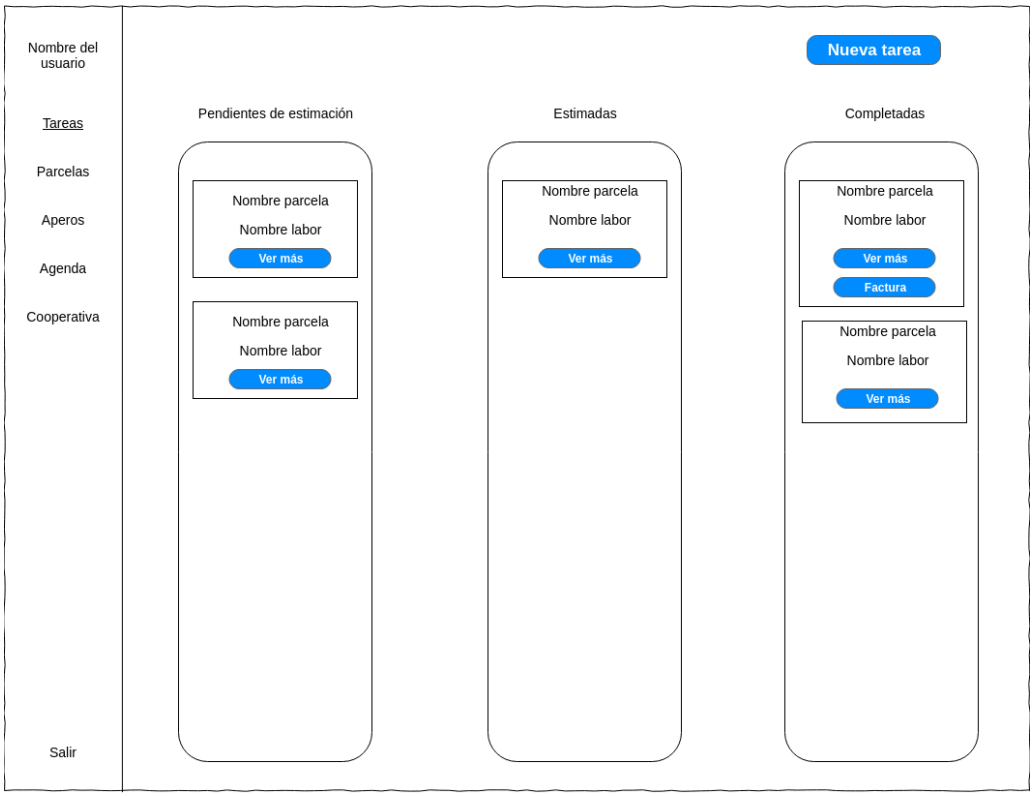

Figura 4.18: Prototipo pantalla ver tareas.

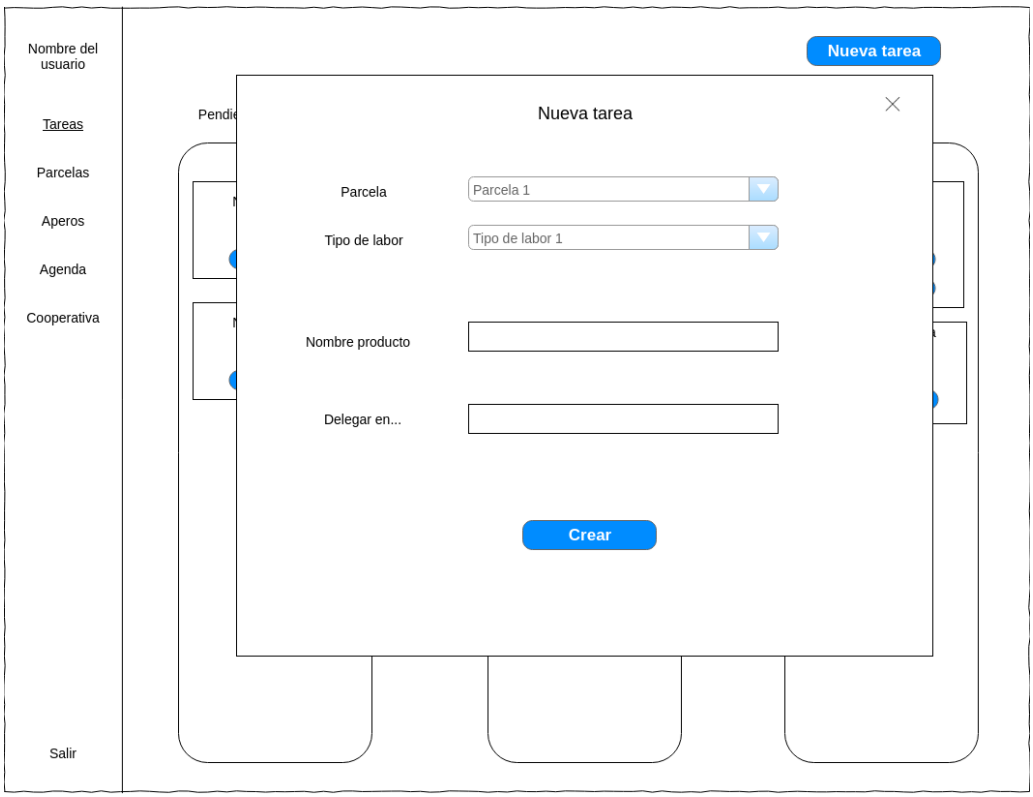

Figura 4.19: Prototipo pantalla nueva tarea.

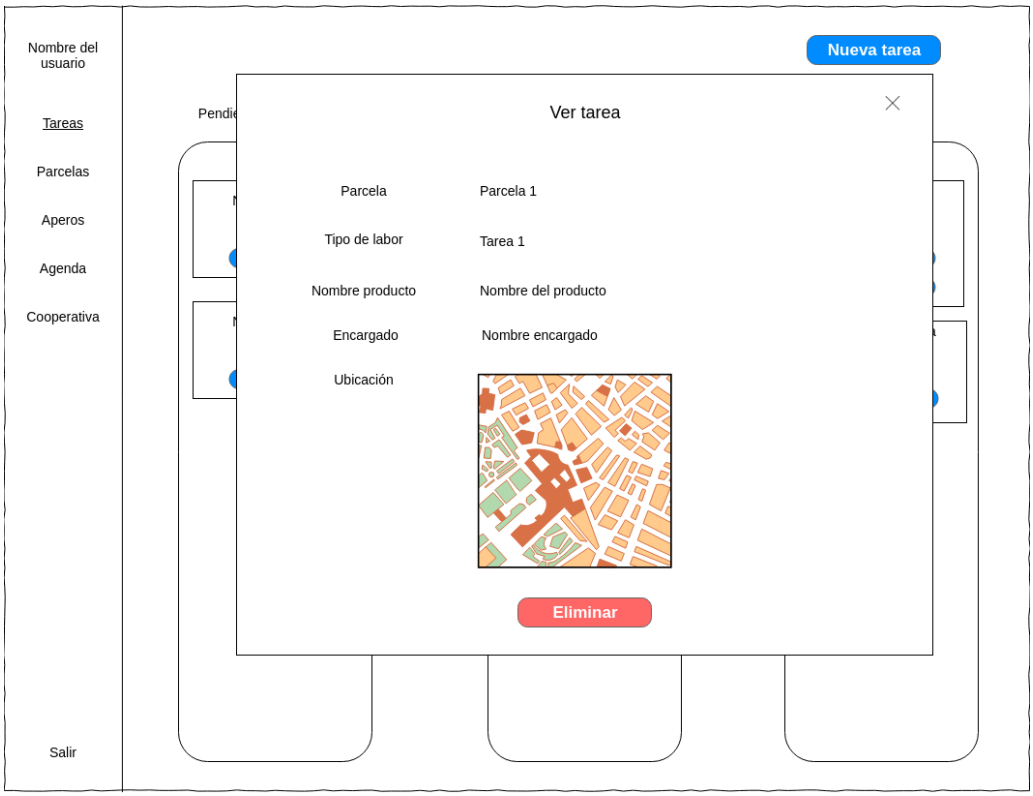

Figura 4.20: Prototipo pantalla ver tarea pendiente de estimación.

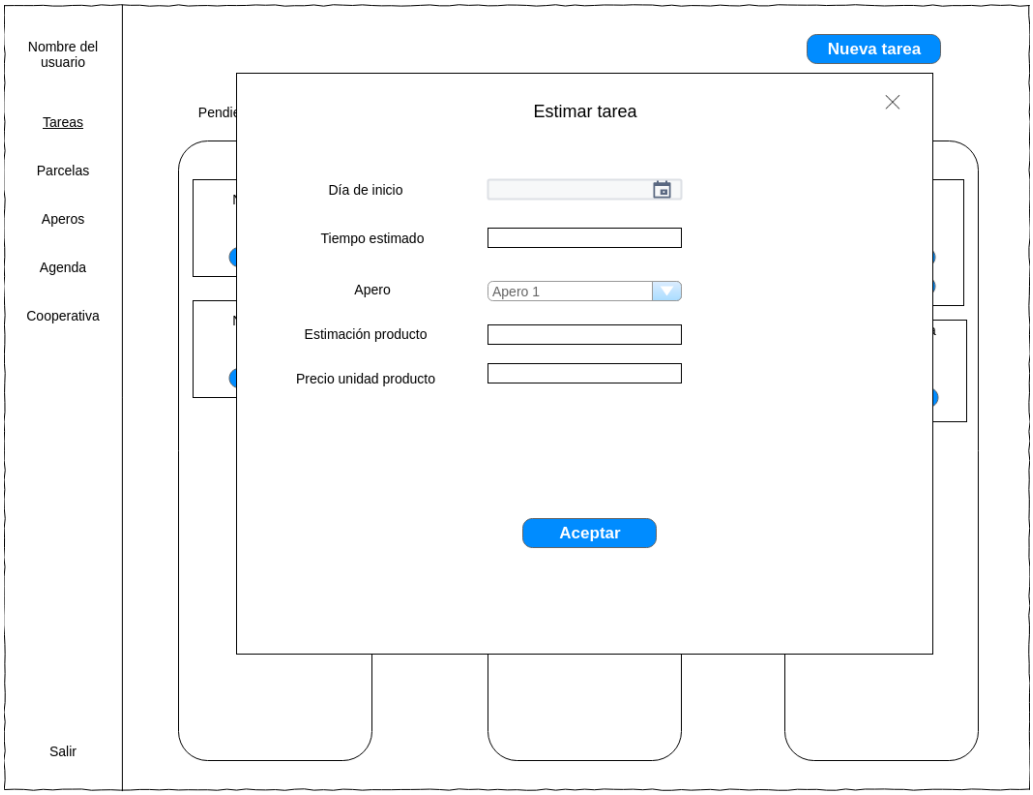

Figura 4.21: Prototipo pantalla estimar tarea.

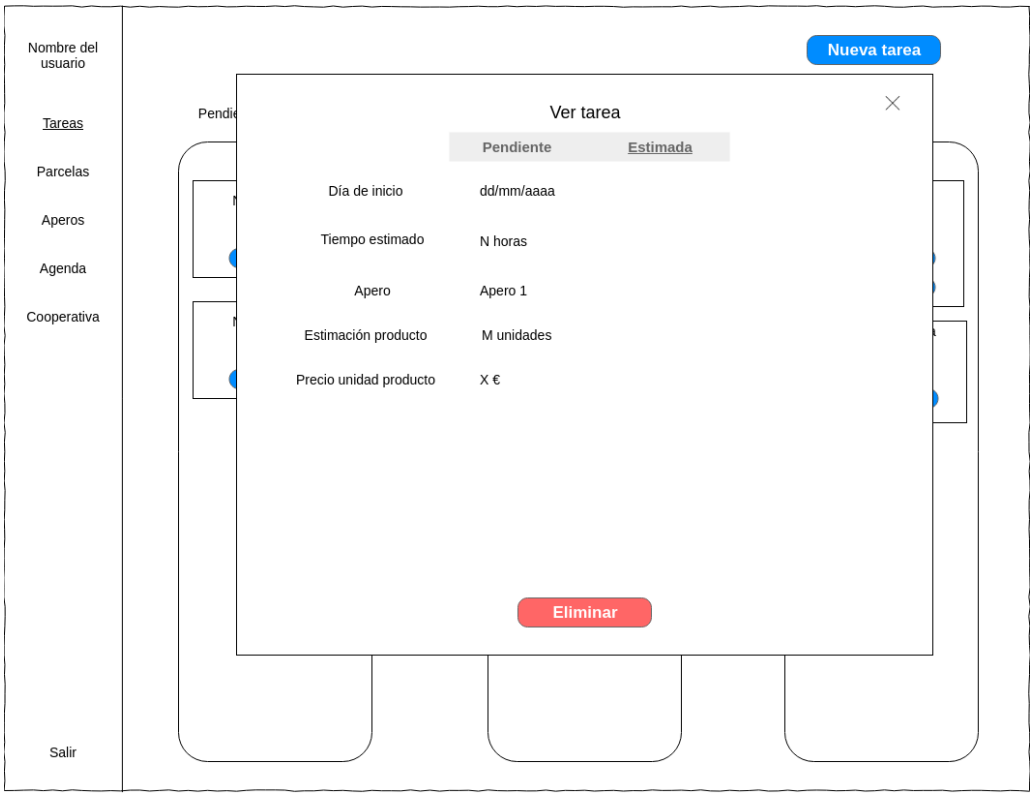

Figura 4.22: Prototipo pantalla ver tarea estimada.

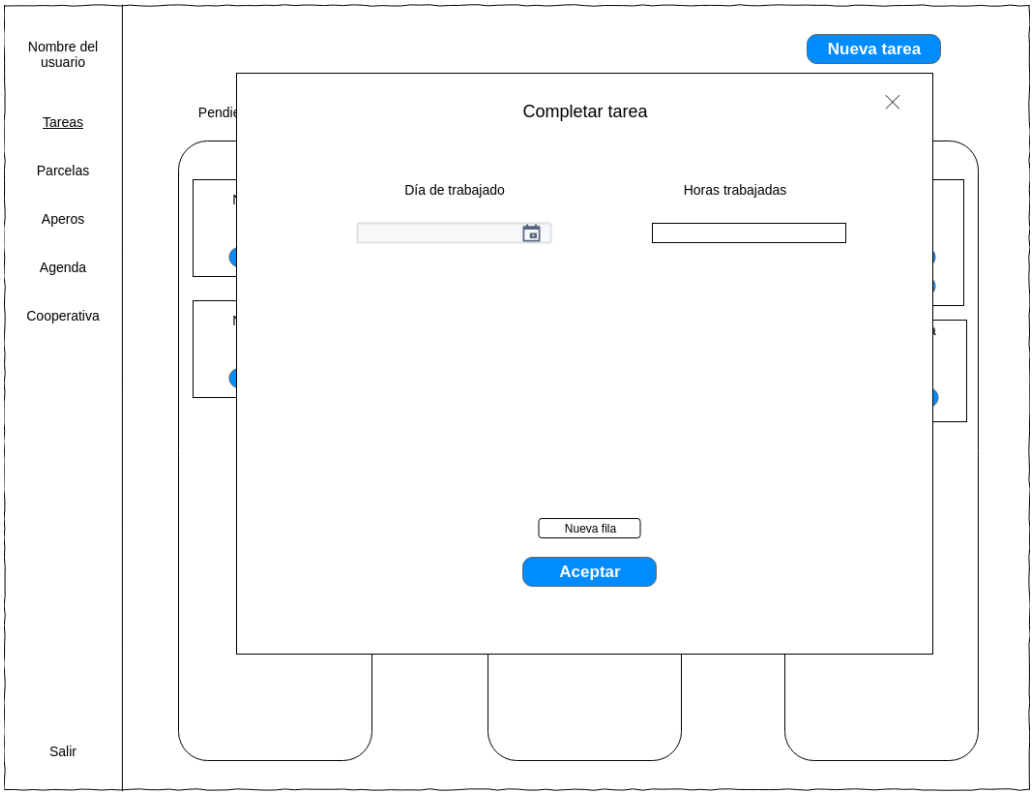

Figura 4.23: Prototipo pantalla completar tarea.

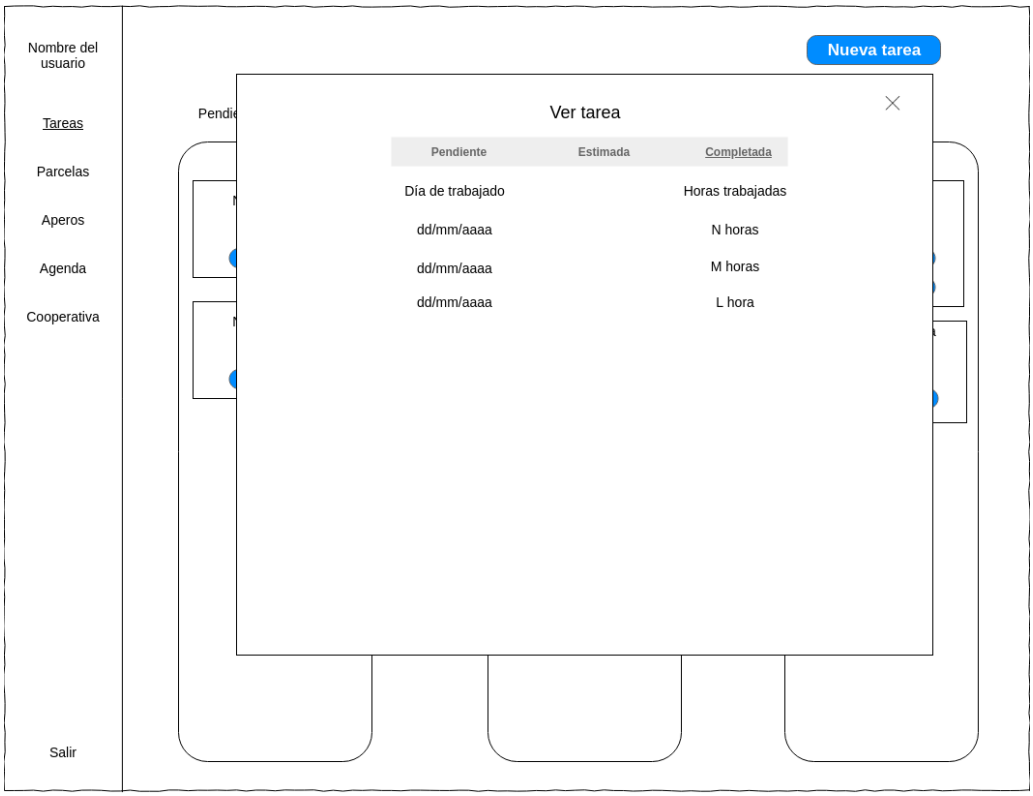

Figura 4.24: Prototipo pantalla ver tarea completa.

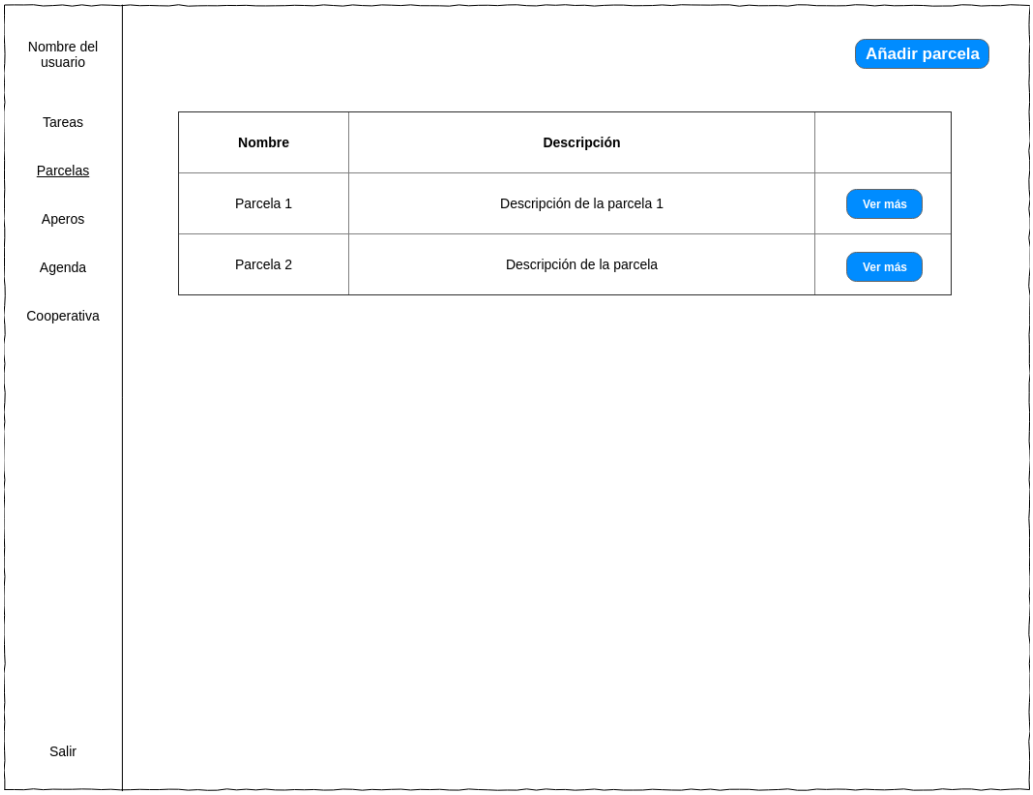

Figura 4.25: Prototipo pantalla ver parcelas.

Figura 4.26: Prototipo pantalla ver parcela.

Figura 4.27: Prototipo pantalla añadir parcela.

Figura 4.28: Prototipo pantalla ver aperos.

Figura 4.29: Prototipo pantalla ver apero.

Figura 4.30: Prototipo pantalla añadir apero.

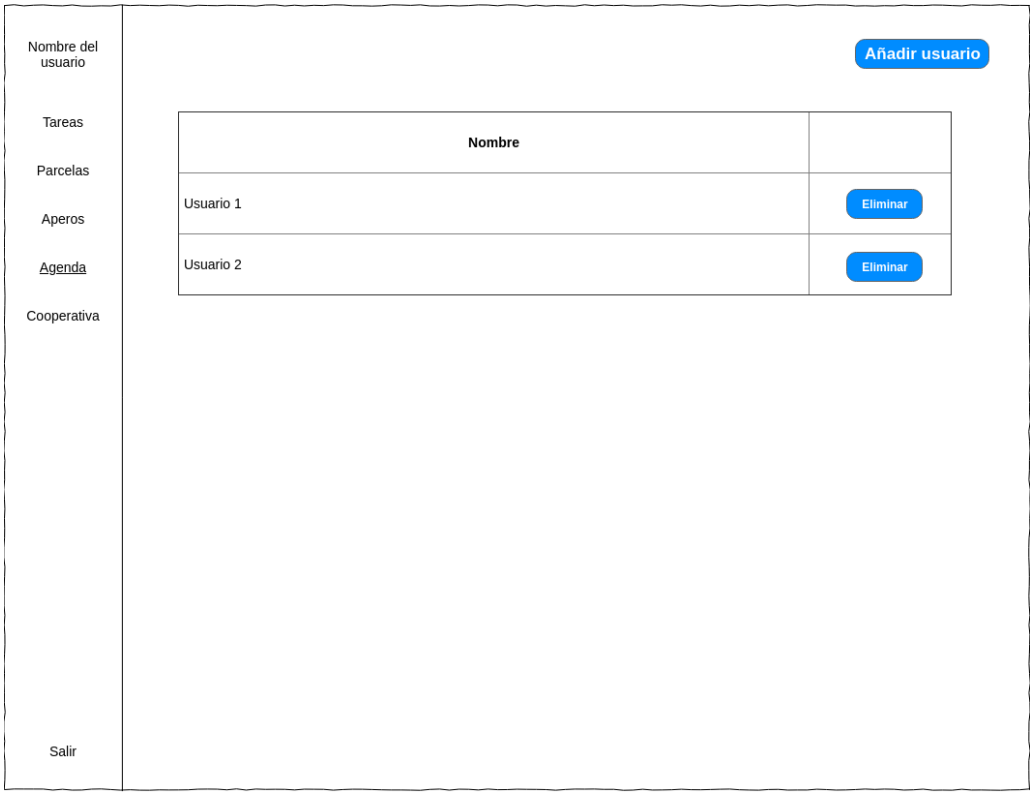

Figura 4.31: Prototipo pantalla ver agenda.

Figura 4.32: Prototipo pantalla añadir usuario a la agenda.

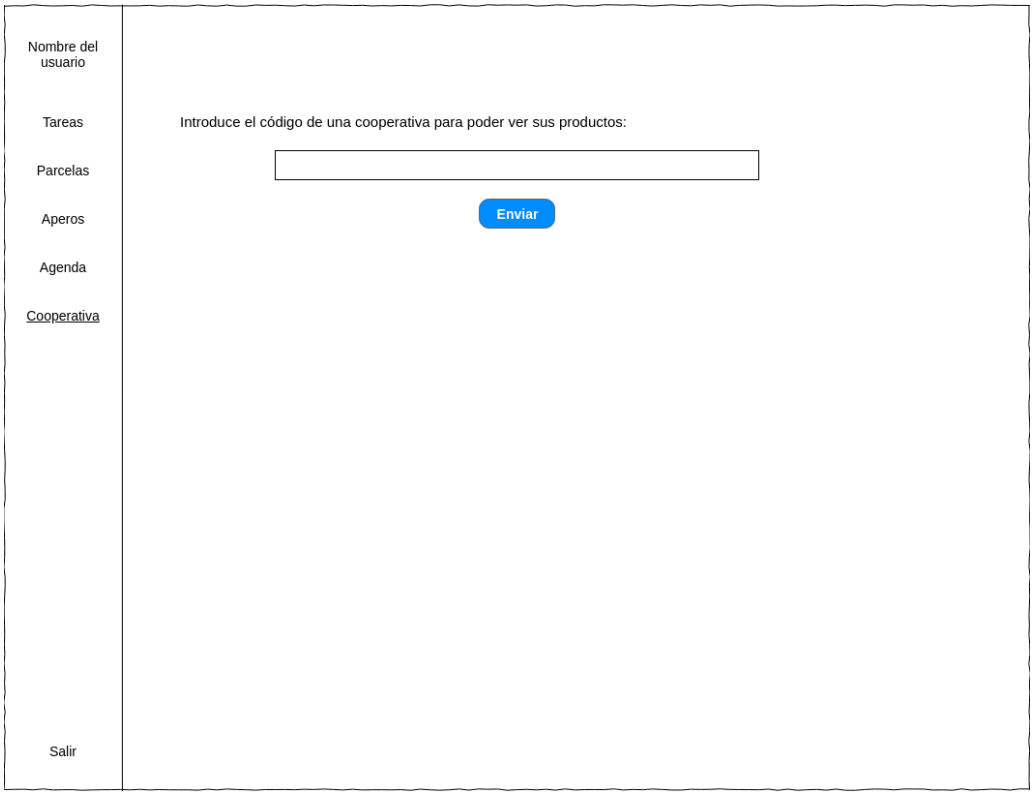

Figura 4.33: Prototipo pantalla agricultor sin cooperativa.

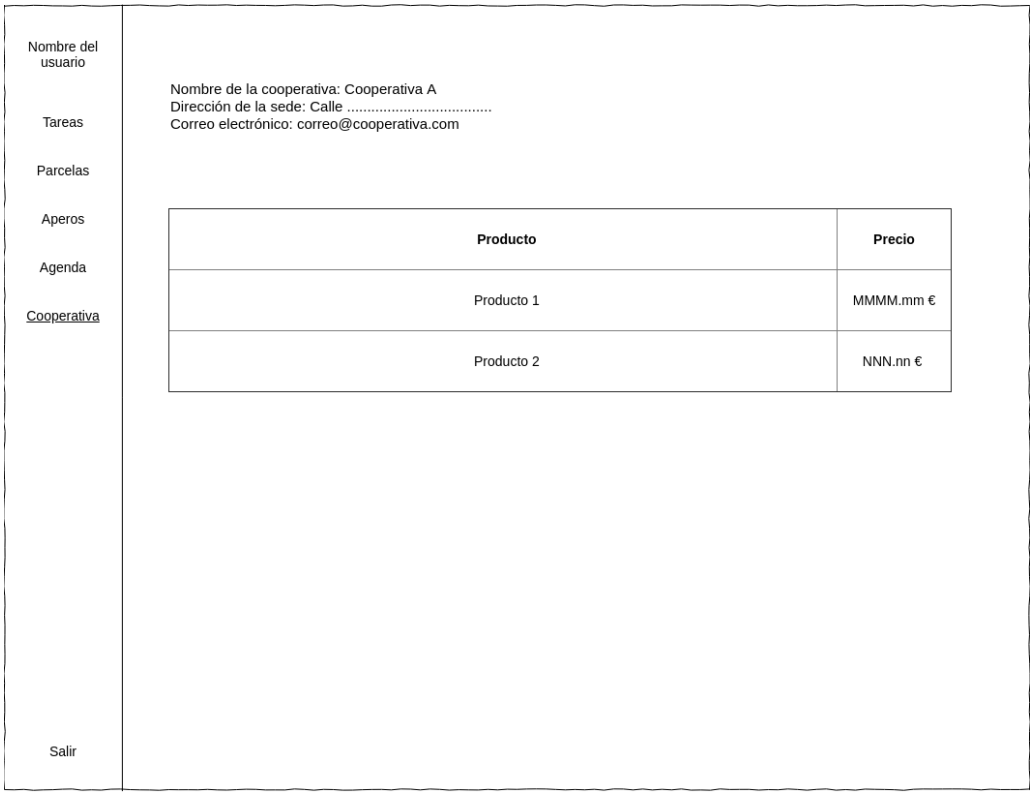

Figura 4.34: Prototipo pantalla ver productos cooperativa.

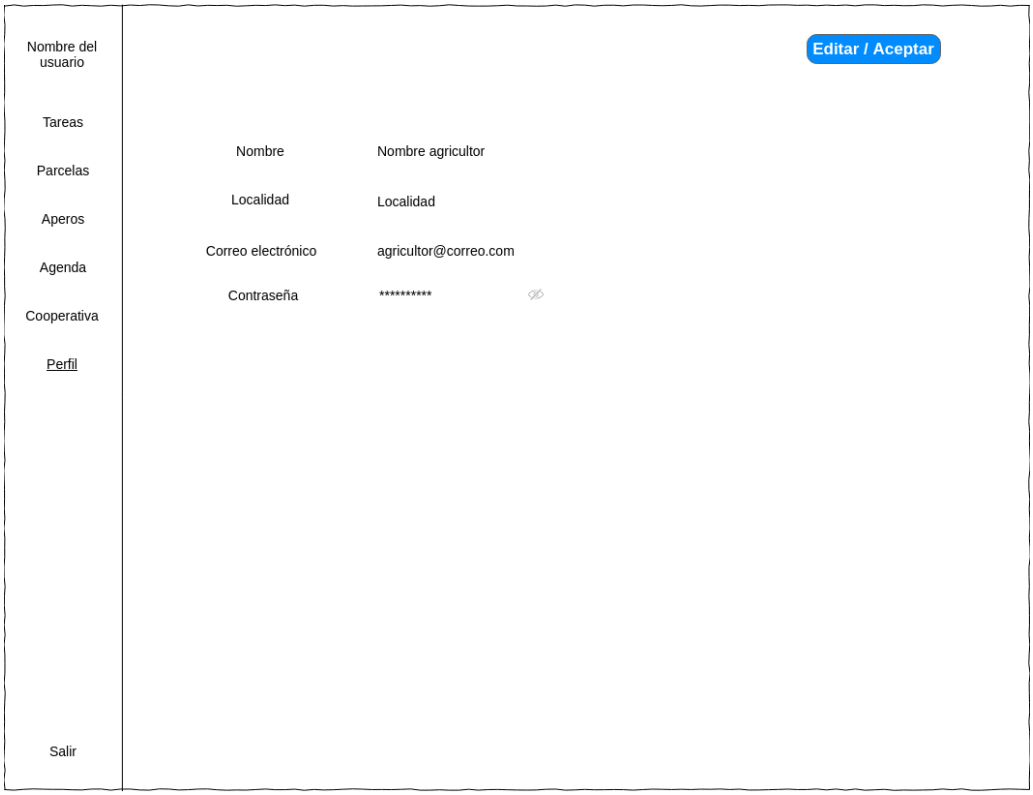

Figura 4.35: Prototipo pantalla ver perfil.

En esta última vista, la apariencia es similar si se está editando el perfil o sólo consultándolo. El único cambio es que si se esta viendo el texto del botón es *Editar* y si se está editando el texto es *Aceptar*.

Actor Cooperativa

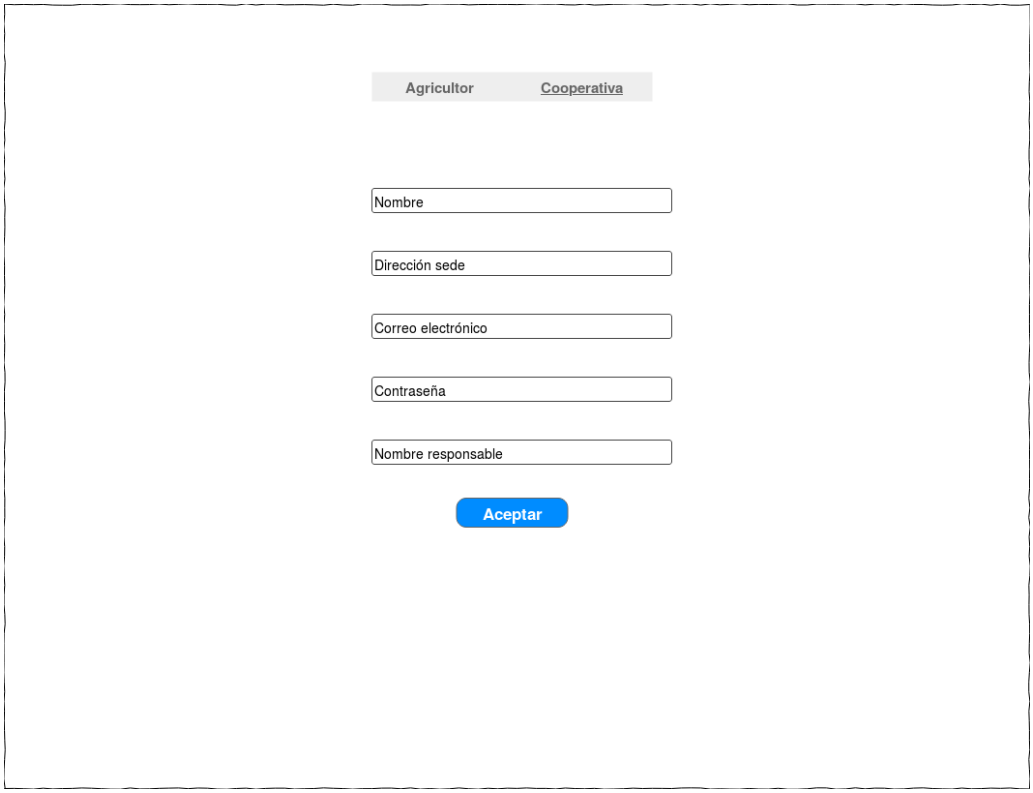

Figura 4.36: Prototipo pantalla registro cooperativa.

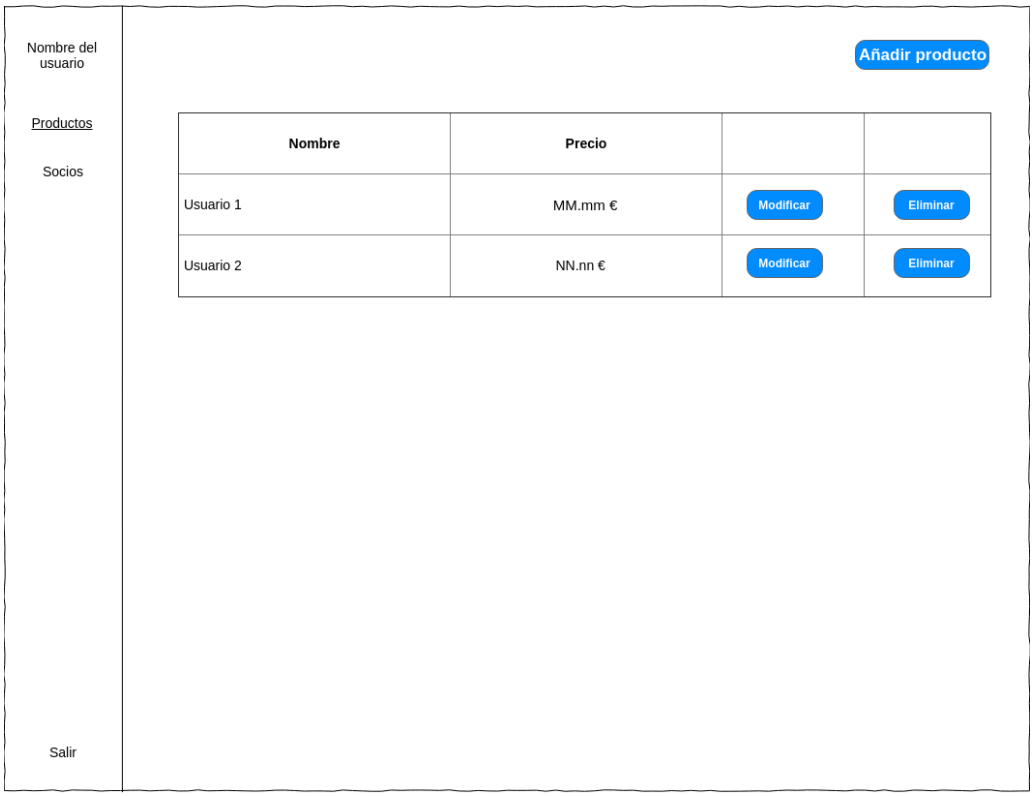

Figura 4.37: Prototipo pantalla ver productos cooperativa.

Figura 4.38: Prototipo pantalla añadir producto cooperativa.

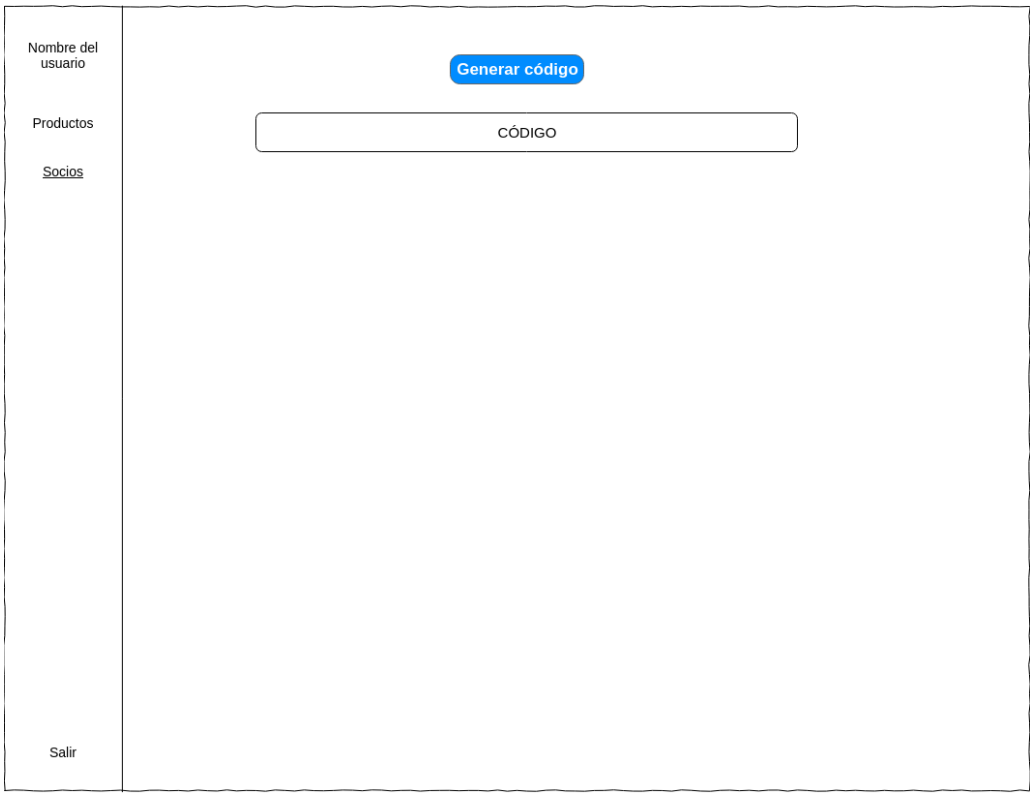

Figura 4.39: Prototipo pantalla generar código.

Actor Administrador

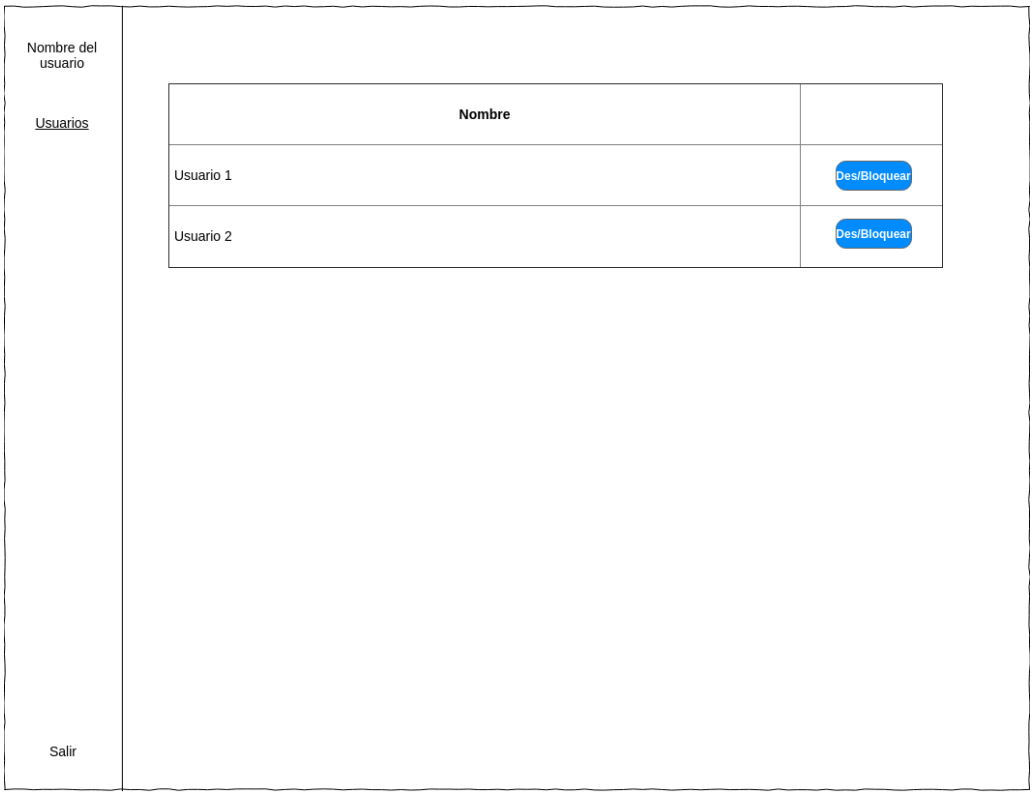

Figura 4.40: Prototipo pantalla ver usuarios.

4.5.2. Aplicación Android

Figura 4.41: Prototipo pantalla login.

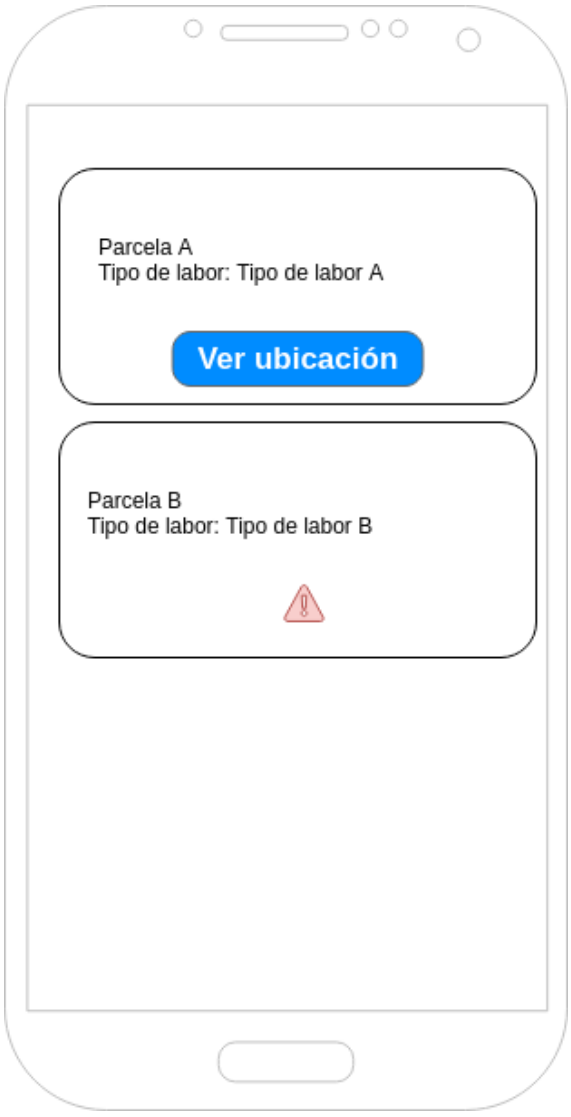

Figura 4.42: Prototipo pantalla ver tareas.

Capítulo 5

Implementación

En este capítulo se expondrán los detalles de la implementación que se consideren interesantes y que puedan ayudar a la comprensión de la implementación del sistema.

5.1. *Back-end*

Como se ha ido mencionado a lo largo de la memoria la parte *back-end* ha sido desarrollado utilizando *SpringBoot* un framework de Java. Este framework permite añadir funcionalidades a las clases de Java mediante las anotaciones que ofrecen sus diferentes proyectos. La primera anotación que aparece cuando se crea una aplicación con *SpringBoot* es *@SpringBootApplication* que se utiliza como punto de origen para buscar los demás componentes de la aplicación y realizar una configuración automática de esta. Otras anotaciones que se pueden encontrar en los distintos paquetes de esta aplicación son las siguientes.

Paquete api

- *@RestController* utilizada para definir servicios *RESTful*.
- *@RequestMapping* utilizada para indicar, según la URL, que peticiones serán atendidas en ese controlador. Por ejemplo:

```
@RequestMapping("/private/agricultores")
```

  Hay que tener en cuenta otros archivos de configuración de la aplicación a la hora de hacer peticiones a las URL de los controladores.

- *@GetMapping* utilizada para relacionar el método del controlador con las operaciones *GET* a la URL que se obtiene al concatenar el valor de *@RequestMapping* con el valor del argumento *value*. Por ejemplo:

```
@GetMapping(value = "/{id}/parcelas") //GET a /api/private/agricultores/{id}/parcelas
public ResponseEntity<?> getParcelas(@PathVariable Long id)
```

- *@PostMapping* utilizada para relacionar el método del controlador con las operaciones *POST* a la URL que se obtiene al concatenar el valor de *@RequestMapping* con el valor del argumento *value*. Por ejemplo:

```
@PostMapping(value = "/{id}/parcelas") // POST a /api/private/agricultores/{id}/parcelas
public ResponseEntity<?> registrarNuevaParcela(@RequestBody Parcela entity, @PathVariable
    Long id)
```

- *@PutMapping* utilizada para relacionar el método del controlador con las operaciones *PUT* a la URL que se obtiene al concatenar el valor de *@RequestMapping* con el valor del argumento *value*. Por ejemplo:

```
@PutMapping(value = "/{id}/parcelas/{idParcela}/tareas/{idTarea}") // PUT a /api/private/
    agricultores/{id}/parcelas/{idParcela}/tareas/{idTarea}
public ResponseEntity<?> modificarTarea(@PathVariable Long id, @PathVariable Long idParcela,
       @PathVariable Long idTarea, @RequestBody Tarea entity)
```

- *@DeleteMapping* utilizada para relacionar el método del controlador con las operaciones *DELETE* a la URL que se obtiene al concatenar el valor de *@RequestMapping* con el valor del argumento *value*. Por ejemplo:

```
@DeleteMapping(value = "/{id}/tareas/{idTarea}") // DELETE a /api/private/agricultores/{id}/
    tareas/{idTarea}
public ResponseEntity<?> eliminarTarea(@PathVariable Long id, @PathVariable Long idTarea)
```

- *@PathVariable* para obtener el valor de algunos de los campos de la URL Por ejemplo:

```
@GetMapping(value = "/{id}")
public ResponseEntity<?> getUsuario(@PathVariable Long id) // El valor de variable id es el
    que se pasa en la URL.
```

- *@RequestBody* para asignar a una variable el cuerpo de un petición HTTP. Por ejemplo:

```
@PutMapping(value = "/{id}")
public ResponseEntity<?> modificarAgricultor(@PathVariable Long id, @RequestBody
    UsuarioAgricultor entity)
```

- *@RequestParam* para asignar el valor de una query en la URL a una variable. Entre los argumentos que se han utilizado están *required* de tipo boolean, *name* de tipo string y *defaultValue*. Por ejemplo:

```
@GetMapping(value = "/{id}/tareas") // GET /api/private/agricultores/{id}/tareas?hoy=true
 public ResponseEntity<?> getTareas(@PathVariable Long id,
    @RequestParam(name = "hoy", required = false, defaultValue = "false") Boolean hoy)
```

- *@Autowired* para crear una instancia de una clase. Por ejemplo:

```
@Autowired
private IRepoUsuario iRepoUsuario;
```

- *@JsonView* para indicar que propiedades se quieren serializar/deserializar. Por ejemplo:

```
@JsonView(Views.ListView.class)
```

Paquete model

- *@Entity* presente en la mayoría de las clases de este paquete se utiliza para representar una clase en una tabla de la base de datos. Como argumentos se ha utilizado *name* para indicar el nombre de la tabla. Por ejemplo:

```
@Entity(name = "ProductosCooperativas")
public class ProductoCooperativa
```

- *@Id* utilizada para indicar que un atributo hará la función de *primary key* en la table.

```
@Id
private Long identificador;
```

- *@GeneratedValue* utilizada para generar los identificadores de las tablas. El único argumento que se ha utilizado ha sido *strategy* con valor *GenerationType.IDENTITY* que asigna identificadores de 1 a n a cada elemento de la tabla. Existen otros valores como *Sequence* que comparte el contador para toda la base de datos o *Table* que obtiene los valores de otra tabla, pero no se ha considerado oportuno emplearlos.

```
@GeneratedValue(strategy = GenerationType.IDENTITY)
private Long identificador;
```

- *@Column* utilizada representar este atributo como una columna en la base de datos. De esta anotación se han usado los atributos *name* para indicar el nombre de la columna, *nullable* para indicar si esa columna puede estar vacía o no o *updatable* para indicar si esa columna se podía actualizar. Por ejemplo:

```
@Column(name = "id", nullable = false, updatable = false)
private Long identificador;
```

- *@OneToMany* utilizada para representar relaciones 1 - *. Esta anotación se añade en el lado 1 de la relación. Entre los argumentos que se han utilizado están *mappedBy* cuyo valor es nombre del atributo del otro lado de la relación, *orphanRemoval* para eliminar los elementos en cascada cuando se elimina el elemento "padre" y *fetch* para indicar si se quiere que se instancien los "hijos" cuando se crea una instancia del "padre". Por ejemplo:

```
@OneToMany(mappedBy = "productoCooperativa", orphanRemoval = true, fetch = FetchType.EAGER)
private Set<PrecioProducto> preciosProductos;
```

- *@ManyToOne* utilizada de forma inversa que *@OneToMany* es decir, en el lado * de una relación 1 - *. El argumento que se ha utilizado ha sido *optional* para indicar si ese atributo puede tener valor nulo. Por ejemplo:

```
@ManyToOne(optional = false)
private ProductoCooperativa productoCooperativa;
```

- *@ManyToMany* y *@JoinTable* utilizadas para representar las relaciones * - *. De la primera anotación se ha usado el argumento *fetch* y de la segunda *name* para indicar el nombre de la tabla intermedia, *joinColumns* y *inverseJoinColumns* para indicar las columnas de la tabla intermedia. Por ejemplo:

```
@ManyToMany(fetch = FetchType.EAGER)
@JoinTable(name = "rel_apero_tipo_labor", joinColumns = @JoinColumn(name = "apero"),
    inverseJoinColumns = @JoinColumn(name = "tipoLabor"))
private Set<TipoLabor> tipoLabores;
```

```
@ManyToMany(mappedBy = "tipoLabores")
private Set<Apero> aperos;
```

- *@JsonIgnore* o *@JsonIgnoreProperties* para no serializar/deserializar alguno de los atributos. Por ejemplo:

```
@JsonIgnore
private Set<Tarea> tareas;
```

```
@JsonIgnoreProperties(value = { "authorities", "accountNonExpired", "accountNonLocked", "
    credentialsNonExpired","enabled" })
```

- *@MappedSuperclass* utilizada para que las clases que heredan de esta incluyan en su tabla los atributos de esta. Por ejemplo:

```
@MappedSuperclass
public class ProductoTarea
```

- *@Inheritance* para indicar la combinación que deben seguir la clase y la subclase cuando se representen en la base de datos. El argumento que se ha utilizado es *strategy* con valor *InheritanceType.JOINED* para que en la tabla de la clase padre incluya los atributos que están en común y las tablas de las subclases sólo contengan sus atributos. Por ejemplo:

```
@Inheritance(strategy = InheritanceType.JOINED)
public class Usuario
```

Paquete repository

En este paquete no se han utilizado anotaciones pero como punto interesante de la implementación cabe comentar que las interfaces que extienden *JpaRepository* se pueden definir métodos de consulta a la base de datos indicando el nombre de las propiedades que se quieren filtrar. Por ejemplo, el método

```
List<Apero> findByPropietarioAndTipoLaboresContaining(UsuarioAgricultor usuarioAgricultor, TipoLabor
    idTipoLabor);
```

hará una consulta a la base de datos filtrando por el propietario del apero y que en el conjunto de labores que pueda realizar ese apero contenga una determinada.

Paquete security

A nivel de anotaciones en este paquete solo aparecen tres:

- *@Autowired* que se ha mencionado en apartados anteriores.
- *@Bean* que se utiliza para que *SpringBoot* gestione las instancias de ese objeto.
- *@EnableWebSecurity* que se utiliza para que cuando se configure la aplicación tenga en cuenta el contenido de esa clase.

Sin embargo, la funcionalidad de este paquete es tiene bastante peso en el correcto funcionamiento del sistema ya que se encarga de validar quién hace las peticiones y comprobar que ese usuario sea capaz de ello. Esto se realiza con dos filtros:

1. *AuthenticationFilter* que se lanza cuando se un usuario se identifica en el sistema y comprueba las credenciales que se envían en el cuerpo de la petición y, si son válidas, devuelve un token.
2. *AuthorizationFilter* que se lanza cuando se intenta acceder a alguna de las URLs de la aplicación. Este filtro obtiene el token de autenticación en la cabecera de la petición se haya enviado y decodificando este comprueba que el usuario que ha hecho la solicitud tiene los permisos para acceder al contenido. Los permisos se definen en el método *configure* de la clase *SecurityConfig* con los métodos *antMatchers(String... antPatters)* para indicarle la URL y *hasRole* o *hasAnyRole* para indicar con un *String* o un array de este el rol o roles que pueden acceder a esa URL.

5.2. *Front-end*

5.2.1. Aplicación web

Para realizar un repaso de la implementación de la aplicación web desarrollada en *Angular* me voy a centrar en los elementos que tienen más peso a la hora del funcionamiento de la aplicación: los componentes y los servicios.

Componentes

Los componentes se han generado utilizando el comando *ng g c components/Nombre* donde:

1. ng es el comando relacionado con *Angular*.

2. g de *generate* se utiliza para crear nuevos elementos en la aplicación.

3. c de *component* para crear un nuevo componente.

4. components hace referencia a la carpeta donde se encuentran los componentes.

5. Y por último Nombre que hace referencia al nombre del componente.

Si la ejecución se ha completado correctamente se habrán creado cuatro nuevos archivos en este caso bajo la carpeta *components/nombre*:

- nombre.component.css para definir el estilo del componente.

- nombre.component.html para definir la plantilla del componente.

- nombre.component.spec.ts para hacer las pruebas unitarias del componente.

- nombre.component.ts para añadir la funcionalidad del componente.

Los dos que tienen más peso son los archivos nombre.component.html y nombre.component.ts. En el primero combinando etiquetas HTML, directivas de Angular (*ngIf, *ngFor,*ngSwitch ...) y etiquetas que surgen al crear nuevos componentes como se explicará más tarde se crean las vistas de las aplicaciones. Cuando se abre el archivo nombre.component.ts lo primero que se puede observar es lo siguiente:

```
@Component({
selector: "app-nombre",
templateUrl: "./nombre.component.html",
styleUrls: ["./nombre.component.css"]
})
```

1. selector hace referencia a una "nueva" etiqueta HTML que al incluirla en alguno de los archivos *.html de la aplicación incrustará la vista y la funcionalidad de este componente.

2. templateUrl hace referencia al archivo con el código HTML del componente.

3. styleUrls es un array con diferentes archivos *.css de los cuales coger el estilo.

Para implementar la funcionalidad de los componentes se ha seguido el mismo procedimiento.Estos tienen definido un ciclo de vida formado por ocho estados aunque los únicos que se han utilizado han sido *OnInit* y *OnDestroy*. En el estado *OnInit* lo ques e ha hecho ha sido solicitar los datos que se iban a mostrar en las vistas o inicializar los formularios:

```
ngOnInit() {
    this.tareas$ = this.privateAgricultor.getTareasAgricultor();
    this.subscription.add(
        this.tareas$.subscribe(
            (data: Tarea[]) => {
                this.tareasPteEstimacion = data.filter(
                    item => item.estadoTarea === EstadoTarea.PTE_ESTIMACION
                );
                this.tareasEstimadas = data.filter(
                    item => item.estadoTarea === EstadoTarea.ESTIMADA
                );
                this.tareasCompletadas = data.filter(
                    item => item.estadoTarea === EstadoTarea.COMPLETADA
                );
            },
            error => {
                this.tareasPteEstimacion = [];
                this.tareasEstimadas = [];
                this.tareasCompletadas = [];
                this.snackbarService.mostrar(
                    "No se ha podido cargar las tareas.",
                    "Ok",
                    "snackbar-alert"
                );
            }
        )
    );
}
```

```
ngOnInit() {
    sessionStorage.clear();
    this.subscriptions = new Subscription();
    this.loginForm = new FormGroup({
    username: new FormControl("", [Validators.required, Validators.email]),
    password: new FormControl("", Validators.required)
    });
}
```

En el estado *OnDestroy* lo único que se ha hecho ha sido *unsubscribe* de todas las suscripciones que se han necesitado para el funcionamiento del componente.

Servicios

Los servicios se han generado utilizando el comando *ng g s services/Nombre* donde:

1. ng es el comando relacionado con *Angular*.
2. g de *generate* se utiliza para crear nuevos elementos en la aplicación.
3. s de *service* para crear un nuevo servicio.
4. service hace referencia a la carpeta donde se encuentran los servicios.
5. Y por último Nombre que hace referencia al nombre del servicio.

Esto genera dos archivos nombre.service.spec.ts para hacer las pruebas unitarias de este y nombre.service.ts. Lo primero que no encontramos cuando abrimos el archivo nombre.service.ts es:

```
@Injectable({
    providedIn: "root"
})
```

que se utiliza para que *Angular* puede inyectar una instancia de este servicio en los componentes. La idea principal con la que se han creado los servicios era recuperar los datos desde el *back-end* para ello, lo primero que se necesita era obtener una instancia de *HttpClient* y definir la URL para poder recuperar los datos.

```
constructor(private http: HttpClient) {}
private url: string = `${environment.apiUrl}/private/cooperativas`;
```

Una vez hecho esto, se definieron diferentes métodos según los peticiones que se necesitasen hacer al *back-end*. Todos los métodos tienen una estructura similar: se define la URL y se retorna un *Observable* del tipo esperado que sea la respuesta. Para obtener este valor, se utilizan los métodos get, post, put o delete que proporciona la instancia de *HttpModule*

```
public getParcela(identificadorParcela: number): Observable<Parcela> {
    const endPointURL = `${this.url}/${this.getIdUsuarioLogged()}/parcelas/${identificadorParcela}`;
    return this.http.get<Parcela>(endPointURL);
}
```

Angular Material

Angular Material es una biblioteca que facilita la creación de estilos personalizados y con una apariencia actual para aplicaciones web desarrolladas con este *framework* de *TypeScript*. Cuando se comenzó a implementar la duda surgió la alternativa entre utilizar *Bootstrap* o esta biblioteca y, aunque ambas presentan funcionalidades similares, finalmente se decidió utilizar *Angular Material* por estar mejor adaptada a *Angular*.

Para personalizar el estilo de la aplicación en primer lugar, se añadió la dependencia al proyecto mediante el comando *ng add @angular/material*, se creó un archivo .scss (my-stile.scss) donde utilizando las funciones de esta biblioteca se definieron las pautas que debían seguir las vistas de la aplicación (tamaños de fuente, colores ...) y por último se añadió el nuevo estilo al archivo *angular.json*.

Para poder usar los componentes que presenta esta biblioteca sólo hay que importar el módulo correspondiente y añadir las propiedades correspondientes, por ejemplo, en un botón que sobresalga sobre el fondo

```
import { MatButtonModule } from "@angular/material/button";
...
imports: [
    MatButtonModule
    ...
]
```

```
<button mat-raised-button type="submit">Enviar</button>
```

o para mostrar *inputs* de un formulario de esta biblioteca

```
import { MatInputModule } from "@angular/material/input";
...
imports: [
    MatInputModule
    ...
]
```

```
<mat-form-field class="anchura100">
    <input matInput placeholder="Localidad" [formControlName]="'localidad'" />
    <mat-error>
        Este campo es obligatorio.
    </mat-error>
</mat-form-field>
```

5.2.2. Aplicación *Android*

La aplicación para dispositivos *Android* es una aplicación sencilla formada unicamente por dos actividades: *LoginActivity* y *MainActivity*. La funcionalidad de la primera es simplemente validar un formulario y realizar una petición al *back-end* para poder obtener el token y guardarlo en las *SharedPreferences*. En la segunda, una vez que se ha creado lo que se hace una consulta al *back-end* para obtener las tareas que el agricultor tiene programadas para ese día. Una parte interesante es la gestión de los permisos. La aplicación necesita obtener la última ubicación conocida del dispositivo para ordenar las tareas según la distancia entre el agricultor y la parcela, si el agricultor no proporciona los permisos, esta ordenación no se puede hacer. El procedimiento para hacer esta comprobación ha sido el siguiente: una vez se han recuperado los datos, se comprueba la version de *Android* del dispositivo y si ese necesario si ya han sido concedidos los permisos, en el caso que ya estuviesen concedidos, se muestran las tareas ordenadas, pero si no lo estuviesen se muestra un *dialog* para solicitarlos y la respuesta se maneja sobrescribiendo el método *onRequestPermissionsResult*. En función de la respuesta se muestran las tareas ordenadas o no.

```
@Override
public void onRequestPermissionsResult(int requestCode, String[] permissions, int[] grantResults) {
    if (requestCode == MY_PERMISSIONS_REQUEST_ACCES_FINE_LOCATION) {
        if (grantResults.length > 0
                && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
            mostrarTareas(true);
        } else {
            mostrarTareas(false);
        }
    }
}
```

```
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
                if (MainActivity.this.checkSelfPermission(Manifest.permission.ACCESS_FINE_LOCATION)
                      != PackageManager.PERMISSION_GRANTED && checkSelfPermission(Manifest.
                     permission.ACCESS_COARSE_LOCATION) != PackageManager.PERMISSION_GRANTED) {
                    if (ContextCompat.checkSelfPermission(MainActivity.this, Manifest.permission.
                         ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED) {
                        if (ActivityCompat.shouldShowRequestPermissionRationale(MainActivity.this,
                             Manifest.permission.ACCESS_FINE_LOCATION)) {
                            Resources resources = MainActivity.this.getResources();
                            Toast.makeText(MainActivity.this, resources.getText(R.string.
                                permisoUbicacion), Toast.LENGTH_LONG).show();
                            mostrarTareas(false);
                        } else {
                            ActivityCompat.requestPermissions(MainActivity.this, new String[]{
                                Manifest.permission.ACCESS_FINE_LOCATION},
                                MY_PERMISSIONS_REQUEST_ACCES_FINE_LOCATION);
                        }
                    }
                } else {
                    mostrarTareas(true);
                }
            }
```

```
    private void mostrarTareas(boolean ordenar) {
        if (tareas.size() > 0) {
            if (ordenar) {
                fusedLocationProviderClient.getLastLocation().addOnCompleteListener(this, new
                     OnCompleteListener<Location>() {
                    @Override
                    public void onComplete(@NonNull Task<Location> task) {
                        Location loc = task.getResult();
                        if (loc != null) {
                            tareas.sort(new OrdenarParcelas(loc));
                            TareaAdapter adapter = new TareaAdapter(MainActivity.this, tareas);
                            listaTareas.setDividerHeight(0);
                            listaTareas.setAdapter(adapter);
                        } else {
                            TareaAdapter adapter = new TareaAdapter(MainActivity.this, tareas);
                            listaTareas.setDividerHeight(0);
                            listaTareas.setAdapter(adapter);
                            Toast.makeText(getApplicationContext(), "No se ha podido obtener la ubucacion.",
                                 Toast.LENGTH_LONG).show();
                        }
                    }
                });
            }else{
                TareaAdapter adapter = new TareaAdapter(MainActivity.this, tareas);
                listaTareas.setDividerHeight(0);
                listaTareas.setAdapter(adapter);
            }
        } else {
            listaTareas.setVisibility(View.GONE);
            noHayTareas.setVisibility(View.VISIBLE);
        }
    }
```

Capítulo 6

Pruebas

6.1. Pruebas sobre la API REST

Para realizar estas pruebas se han definido las peticiones en la herramienta *Postman* agrupándolas según los diferentes roles que aparecen en el sistema.

6.1.1. Pruebas comunes a todos los usuarios

Login

Peticiones lanzadas contra la URL */api/public/login*

Descripción	Operación	Cuerpo	Código esp.	Cuerpo esp.	Resultado
Login correcto	POST	Username y contraseña	200	Token	OK
Login incorrecto	POST	Username y contraseña	403	Detalles del error	OK

Tabla 6.1: Pruebas *login*.

Pruebas de seguridad

Pruebas relacionadas con el RNF-3.

Descripción	Operación	Cuerpo	Código esp.	Cuerpo esp.	Resultado
Petición con token a una URL permitida para el usuario	*	*	Distinto a 403	*	OK
Petición sin token a una URL permitida para el usuario	*	*	403	"Access denied"	OK
Petición con token a una URL no permitida para el usuario	*	*	403	"Access denied"	OK
Petición sin token a una URL no permitida para el usuario [1]	*	*	403	"Access denied"	OK

Tabla 6.2: Pruebas seguridad.

Una vez comprobado estos puntos, todas las peticiones se harán con token y a URLs permitidas para el usuario.

[1]Esta prueba es similar a realizar una petición sin token a una URL permitida porque el sistema no puede comprobar que usuario está lanzando la petición. Por lo tanto, la segunda y cuarta prueba se podrían unir en Petición sin token a cualquier URL (permitida o no).

6.1.2. Pruebas relacionadas con las operaciones de los agricultores

Registro de un agricultor

Pruebas relacionadas con los requisitos funcionales 2. Peticiones lanzadas contra la URL /api/public/agricultores.

Descripción	Operación	Cuerpo	Código esp.	Cuerpo esp.	Resultado
Registro de un agricultor	POST	Usuario agricultor	201	Usuario agricultor	OK
Registro de un agricultor con alguno de los campos obligatorios nulo	POST	Usuario agricultor con algún campo obligatorio nulo	400	Mensaje de error	OK
Registro de un agricultor con un correo ya existente	POST	Usuario agricultor	400	Mensaje de error	OK

Tabla 6.3: Pruebas registro agricultor.

Gestión de parcelas

Pruebas relacionadas con los requisitos funcionales 14,15,16 y 17.
Peticiones lanzadas contra /api/private/agricultores/idAgricultor/parcelas o contra /api/private/agricultores/idAgricultor/parcelas/idParcela

Descripción	Operación	Cuerpo	Código esp.	Cuerpo esp.	Resultado
Registrar una parcela	POST	Parcela	201	Parcela	OK
Registrar una parcela con ubicación	POST	Parcela	201	Parcela	OK
Registrar una parcela con alguno de los campos obligatorios nulo	POST	Parcela con algún campo obligatorio nulo	400	Mensaje de error	OK
Ver parcelas registradas	GET	Vacío	200	Array de parcelas	OK
Ver una parcela en concreto	GET	Vacío	200	Parcela	OK

Tabla 6.4: Pruebas gestión parcelas.

Gestión de aperos

Pruebas relacionadas con los requisitos funcionales 19,20,21.
Peticiones lanzadas contra las URLs /api/private/agricultores/idAgricultor/aperos o contra /api/private/agricultores/idAgricultor/aperos/idApero.

Descripción	Operación	Cuerpo	Código esp.	Cuerpo esp.	Resultado
Registrar un apero	POST	Apero	201	Apero	OK
Registrar un apero con matrícula	POST	Apero	201	Apero	OK
Registrar un apero con algún campo obligatorio nulo	POST	Apero con algún campo nulo	400	Mensaje de error	OK
Recuperar aperos registrados	GET	Vacío	200	Array de aperos	OK
Ver los detalles de algún apero	GET	Vacío	200	Apero	OK

Tabla 6.5: Pruebas gestión de aperos.

Gestion de la agenda

Pruebas relacionadas con los requisitos funcionales 22 y 23. Peticiones lanzadas contra las URLs /api/private/agricultores/id/registrosAgenda o /api/private/agricultores/id/registrosAgenda/idRegistro

Descripción	Operación	Cuerpo	Código esp.	Cuerpo esp.	Resultado
Se añade un usuario a la agenda	POST	Usuario	204	-	OK
Se añade un usuario que ya estaba en la agenda	POST	Usuario	400	Mensaje de error	OK
Se elimina un usuario de la agenda	DELETE	-	204	-	OK
Se elimina un usuario de la agenda que no estaba en ella	DELETE	-	400	Mensaje de error	OK
Se recuperan los usuarios de la agenda	GET	-	200	Array de usuarios	OK

Tabla 6.6: Pruebas gestión de la agenda.

Gestión de tareas

Pruebas relacionadas con los requisitos funcionales 3,4,5,6,7,8,9,10,12. Las peticiones han sido lanzadas a las URLS /api/private/agricultores/id/parcelas/idParcela/tareas o /api/private/agricultores/id/parcelas/idParcela/-tareas/idTarea.

Descripción	Operación	Cuerpo	Código esp.	Cuerpo esp.	Resultado
Registrar una tarea	POST	Tarea	201	Tarea	OK
Registrar una tarea con algún campo obligatorio nulo	POST	Tarea con algún campo obligatorio nulo	400	Mensaje de error	OK
Registrar una tarea que necesita producto	POST	Tarea	201	Tarea	OK
Registrar una tarea que necesita producto sin indicar el producto	POST	Tarea sin producto	400	Mensaje de error	OK
Estimar una tarea	PUT	Tarea	200	Tarea	OK
Estimar una tarea que necesita producto	PUT	Tarea	200	Tarea	OK
Estimar una tarea que necesita producto sin indicar el producto	PUT	Tarea	400	Mensaje de error	OK
Completar una tarea	PUT	Tarea	200	Tarea	OK
Completar una tarea sin indicar las horas empleadas	PUT	Tarea	400	Mensaje de error	OK
Completar una tarea que necesita producto	PUT	Tarea	200	Tarea	OK
Completar una tarea que necesita producto sin indicar el producto	PUT	Tarea sin indicar el producto	400	Mensaje de error	OK
Completar una tarea que recoge producto	PUT	Tarea	200	Tarea	OK
Completar una tarea que recoge producto sin indicar el producto	PUT	Tarea sin indicar el producto	400	Mensaje de error	OK
Eliminar una tarea con estado pendiente de estimación	DELETE	-	-	-	OK
Eliminar una tarea que no existe	DELETE	-	400	Mensaje de error	OK
Eliminar tarea con estado estimada con fecha de inicio mayor de dos días de la fecha de hoy	DELETE	-	-	-	OK
Eliminar tarea con estado estimada con fecha de inicio menor de dos días de la fecha de hoy	DELETE	-	400	Mensaje de error	OK

Tabla 6.7: Pruebas gestión de tareas. Parte 1.

Descripción	Operación	Cuerpo	Código esp.	Cuerpo esp.	Resultado
Cambiar una tarea con estado pendiente de estimación a estimada	PUT	Tarea	400	Mensaje de error	OK
Obtener todas las tareas de una parcela	GET	-	200	Un array de tareas	OK
Obtener las tareas de un agricultor (bien creadas por él, bien es responsable de ellas)	GET	-	200	Un array de tareas	OK

Tabla 6.8: Pruebas gestión de tareas. Parte 2

6.1.3. Pruebas relacionadas con las operaciones de los responsables de cooperativas

Registro de un responsable de cooperativa

Pruebas relacionadas con el requisito funcional 2.
Peticiones lanzadas contra la URL /api/public/cooperativas.

Descripción	Operación	Cuerpo	Código esp.	Cuerpo esp.	Resultado
Registro de un responsable de cooperativa	POST	Usuario responsable de cooperativa	201	Usuario responsable de cooperativa	OK
Registro de un responsable de cooperativa con algún campo obligatorio nulo	POST	Usuario responsable de cooperativa con un campo obligatorio nulo	400	Mensaje de error	OK
Registrar un usuario de cooperativa con un correo que ya está en el sistema	POST	Usuario responsable de cooperativa	400	Mensaje de error	OK

Tabla 6.9: Pruebas registro cooperativa.

Gestión de productos

Pruebas relacionadas con los requisitos funcionales 29,30,31.
Peticiones lanzadas contra las URLs /api/private/cooperativas/id/productos o /api/private/cooperativas/id/productos/idProducto.

Descripción	Operación	Cuerpo	Código esp.	Cuerpo esp.	Resultado
Registrar un nuevo producto	POST	Producto	201	Producto	Fallo: no se comprobaba que se indicase el precio del producto.
Registrar un nuevo producto con algún campo obligatorio nulo	POST	Producto	400	Mensaje de error	OK
Recuperar productos de una cooperativa	GET	-	200	Array de productos	OK
Recuperar un producto de una cooperativa	GET	-	200	Producto	OK
Eliminar un producto de una cooperativa	DELETE	-	204	-	OK
Eliminar un producto de una cooperativa que no existe	DELETE	-	400	Mensaje de error	OK

Tabla 6.10: Pruebas gestión de productos.

Gestión de precios de los productos

Peticiones lanzadas contra las URLs /api/private/cooperativas/id/productos o /api/private/cooperativas/id/-productos/idProducto.

Descripción	Operación	Cuerpo	Código esp.	Cuerpo esp.	Resultado
Registrar un precio de un producto	POST	Precio producto	201	Precio producto	OK
Registrar un precio de un producto con algún campo obligatorio nulo	POST	Precio producto	400	Mensaje de error	OK

Tabla 6.11: Pruebas gestión precios de productos.

6.1.4. Pruebas relacionadas con las operaciones del administrador

Pruebas relacionadas con los requisitos funcionales 24,25,26.
Peticiones lanzadas contra las URLs /api/private/usuarios/idUsuario

Descripción	Operación	Cuerpo	Código esp.	Cuerpo esp.	Resultado
Bloquear un usuario	PUT	Usuario	200	Usuario	OK
Desbloquear un usuario	PUT	Usuario	200	Usuario	OK

Tabla 6.12: Pruebas gestión de usuarios.

6.2. Pruebas front-end

6.2.1. Pruebas aplicación web

Caso de uso	Prueba realizada	Resultado esperado	Prueba superada
Registrarse como agricultor	Registro normal	Se muestra un mensaje de operación completada	OK
	Registro sin indicar algún campo obligatorio	Se muestra el campo que hay que rellenar	OK
	Registro con un correo electrónico que ya está en el sistema	Se muestra el error en el campo correspondiente	OK
	No se ha podido conectar con el servidor	El sistema muestra un error	OK
Registrarse como Responsable de cooperativa	Registro normal	Se muestra un mensaje de operación completada	OK
	Registro sin indicar algún campo obligatorio	Se muestra el campo que hay que rellenar	OK
	Registro con un correo electrónico que ya está en el sistema	Se muestra el error en el campo correspondiente	OK
	No se ha podido conectar con el servidor	El sistema muestra un error	OK
Identificarse en el sistema	Login correcto	Se carga la página *home*	OK
	Login incorrecto	Se muestra un mensaje de error	OK
	Alguno de los campos del formulario está vacío	Se muestra un error en el campo correspondiente	OK
	El correo electrónico no sigue el formato adecuado	Se muestra un error en el campo correspondiente	OK
	Identificarse en el sistema con una cuenta bloqueada	El sistema muestra un mensaje de error	OK
	No se ha podido conectar con el servidor	Se muestra un mensaje de error	OK

Tabla 6.13: Pruebas aplicación web. Parte 1.

Caso de uso	Prueba realizada	Resultado esperado	Prueba superada
Restablecer contraseña	El usuario introduce el correo electrónico y la nueva contraseña	El sistema muestra un mensaje de operación completada	OK
	El correo electrónico introducido no está en el sistema	El sistema muestra un mensaje de error	OK
	El usuario cancela	El sistema muestra la página de login	OK
	Error al conectar con el servidor	El sistema muestra un mensaje de error	OK
Ver parcelas registrardas	El usuario hace click en Parcelas	El sistema muestra las parcelas	OK
	El usuario hace click en Parcelas >Mapa	El sistema muestra las parcelas con ubicación en un mapa	OK
	No se ha podido conectar con el servidor	El sistema muestra un mensaje de error	OK
Registrar parcela	Registrar parcela sin ubicación	Se muestra un mensaje de operación completada	OK
	Registrar parcela con ubicación	Se muestra un mensaje de operación completada	OK
	Registrar parcela con alguno de los campos obligatorios vacíos	Se muestra el campo que hay que rellenar	OK
	Se hace click en Cancelar	Se muestra un mensaje indicando que no se ha completado la operación	OK
	Error al conectar con el servidor	Se muestra un mensaje de error	OK
Ver detalles parcela	El usuario hace click en ver detalles de una parcela en la lista	El sistema le muestra los detalles de la parcela	OK
	El usuario hace click en un marcador del mapa	El sistema le muestra los detalles de la parcela	OK
	Error al conectar con el servidor	El sistema muestra un mensaje de error	OK
Ver aperos registrados	El usuario hace click en Aperos	El sistema muestra los aperos	OK
	No se ha podido conectar con el servidor	El sistema muestra un mensaje de error	OK

Tabla 6.14: Pruebas aplicación web. Parte 2.

Caso de uso	**Prueba realizada**	**Resultado esperado**	**Prueba superada**
Registrar nuevo apero	Registro completo	Se añade un nuevo apero	OK
	Registro de un nuevo apero con matrícula	Se añade un nuevo apero	OK
	La matrícula no sigue el patrón indicado	Se muestra un error en el campo correspondiente	OK
	Se intenta guardar un con alguno de los campos obligatorios nulo	El sistema muestra un error en el campo correspondiente	OK
	No se puede conectar con el servidor	El sistema muestra un mensaje de error	OK
	Se cancela	El sistema muestra un mensaje indicando que no se ha completado la operación	OK
Ver detalles apero	El usuario hace click en el apero que quiere consultar	El sistema muestra los detalles del apero	OK
Unirse a una cooperativa	El usuario hace click en Cooperativa introduce el código	El sistema muestra los productos de la cooperativa	OK
	El código introducido no sigue el formato correspondiente	El sistema marca como inválido el campo correspondiente	OK
	El código no pertenece a ninguna cooperativa	El sistema muestra un mensaje de error	OK
Ver productos de una cooperativa	El usuario hace click en Cooperativa	El sistema muestra los productos de la cooperativa	OK
	No se puede conectar con el servidor	El sistema muestra un mensaje de error	OK
Ver usuarios de la agenda	El usuario hace click en Agenda	El sistema muestra los usuarios de la agenda	OK
	No se puede conectar con el servidor	El sistema muestra un mensaje de error	OK

Tabla 6.15: Pruebas aplicación web. Parte 3.

Caso de uso	Prueba realizada	Resultado esperado	Prueba superada
Añadir agricultor a la agenda	El usuario agrega un nuevo usuario a la agenda	El nuevo usuario se añade a la agenda	OK
	El actor usuario intenta agregar un usuario a la agenda sin indicar el nombre	El sistema marca como erróneo el campo correspondiente	OK
	El usuario cancela	El sistema muestra un mensaje indicando que no se ha completado la operación	OK
	El usuario intenta añadir un usuario que ya estaba en la agenda	El sistema muestra un mensaje de error	OK
	No se ha podido conectar con el servidor	El sistema muestra un mensaje de error	OK
Eliminar agricultor de la agenda	El usuario elimina un usuario	El sistema eliminar el usuario muestra un mensaje de operación completada.	OK
	No se ha podido conectar con el servidor	El sistema muestra un mensaje de error	OK
Ver tareas de las que es responsable y creadas por él	El usuario hace click en Tareas	El sistema muestra las tareas	OK
	No se ha podido conectar con el servidor	El sistema muestra un mensaje de error	OK
Eliminar una tarea	El usuario elimina una tarea con estado pendiente de estimación	El sistema elimina la tarea y muestra un mensaje de operación completada	OK
	El usuario elimina una tarea con estado estimada con fecha de inicio mayor al menos dos días al día de hoy	El sistema elimina la tarea y muestra un mensaje de operación completada	OK
	El sistema elimina una tarea con estado estimada con fecha de inicio con una diferencia menor a dos días del día de hoy	El sistema muestra un mensaje de error	OK
	No se ha podido conectar con el servidor	El sistema muestra un mensaje de error	OK

Tabla 6.16: Pruebas aplicación web. Parte 4.

Caso de uso	Prueba realizada	Resultado esperado	Prueba superada
Crear una tarea	El usuario completa el registro	El sistema crea una nueva tarea y muestra un mensaje de operación completada	OK
	El usuario selecciona como encargado un usuario distinto al suyo	El sistema crea una tarea asignando como encargado al otro usuario	OK
	El usuario crea una tarea cuya labor necesita producto	El sistema solicita que introduzca a mayores el nombre del producto y la unidad	OK
	El usuario registra una tarea sin indicar algún campo obligatorio	El sistema muestra un error en el campo correspondiente	OK
	El usuario marca como encargado un usuario que no dispone del apero para realizar ese tipo de labor	El sistema muestra un mensaje de error	OK
	El usuario cancela	El sistema muestra un mensaje de que no se ha podido completar la operación	OK
	No se ha podido conectar con el servidor	El sistema muestra un mensaje de error	OK
Estimar una tarea	El usuario estima la tarea	El sistema guarda los cambios y actualiza la tarea como estimada	OK
	Si el tipo de labor de la tarea necesita un producto, el sistema pide a mayores que se indique la cantidad estimada y precio por unidad del producto	El sistema pide que se indiquen estos campos	OK
	El usuario estima la tarea sin indicar algún campo obligatorio	El sistema marca como erróneo el campo correspondiente	OK
	El usuario selecciona un apero que no puede realizar el tipo de labor	El sistema marca como erróneo el campo correspondiente	OK
	El usuario cancela	El sistema muestra un mensaje indicando que no se ha podido completar la operación	OK
	No se ha podido conectar con el servidor	El sistema muestra un mensaje de error	OK

Tabla 6.17: Pruebas aplicación web. Parte 5.

Caso de uso	Prueba realizada	Resultado esperado	Prueba superada
Completar una tarea	El usuario completa una tarea	El sistema guarda los cambios y actualiza la tarea como completada	OK
	Si la tarea necesitaba productos, el sistema solicita que se indique la cantidad utilizada	El sistema solicita estos campos	OK
	Si la tarea recoge algún producto, el sistema solicita que se indique el nombre del producto, la unidad y la cantidad recogida	El sistema solicita estos campos	OK
	El usuario hace click en Generar hoja de costes	El sistema genera un archivo .pdf indicando los detalles de la tarea	OK
	El usuario completa la tarea sin indicar algún campo obligatorio	El sistema marca como erróneo el campo correspondiente	OK
	El usuario cancela	El sistema muestra un mensaje indicando que no se ha podido completar la tarea	OK
	No se ha podido conectar con el servidor	El sistema muestra un mensaje de error	OK
Actualizar una tarea	En una tarea delegada el encargado actualiza una tarea	La tarea se puede actualizar	OK
	En una tarea delegada, el que no es encargado intenta actualizarla	El sistema no permite que la tarea se actualice	OK
Generar código	El usuario hace click en Generar código	El sistema genera un código	OK
Ver productos	El usuario hce click en Productos	El sistema muestra los productos de esa cooperativa	OK
	No se ha podido conectar con el servidor	El sistema muestra un mensaje de error	OK
Añadir producto	El usuario añade un producto	El sistema crea un nuevo producto	OK
	El usuario registra un producto sin indicar alguno de los campos obligatorios	El sistema marca como erróneo el campo correspondiente	OK
	El usuario cancela	El sistema muestra un mensaje indicando que no se ha podido completar la operación	OK
	No se ha podido conectar con el servidor	El sistema muestra un mensaje de error	OK
Eliminar producto	El usuario elimina un producto	El sistema elimina el producto y muestra un mensaje de operación completada	OK
	No se ha podido conectar con el servidor	El sistema muestra un mensaje de error	OK

Tabla 6.18: Pruebas aplicación web. Parte 6.

Caso de uso	**Prueba realizada**	**Resultado esperado**	**Prueba superada**
Modificar precio producto	El usuario modifica el precio del producto	El sistema guarda los cambios y muestra un mensaje de operación completada	
	El usuario cancela	El sistema muestra un mensaje indicando que no se ha podido completar la operación	OK
	El actor modifica el precio sin indicar todos los campos obligatorios	El sistema marca como erróneo el campo correspondiente	OK
	No se ha podido conectar con el servidor	El sistema muestra un mensaje de error	OK
Ver listado usuarios	El usuario hace click en Usuarios	El sistema muestra un listado de los usuarios del sistema	OK
	Error al conectar con el servidor	El sistema muestra un mensaje de error	OK
Bloquear cuenta	El usuario bloquea una cuenta	El sistema actualiza los cambios	OK
	No se ha podido conectar con el servidor	El sistema muestra un mensaje de error	OK

Tabla 6.19: Pruebas aplicación web. Parte 7.

6.2.2. Pruebas aplicación Android

Caso de uso	Prueba realizada	Resultado esperado	Prueba superada
Identificarse	El usuario se identifica en el sistema	El sistema le muestra las tareas que tienes estimadas para el día de hoy	OK
	El usuario no introduce alguno de los campos	El sistema marca como erróneo el campo correspondiente	OK
	Las credenciales introducidas por el usuario son erróneas	El sistema muestra un mensaje de error	OK
	No se ha podido conectar con el servidor	El sistema muestra un mensaje de error	OK
Ver tareas	El usuario ha concedido permiso de acceso a la aplicación	La aplicación muestra las tareas ordenadas por distancia teniendo en cuenta la última ubicación conocida	OK
	El usuario no ha concedido permiso de ubicación a la aplicación	El sistema muestra las tareas	OK

Tabla 6.20: Pruebas aplicación Android.

Capítulo 7

Despliegue

En este capítulo se explicará cómo se ha realizado el despliegue de la aplicación.

7.1. Pasos que se han seguido

1. Una vez se haya decidido la máquina donde se hará el despliegue (en este caso una máquina de la Escuela) se instaló *Docker* y *Docker compose*.
2. Se definieron cinco nuevos archivos:
   - Dos archivos de configuración de las aplicaciones uno llamado *environment-prod.ts* en la aplicación *Angular* y otro llamado *application-prod.properties* en la aplicación *SpringBoot* con la configuración que deben seguir las aplicaciones en este entorno.
   - Dos archivos llamados *Dockerfile*, uno dentro del directorio donde se encuentra la aplicación del *back-end* y otro en el directorio donde se encuentra la aplicación *Angular* que se encargarán de la configuración de los contenedores de estas aplicaciones.
   - Un archivo llamado *docker-compose.yml* en el directorio *Codigo* que se encargará de las comunicaciones entre los tres contenedores (la aplicación *Angular*, el *back-end* y la base de datos).
3. Se generó un archivo *.jar* del *back-end* para ello en la carpeta se ejecutó el comando *./mvnw package -f ./pom.xml*.
4. Se copiaron las carpetas *Codigo/frontGestiontTareas*, *Codigo/gestiontareas* y el archivo *docker-compose.yml* a la máquina del primer paso.
5. Y por último se ejecutó el comando *docker-compose up* lo que los creará tres contenedores para que funcione tanto la aplicación web como el backend.

El resultado es el siguiente:

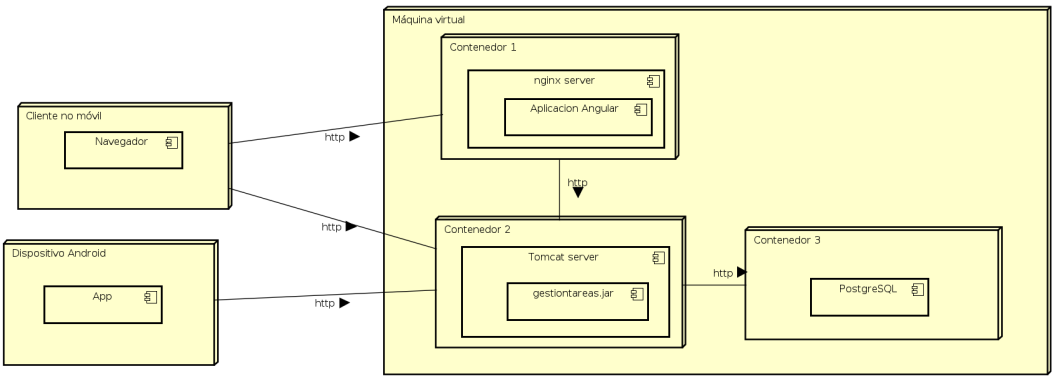

Figura 7.1: Diagrama de despliegue

Nota: si se cambia la máquina donde se hace el despliegue hay que modificar las URLs contra las que lanzan las peticiones las aplicaciones que forman el *front-end*.

Capítulo 8

Seguimiento del proyecto

8.1. Planificación del proyecto

Tras un primer estudio del proyecto este se estimó en 485 horas, sin embargo, en cuatro de las actividades que se definieron el tiempo que se estimó no ha coincidido/aproximado al que se ha dedicado en realidad. Estas cinco actividad son:

- *Diseño de las vistas*. Esta actividad se estimó en 10 horas, pero el tiempo total que se empleó en ella fue de 18 horas.
- *Implementación de la API REST*. Estimada en 30 horas cuando el tiempo total empleado ha sido de 36 horas.
- *Implementación de las vistas de la parte web* e *Implementación de la lógica de la parte web*. Estas dos actividades estaban estimadas en un total de 90 horas, pero el tiempo real que se empleó en realizar estas dos actividades ha sido de 120 horas.
- *Manual de usuario*. Esta actividad se estimó en 5 horas, pero el tiempo real que se ha dedicado ha sido de 12 horas.

En las fases que se ha conseguido dedicar menos tiempo del que se estimó en un principio se encuentran:

- *Implementación de la lógica de la aplicación*. Esta actividad se estimó en 60 horas pero el tiempo real que se ha dedicado ha sido de 30 horas.
- *Implementación del acceso a persistencia*. Esta actividad se estimó en 30 horas y su tiempo real ha sido de 12 horas.

Y para finalizar con esta revisión, sólo se definió una actividad dedicada a las pruebas del *back-end* pero no se tuvo en cuenta una actividad para las pruebas de las aplicaciones del *front-end* a las que se ha dedicado 10 horas.

En resumen:

Horas estimadas iniciales	485
Horas dedicadas a mayores	61
Horas que se han ahorrado	48
Horas totales	485 + 61 - 48 = 498 horas

Tabla 8.1: Horas dedicadas al finalizar el proyecto.

8.2. Funcionalidad del sistema

El proyecto no se ha dado por finalizado hasta que se ha implementado y probado toda la funcionalidad descrita en los primeros capítulos de esta además de haber realizado la documentación correspondiente. Por lo tanto, se considera que se ha cubierto el 100 % de la funcionalidad que se propuso al comienzo del proyecto y no ha sido necesario incluir ningún requisito más durante del desarrollo del proyecto.

8.3. Costes

Con el incremento de las horas necesitadas también se ha incrementado el coste. A mayores de los costes que se habían estimado en un principio (8790 €) hay que añadir 13 horas * 18 €= 234 €. Por lo tanto, el coste total del proyecto sería de 9024 €.

8.4. Riesgos

De los riesgos que se analizaron al comienzo del proyecto sólo ha sido necesario lidiar con uno *Uso de tecnologías y herramientas nuevas*. Durante el desarrollo de la aplicación *Angular*, concretamente la manera en la que se iba a implementar el mapa, se decidió que se iba a utilizar *OpenLayers* pero, ante los continuos problemas que presentaba esta solución se decidió buscar una nueva alternativa sustituyendo el desarrollo que se había hecho por una implementación que ofrecía *Google Maps*. También, durante el desarrollo del *back-end* surgieron dos problemas:

1. Al realizar peticiones a la API REST esta devolvía datos duplicados. Consultando la documentación del framework y otras fuentes ambos sugerían que, si no era necesario mantener un orden dentro de la estructura de datos, se sustituyesen los campos de tipos *List* por campos de tipo *Set*.

2. Cuando en la aplicación *Angular* se lanzaban peticiones de tipo *DELETE*, esta reintentaba la operación dando lugar a errores con código 404 porque la primera petición si que eliminaba el recurso que se quería y el resto no podían eliminarlo (el servidor buscaba un recurso que acababa de ser eliminado). La solución fue modificar el código de la respuesta a 204.

Capítulo 9

Conclusiones

9.1. Conclusiones

Con el desarrollo de este Trabajo de Fin de Grado se ha podido simular el desarrollo de un proyecto completo en el que se ha llegado a los siguientes resultados:

- Se ha realizado todo el proceso de ingeniería desde una idea inicial hasta el diseño del sistema.
- Se ha desarrollado una aplicación web para que los distintos usuarios puedan realizar las operaciones que se describieron al comienzo del proyecto.
- Se ha desarrollado una aplicación nativa para dispositivos Android para que los agricultores puedan saber que tareas tienen que realizar determinado día sin necesidad de tener un ordenador.
- Se ha desarrollado una aplicación para que las dos aplicaciones puedan obtener los datos.
- Se han desarrollado mecanismos de autenticación y autorización para que sólo los usuarios identificados en el sistema accedan a los datos de este y no todos los usuarios tengan acceso a todas las operaciones.
- Se han buscado tecnologías actuales y que permitan un futuro mantenimiento del sistema lo más sencillo posible.
- Se ha hecho una simulación de un despliegue para que el sistema pueda ser utilizado por los diferentes usuarios, aunque este no debería ser el despliegue final.

9.2. Trabajo futuro

Como primer punto que habría que tratar es buscar una forma de despliegue que realmente permita a los usuarios utilizar el sistema porque, como se ha ido comentando a lo largo de esta memoria, la aplicación está desplegada en una máquina virtual cedida por la Escuela.

Una vez resuelto este punto, todo proyecto software lleva relacionado un mantenimiento. Para este mantenimiento se han considerado tres puntos:

1. Correcciones en la aplicación: posibles fallos que se puedan encontrar con el uso de aplicación.
2. Gestión de los usuarios y el contenido: con el rol de administrador se pueden bloquear cuentas que no utilicen el sistema siguiendo la finalidad con la que se creó. Además, con un uso prologado del sistema y al crear contenido puede darse el caso, como podría ocurrir por ejemplo a la hora de cargar muchas tareas, que el tiempo de espera sea muy elevado. Una posible solución sería añadir procesos que se lanzasen con cierta periodicidad y que eliminasen tareas que cumpliesen determinadas condiciones en el caso del ejemplo anterior.
3. Nuevas funcionalidades: después de mostrarles la aplicación a varios usuarios que se ajustaban al perfil de usuario de la aplicación propusieron los siguientes cambios:

    a) El campo descripción no es demasiado útil y en el caso de las parcelas podría ser sustituido por los campos *polígono* y *parcela* que es cómo se identifican las parcelas de forma oficial.

b) Crear un registro de productos y que, cuando se cree una tarea y se especifique el tipo de labor que se quiere realizar, el sistema proponga productos que se puedan emplear para desempeñar esa tarea.

c) Para la aplicación móvil, en lugar de ordenar por cercanía, ordenar de formar que se pierda el menor tiempo posible en realizar el camino entre parcela y parcela. Además, este usuario propuso que la primera tarea que habría que mostrar es la que estuviese más lejos del punto de partida y la última la que estuviese más cerca del punto de partida.

Bibliografía

Android. *Android developers*. Última vez que se visitó: 20 de enero de 2020.
URL: `https://developer.android.com/`.

Angular. *Documentación Angular*. Última vez que se visitó: 16 de enero de 2020. URL: `https://angular.io/`.

— *Documentación Angular Material*. Última vez que se visitó: 3 de enero de 2020.
URL: `https://material.angular.io/`.

Docker. *Documentación Docker*. Última vez que se visitó: 10 de febrero de 2020.
URL: `https://docs.docker.com/`.

Google. *GoogleMaps pricing*. Última vez que se visitó: 16 de enero de 2020.
URL: `https://cloud.google.com/maps-platform/pricing/sheet?hl=es-419`.

Indeed. *Sueldo desarrollador software*. Última vez que se visitó: 20 de octubre de 2019.
URL: `https://www.indeed.es/salaries/desarrollador-de-software-Salaries`.

Larman, Craig. *Applying UML and patterns: an introduction to object-oriented analysis and design and interative development*. Prentice Hall PTR, 2007.

Microsoft. *Documentación Azure*. Última vez que se visitó: 5 de enero de 2020.
URL: `https://docs.microsoft.com/es-es/azure/`.

— *Documentación Visual Studio Code*. Última vez que se visitó: 29 enero de 2020.
URL: `https://code.visualstudio.com/docs`.

OpenLayers. *OpenLayers*. Última vez que se visitó: 12 de enero de 2020. URL: `https://openlayers.org/`.

restfulapi.net. *Códigos respuesta servicio REST*. Última vez que se visitó: 25 de diciembre de 2019.
URL: `https://restfulapi.net/http-status-codes/`.

Richardson, Leonard y Sam Ruby. *RESTful web services:* OReilly, 2007.

SpringBoot. *Spring Projects*. Última vez que se visitó: 25 de diciembre de 2019.
URL: `https://spring.io/projects/spring-boot`.

Anexos

Anexo A

Manual de usuario

Para acceder al sistema se puede acceder bien por un navegador o bien por la aplicación móvil.

A.1. Navegador

A.1.1. Operaciones comunes a todos los usuarios

Acceso

La URL del sistema es `http://virtual.lab.inf.uva.es:20042/`. Una vez que hemos accedido a ella lo primero que aparecerá será la siguiente pantalla.

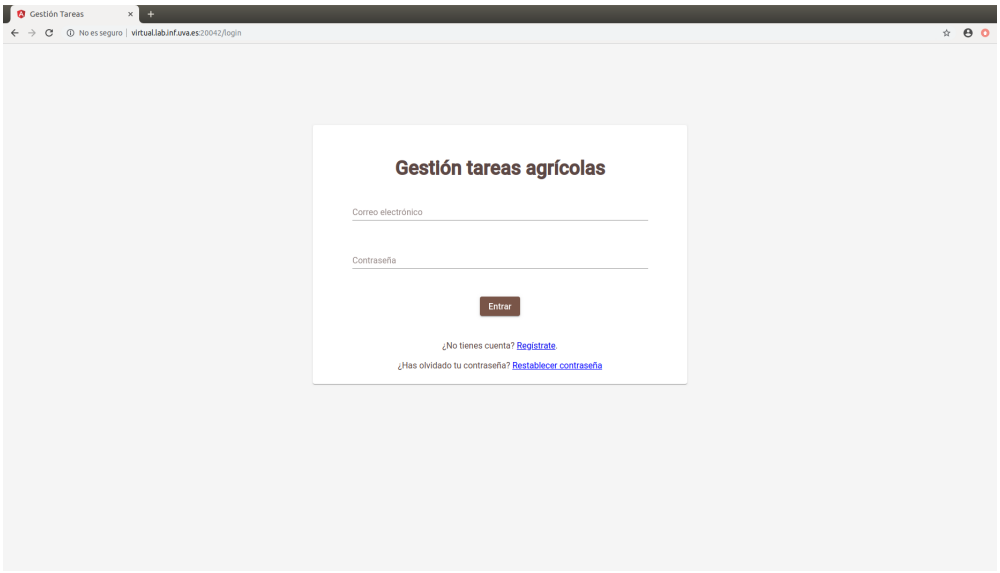

Figura A.1: Pantalla inicial

Identificarse en el sistema

Para identificarse en el sistema sólo hay que indicar el correo electrónico o la contraseña con las que se realizó el registro (si se tratase de un agricultor o un responsable de cooperativa). Una vez se hayan introducido en sus campos correspondientes, habrá que hacer click en *Entrar* y pueden suceder cuatro cosas diferentes:

- Las credenciales son correctas y el sistema carga la página *home* correspondiente a cada tipo de usuario.
- Las credenciales no son correctas y el sistema muestra el siguiente mensaje:
- La cuenta ha sido bloqueada por el administrador:
- No se ha podido conectar con el servidor

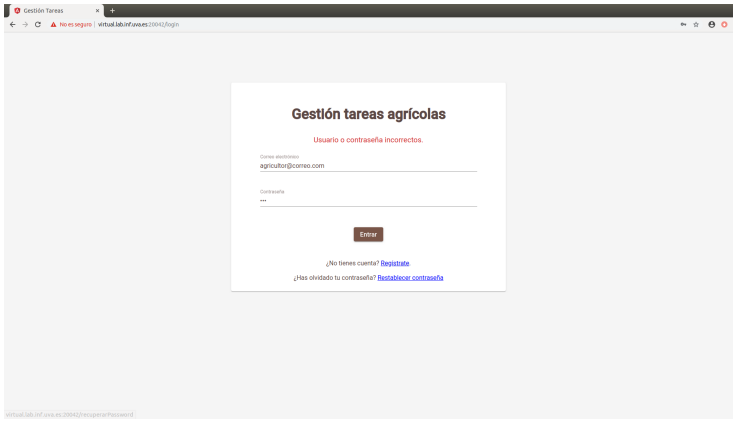

Figura A.2: Pantalla Identificarse en el sistema con unas credenciales incorrectas.

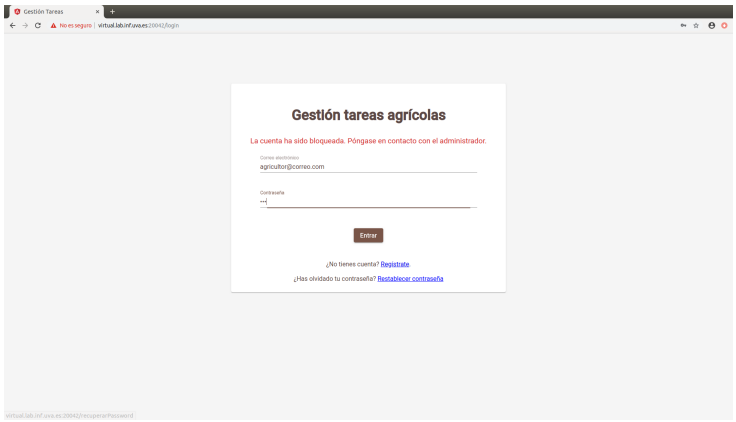

Figura A.3: Pantalla Identificarse en el sistema con una cuenta bloqueada.

Restablecer la contraseña

Si el usuario ha olvidado la contraseña o quiere establecer una contraseña nueva en primer lugar, desde la pantalla inicial, tendrá que hacer click en *Restablecer contraseña*.

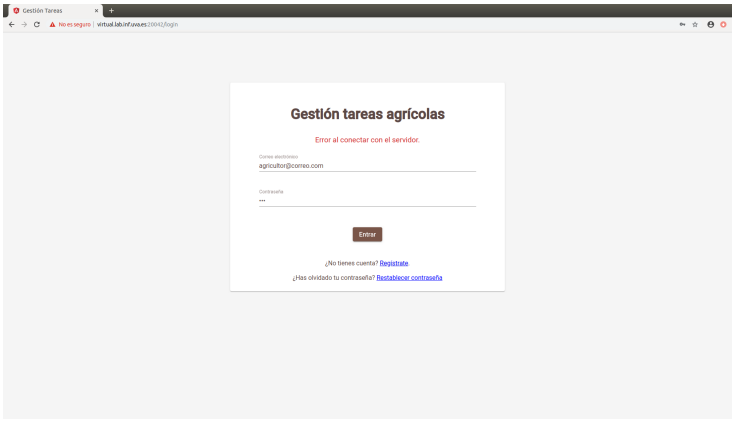

Figura A.4: Pantalla Identificarse en el sistema sin poder conectar con el servidor.

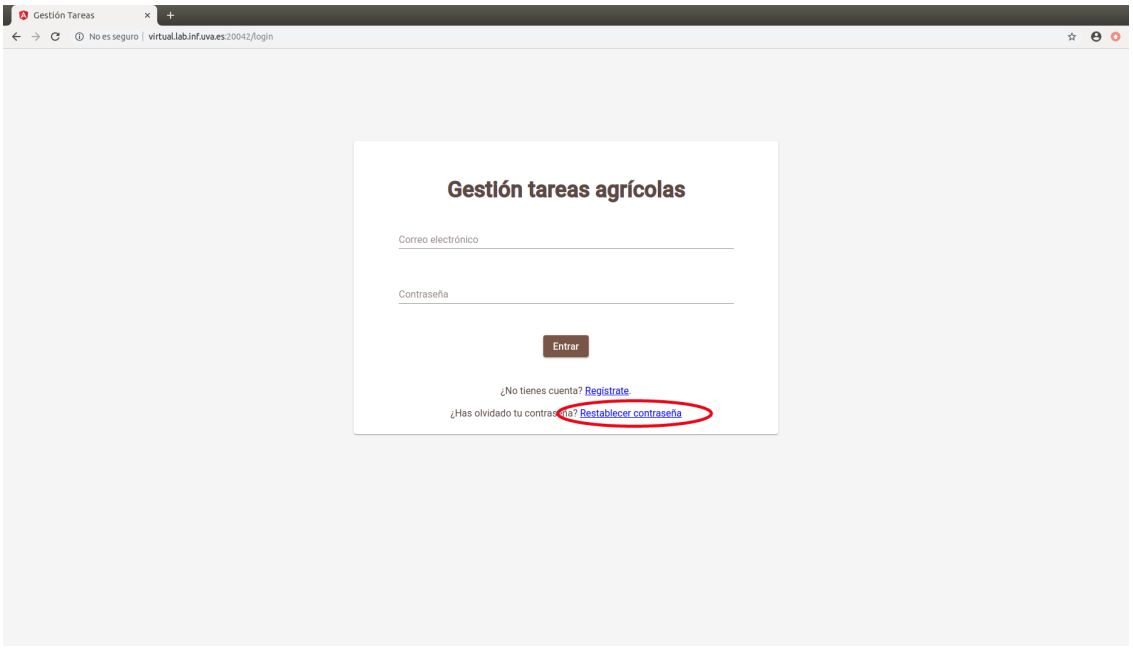

Figura A.5: Restablecer contraseña - Paso 1.

Este enlace cargará la siguiente página.

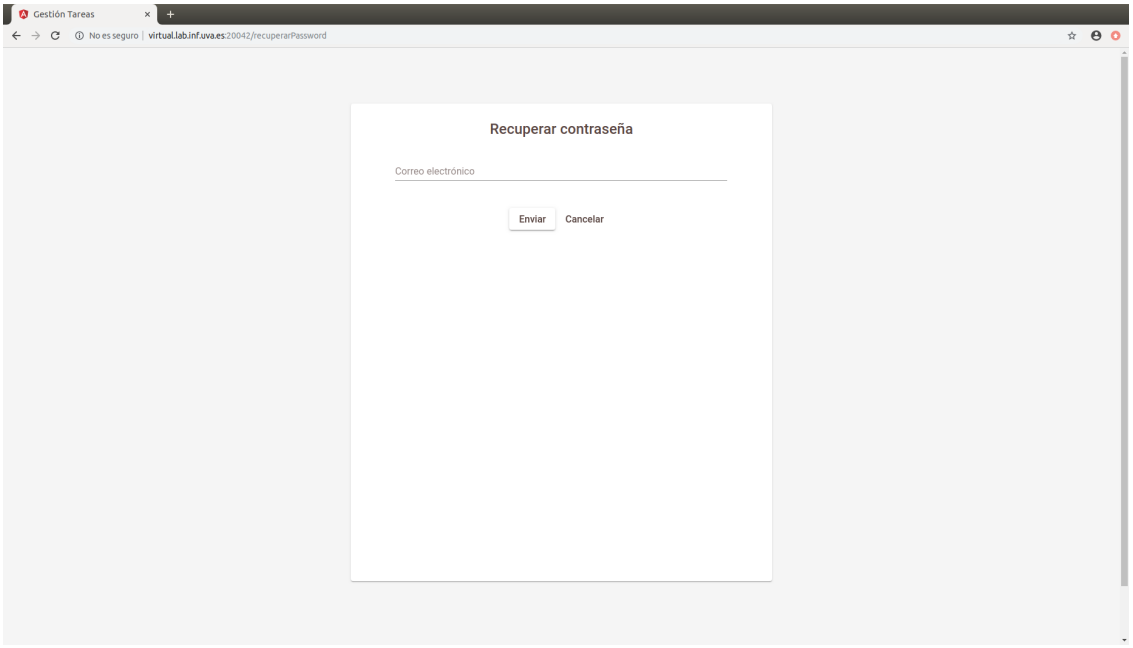

Figura A.6: Restablecer contraseña - Paso 2.

El usuario indicará su correo electrónico y hará click en Enviar. Si el correo electrónico está en el sistema, el sistema solicitará la nueva contraseña.

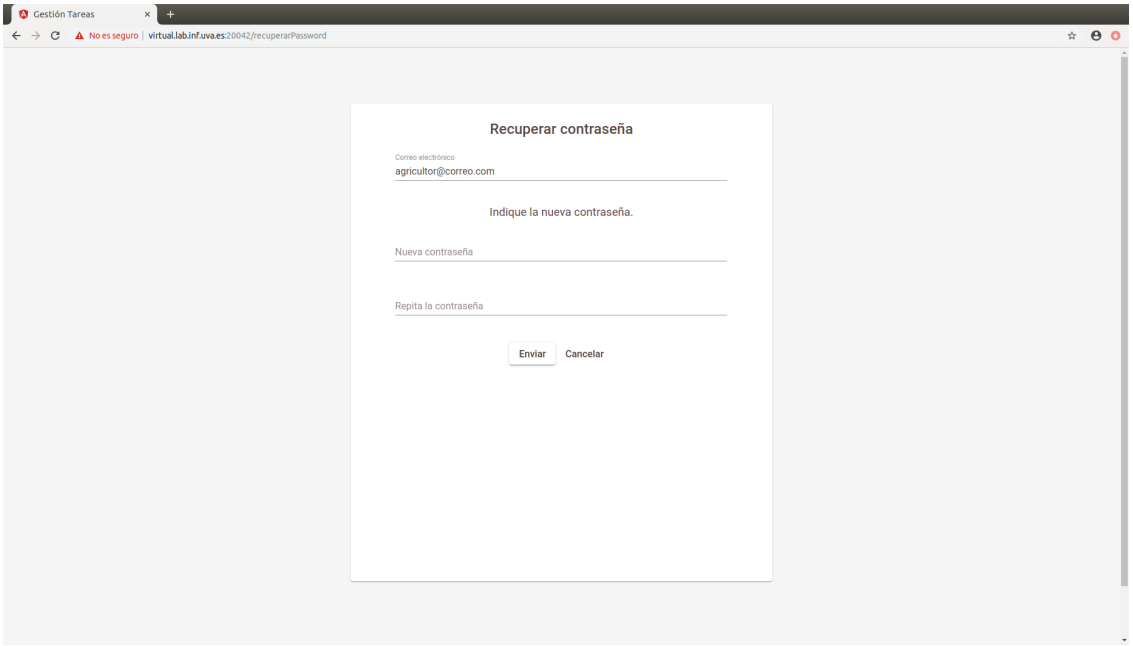

Figura A.7: Restablecer contraseña - Paso 3.

Una vez el usuario haya introducido la nueva contraseña, hará click en Enviar lo que actualizará la contraseña. Si ocurriese algún problema, la aplicación mostrará un mensaje de error.

A.1.2. Registros en el sistema

Registrarse como agricultor

Para poder acceder a la funcionalidad del sistema los agricultores tendrán que realizar un registro. Para hacer este registro hay que seguir los siguientes pasos. Desde la pantalla principal hacer click en Registrarse.

Una vez haya cargado la página y con la pestaña Agricultor de la parte superior seleccionada introducir los datos y una vez que el formulario este completo, hacer click en Registrarse. Si el registro se ha realizado de forma correcta, hacer click en Ok para que el sistema nos redirija a la página principal para poder acceder al sistema.

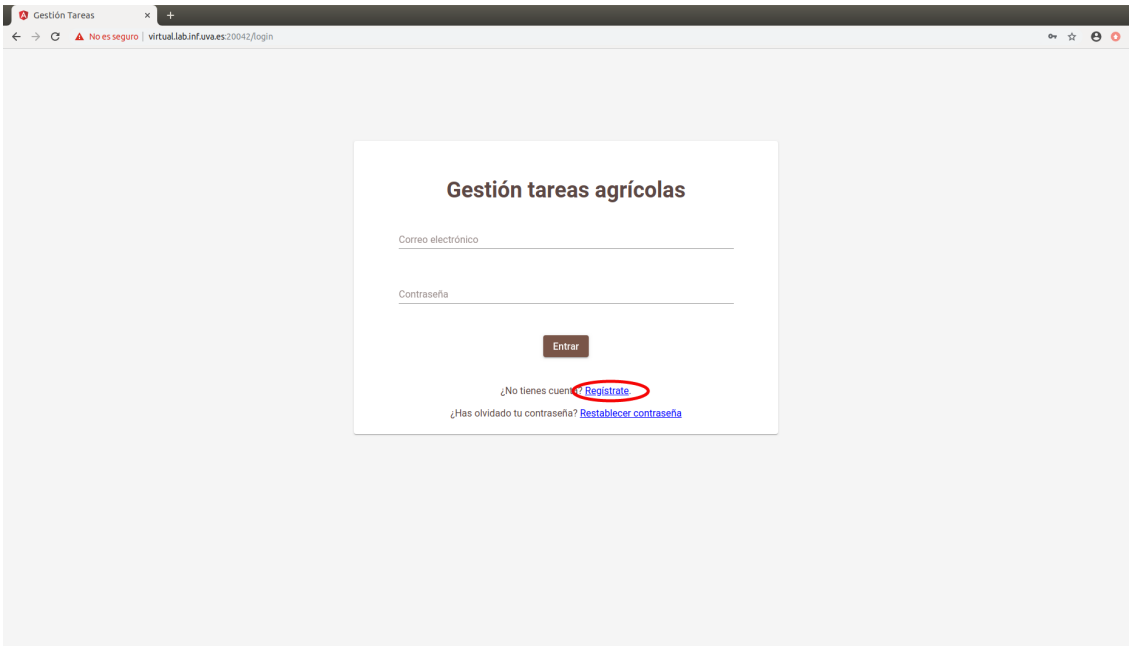

Figura A.8: Enlace al formulario para registrarse en el sistema.

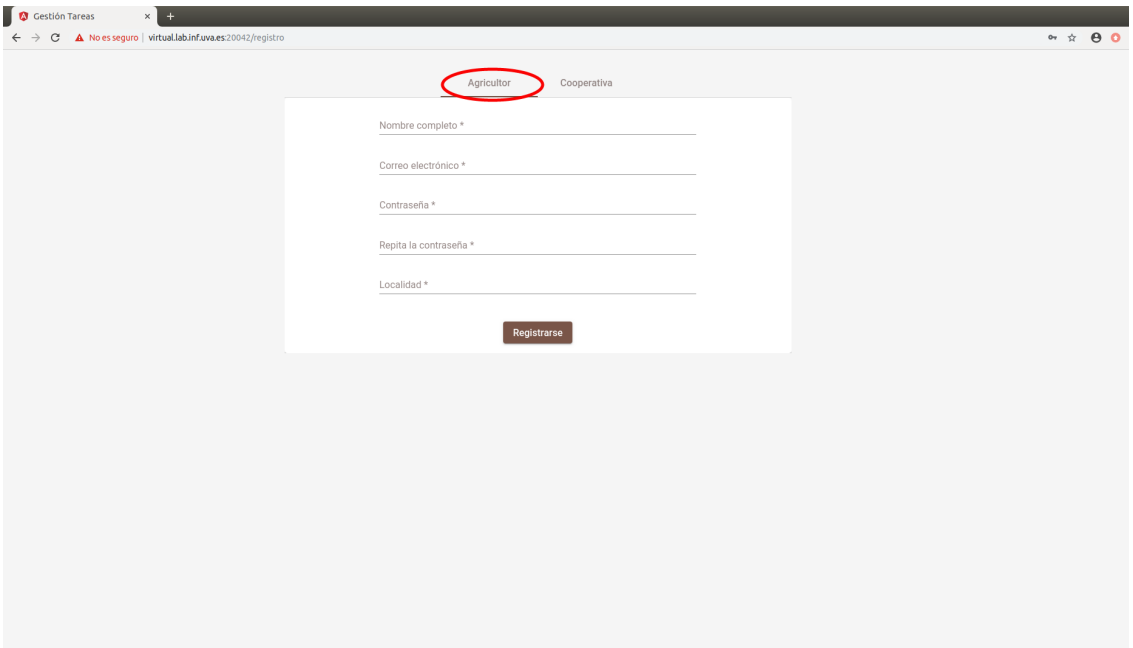

Figura A.9: Formulario registro.

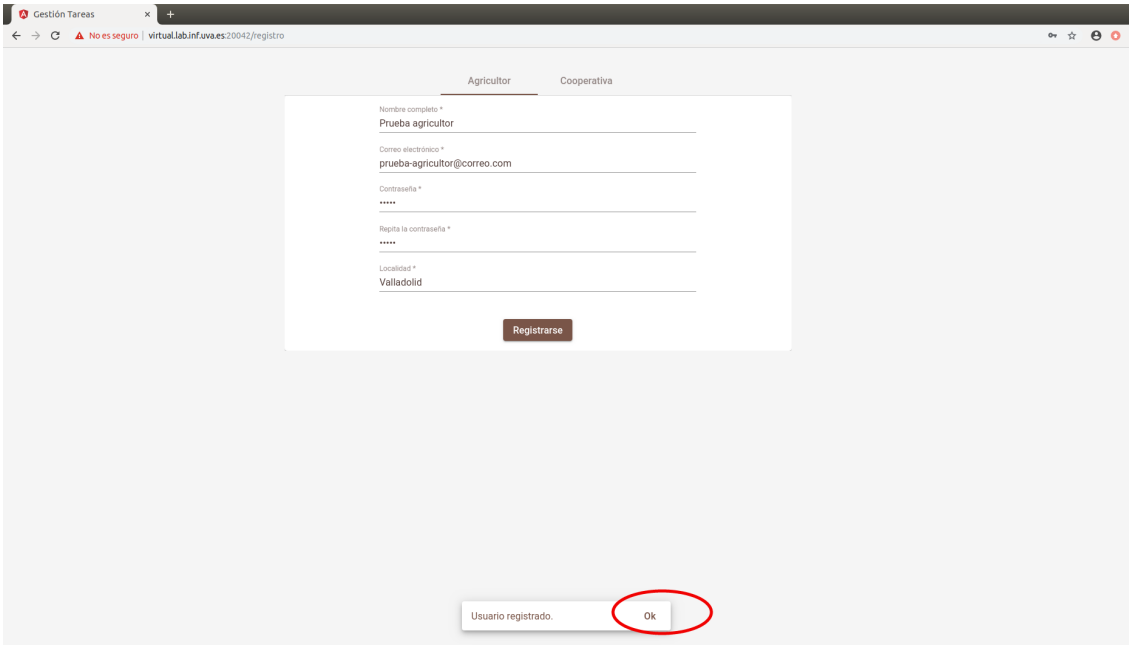

Figura A.10: Registro completado.

Registrarse como cooperativa

Para registrarse como cooperativa lo primero que hay que hacer es acceder al sistema. Una vez la página haya cargado, habrá que hacer click en Registrarse lo que cargará el siguiente formulario.

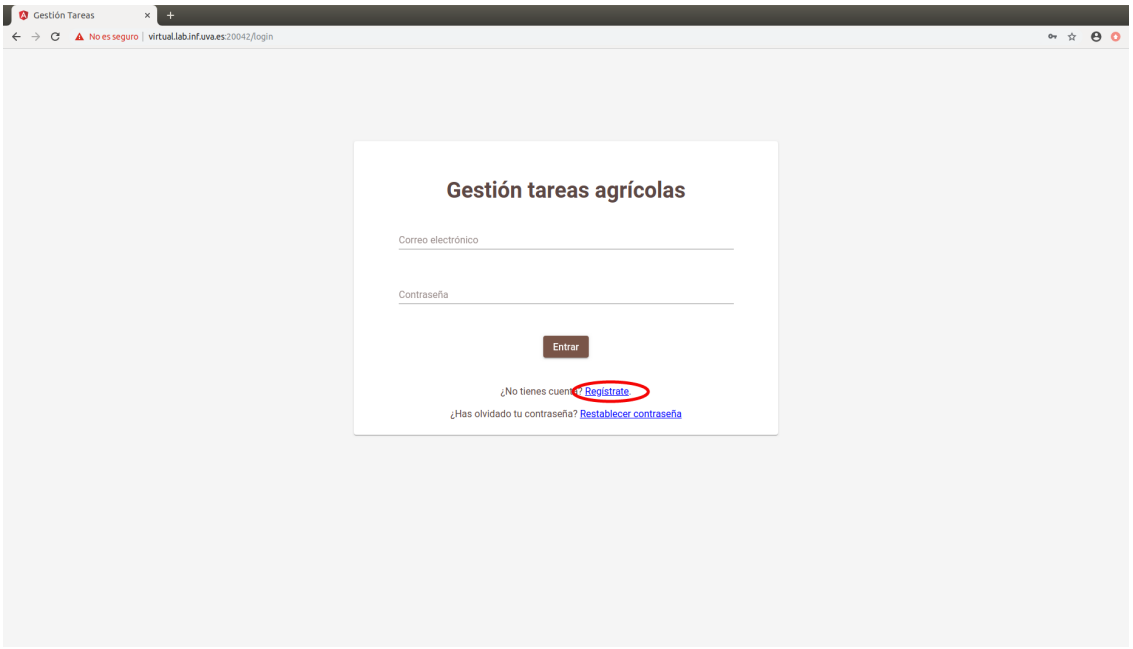

Figura A.11: Enlace al formulario para registrarse en el sistema.

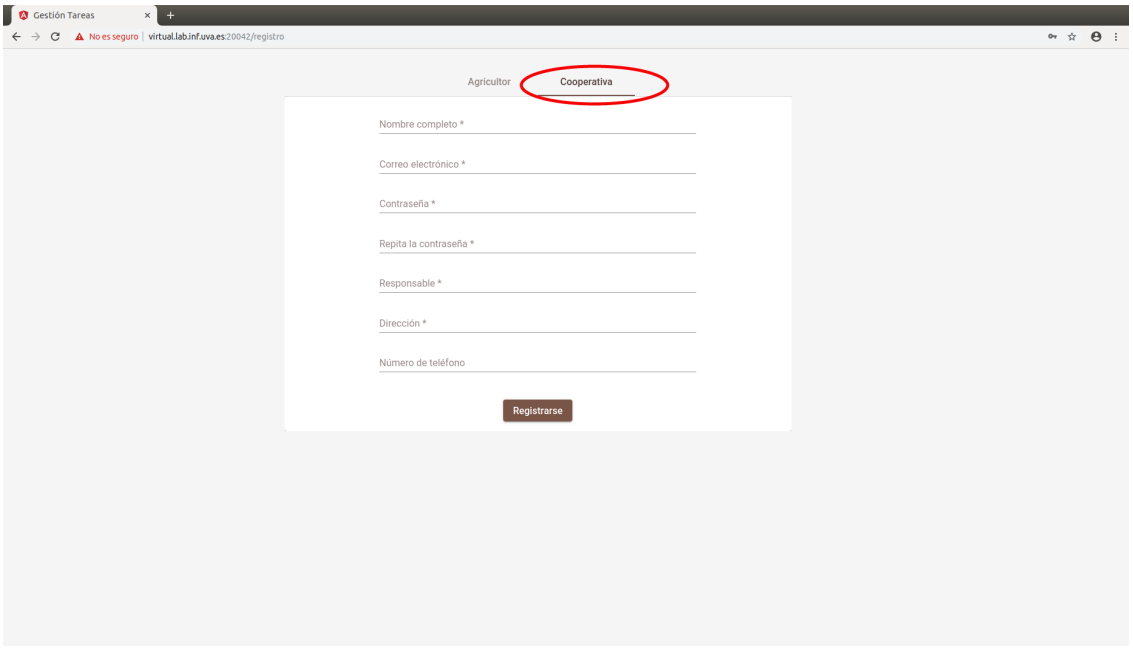

Figura A.12: Formulario registrarse como cooperativa.

Con la pestaña Cooperativa marcada, rellenar el formulario y una vez que estén todos los datos, hacer click en Registrarse.
Si el sistema ha registrado al usuario correctamente, el sistema mostrará un mensaje de operación completada que al hacer click en Ok nos redirigirá a la página inicial.

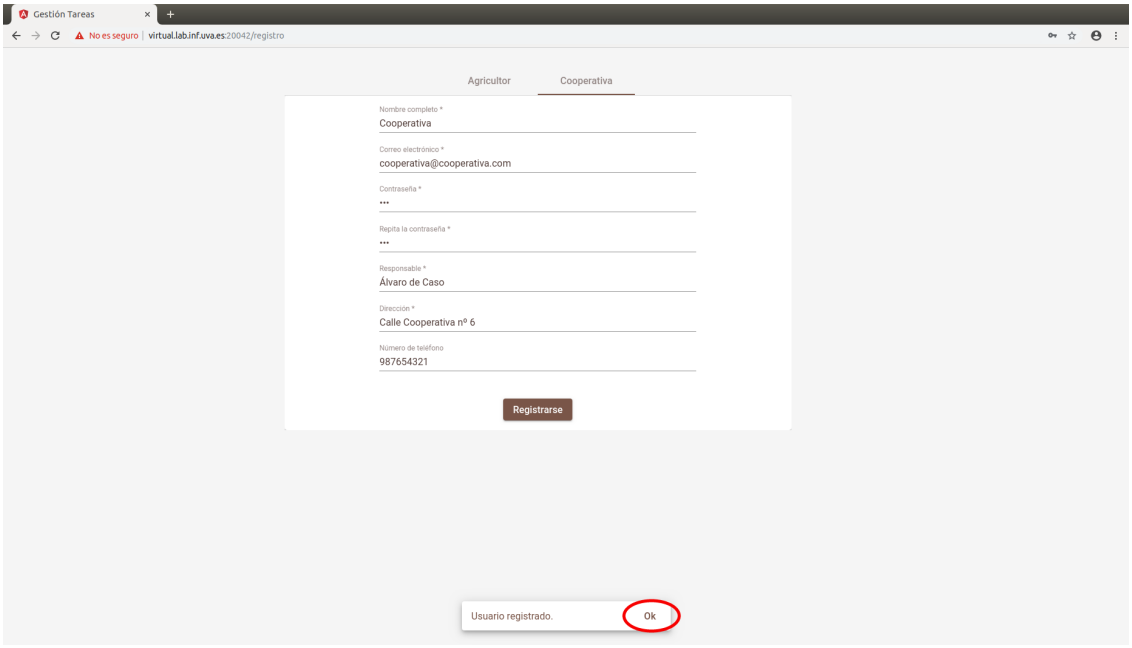

Figura A.13: Cooperativa registrada.

A.1.3. Operaciones relacionadas con los agricultores

Primera página

Una vez que un usuario se haya identificado de forma exitosa en el sistema, la página que se cargará será la siguiente.

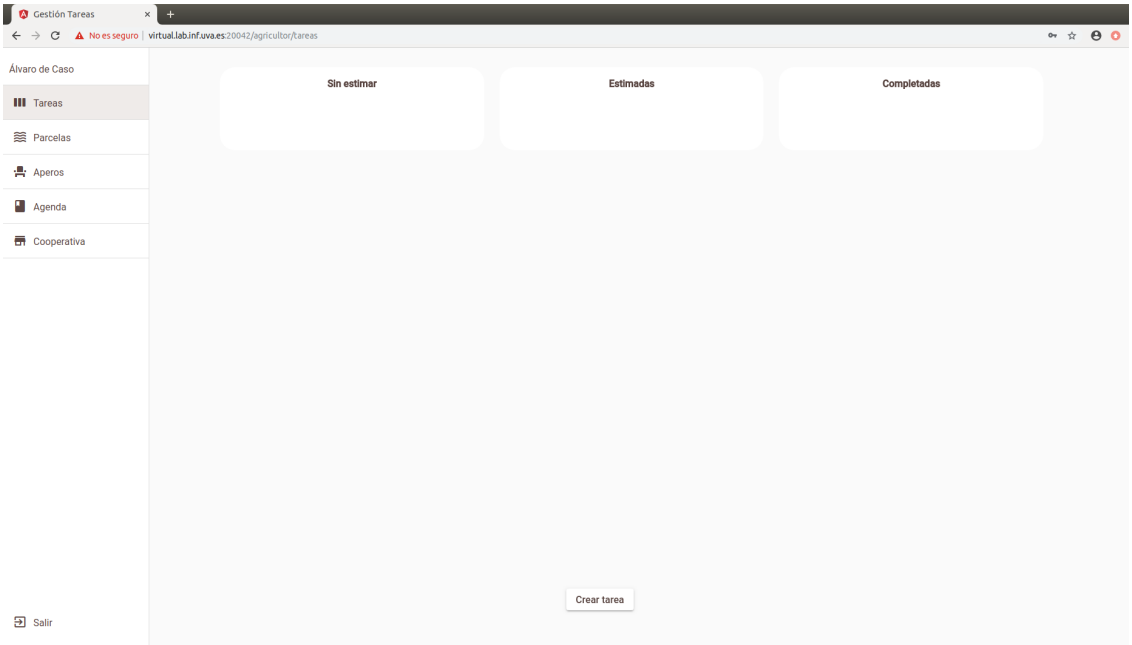

Figura A.14: Página principal de un agricultor

En el menú de la izquierda podemos observar los siguientes botones:

- *Tareas* para cargar la página que se utilizará para la gestión de las tareas. Esta es la opción que se carga por defecto cuando un agricultor se identifica en el sistema.
- *Parcelas* para cargar la página que se usará para gestionar las parcelas.
- *Aperos* para cargar la página que se usará para gestionar los aperos.
- *Agenda* para cargar la página que se usará para gestionar la agenda.

- *Cooperativa* para cargar la página dónde se podrá unir a una cooperativa o consultar los precios si el usuario ya forma parte de alguna.

- *Salir* en la parte inferior izquierda de la pantalla, para cerrar sesión el sistema y volver a la página inicial (Figura 7.1).

Ver parcelas registradas

Para ver las parcelas que un usuario tiene registradas en el sistema sólo tiene que hacer click en Parcelas. Esto mostrará todas las parcelas que el usuario ha registrado, pero también existe la opción de ver las parcelas que se ha indicado ubicación en un mapa. Para ello hay que hacer click en Mapa ubicado en la parte superior de la pantalla.

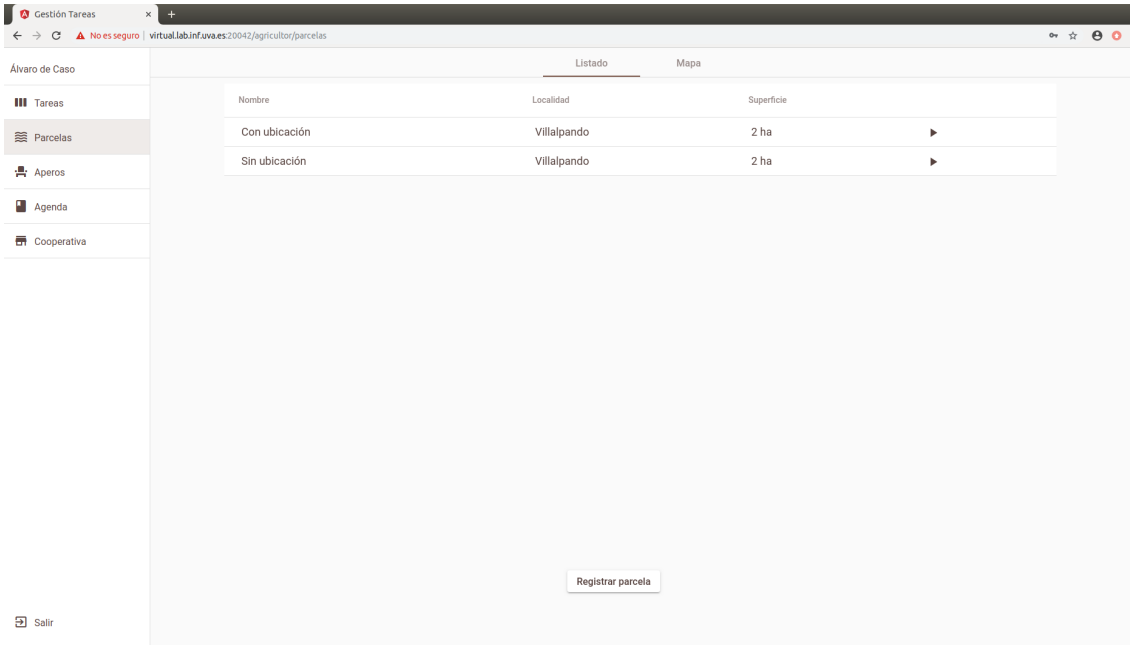

Figura A.15: Ver parcelas lista.

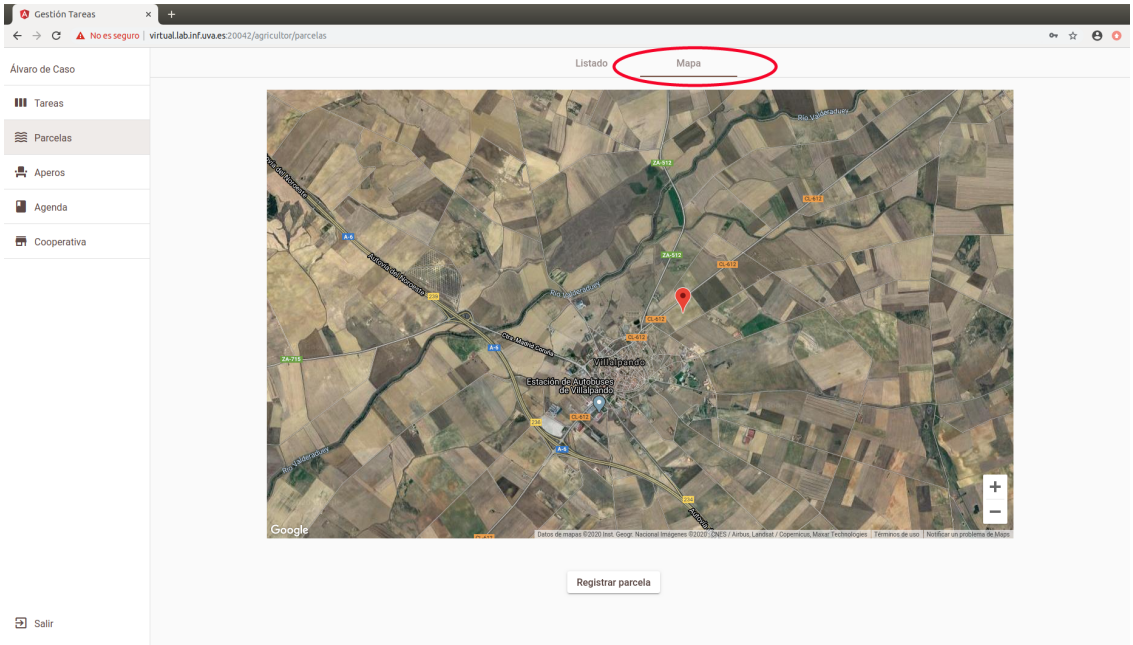

Figura A.16: Ver parcelas en el mapa.

Ver detalles de una parcela

Al igual que ver las parcelas registradas, ver los detalles de una parcela también se puede hacer de dos formas:

- Para todas las parcelas sus detalles se pueden ver desde la lista (Figura *Ver parcelas lista*) haciendo click en la flecha de la cuarta columna de la tabla

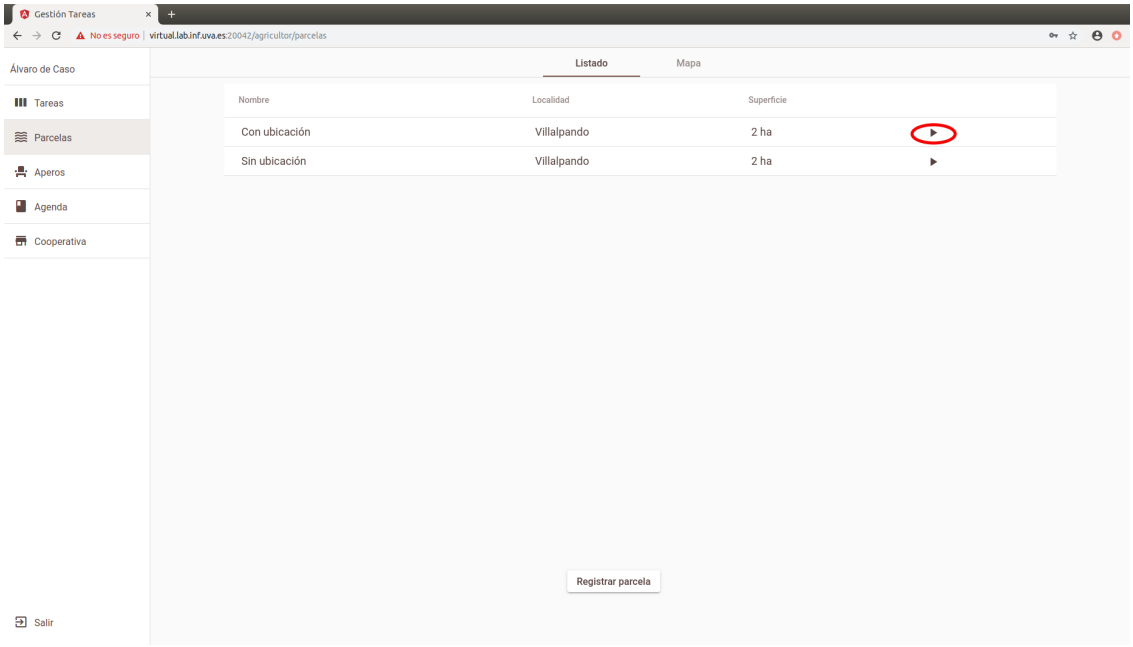

Figura A.17: Ver detalles parcela en la lista.

- Para las parcelas que tienen ubicación sus detalles se pueden ver tanto desde la lista como desde el mapa haciendo click en el icono que marca la ubicación de la parcela.

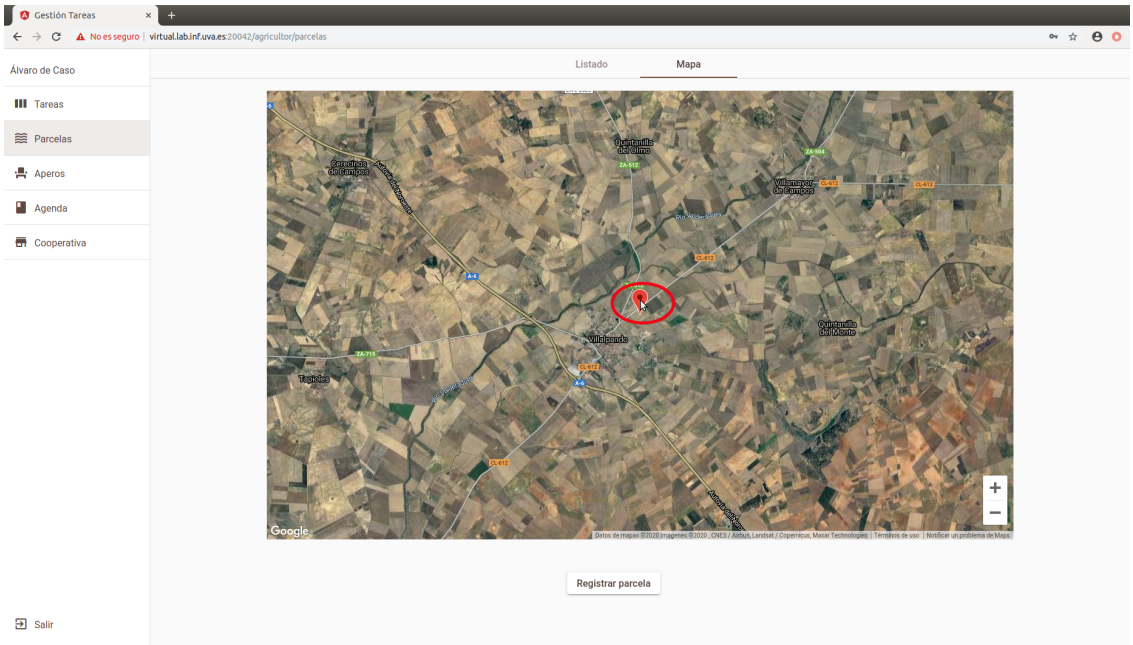

Figura A.18: Ver detalles parcela en el mapa.

En ambos casos se abrirá el siguiente *pop-up* mostrando los detalles de la parcela y su historial de tareas.

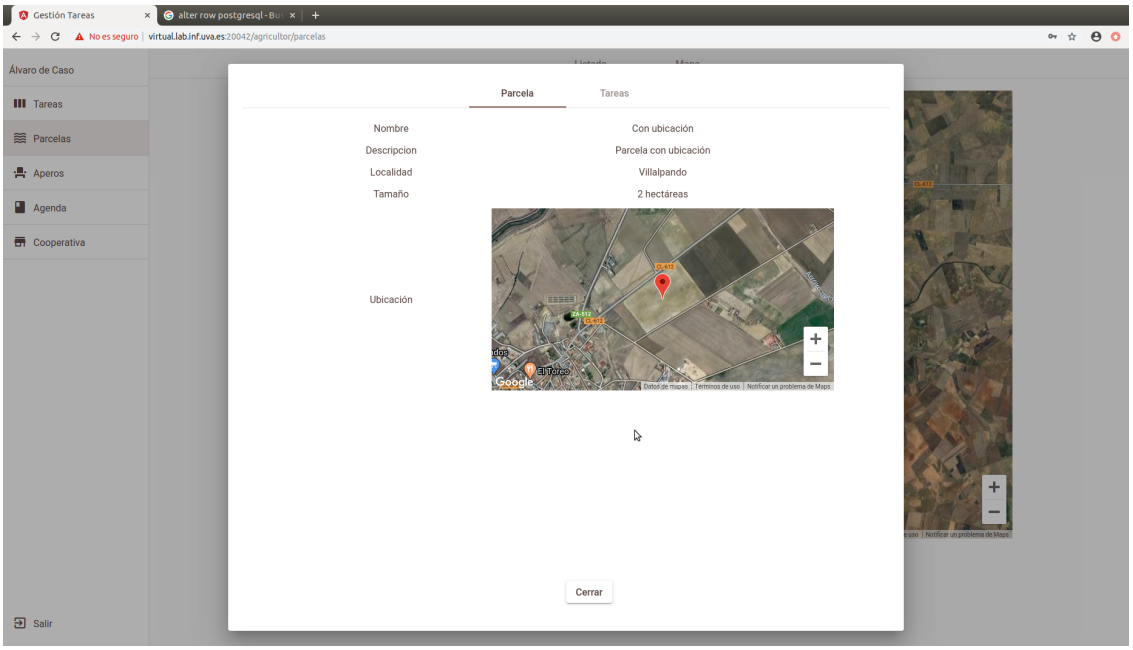

Figura A.19: Ver detalles parcela.

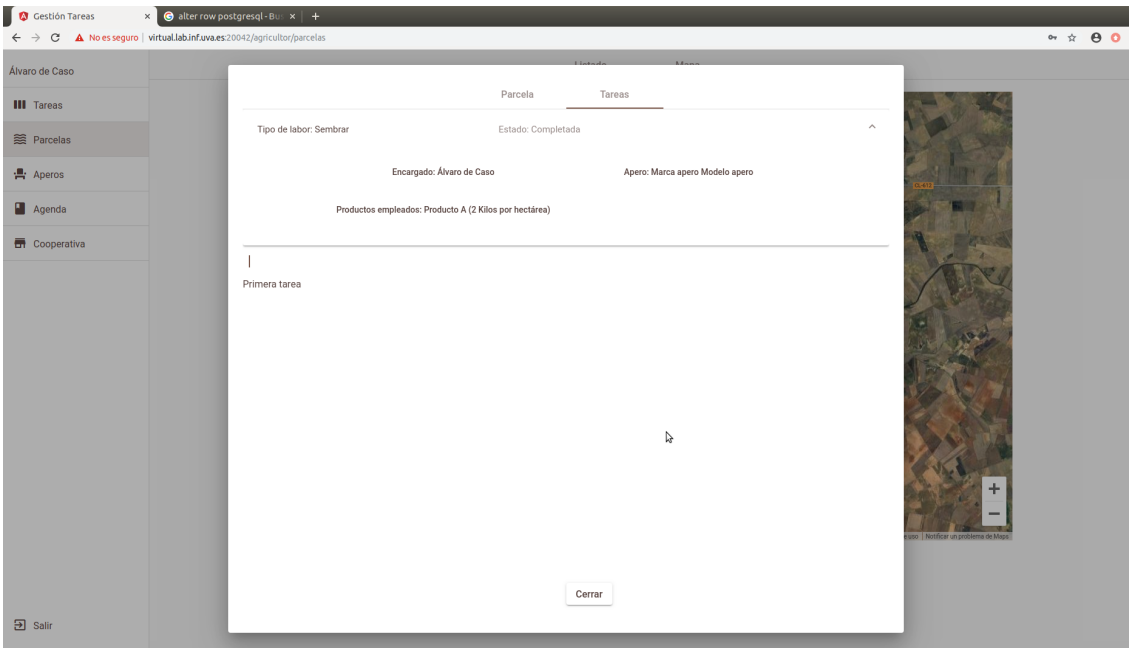

Figura A.20: Ver historial de tareas de una parcela.

Registrar parcela

Para registrar una parcela en primer lugar habrá que hacer click en el botón Parcelas de la parte izquierda de la pantalla. Esto cargará la página Ver parcelas lista y haciendo click en el botón Registrar parcela lo que abrirá el siguiente formulario.

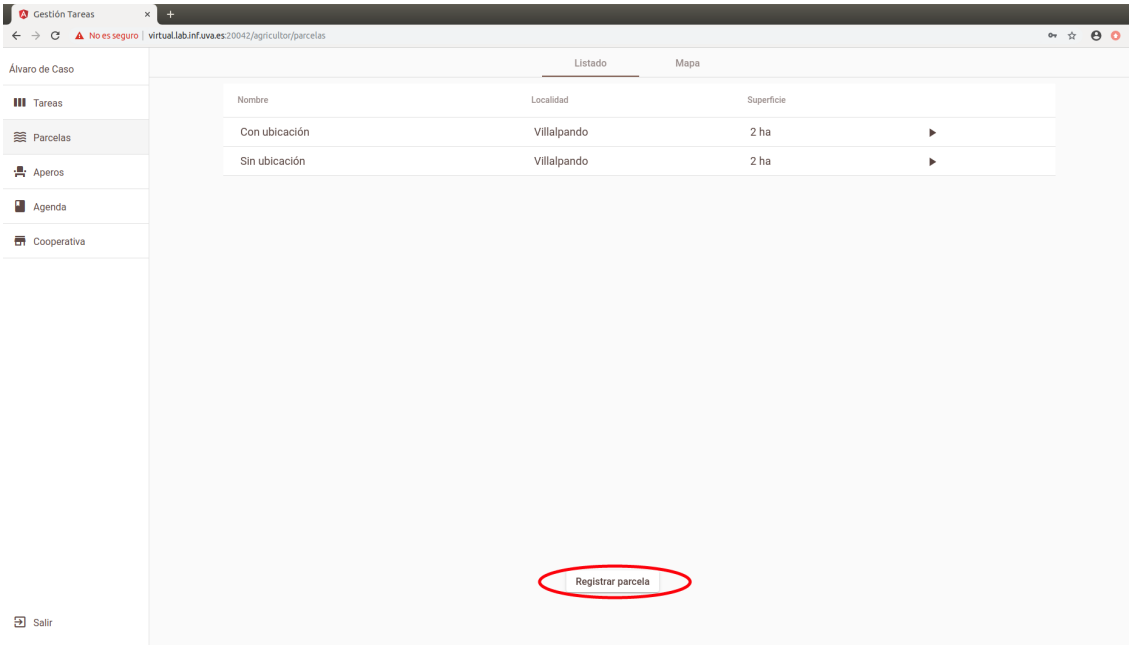

Figura A.21: Botón registrar parcela.

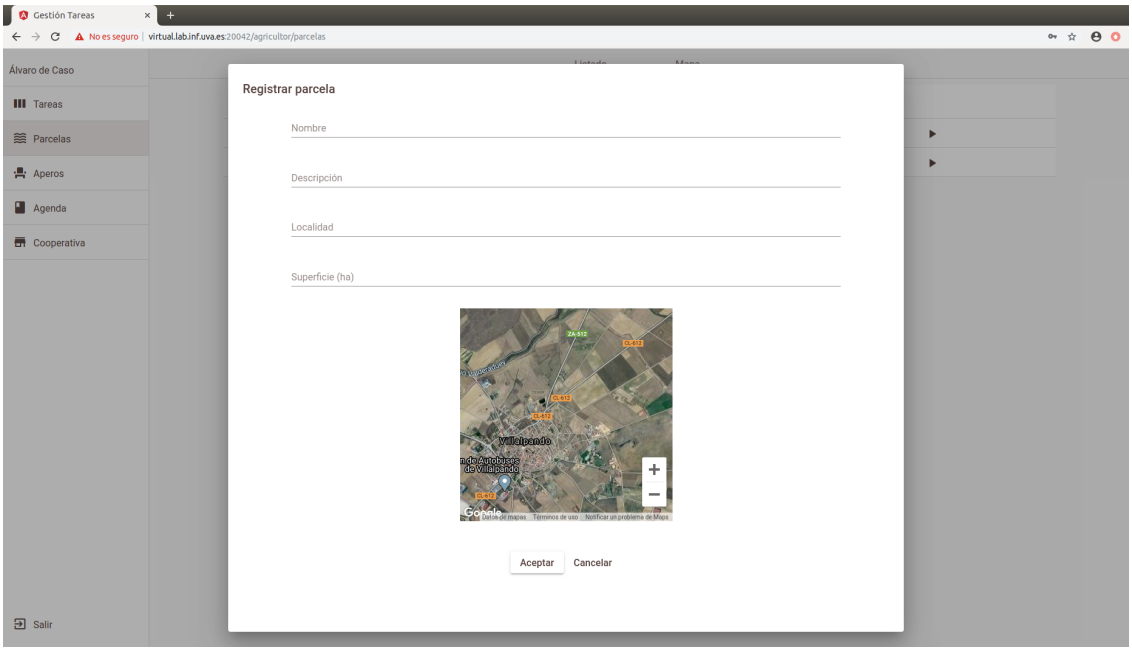

Figura A.22: Formulario registrar parcela.

En este formulario habrá que indicar al menos los campos que son obligatorios y pulsar Aceptar para registrar la parcela. Para indicar la ubicación de una parcela habrá que buscar la zona donde se encuentra la parcela (utilizando los controles de un mapa de Google) y hacer click en la parcela de esta.

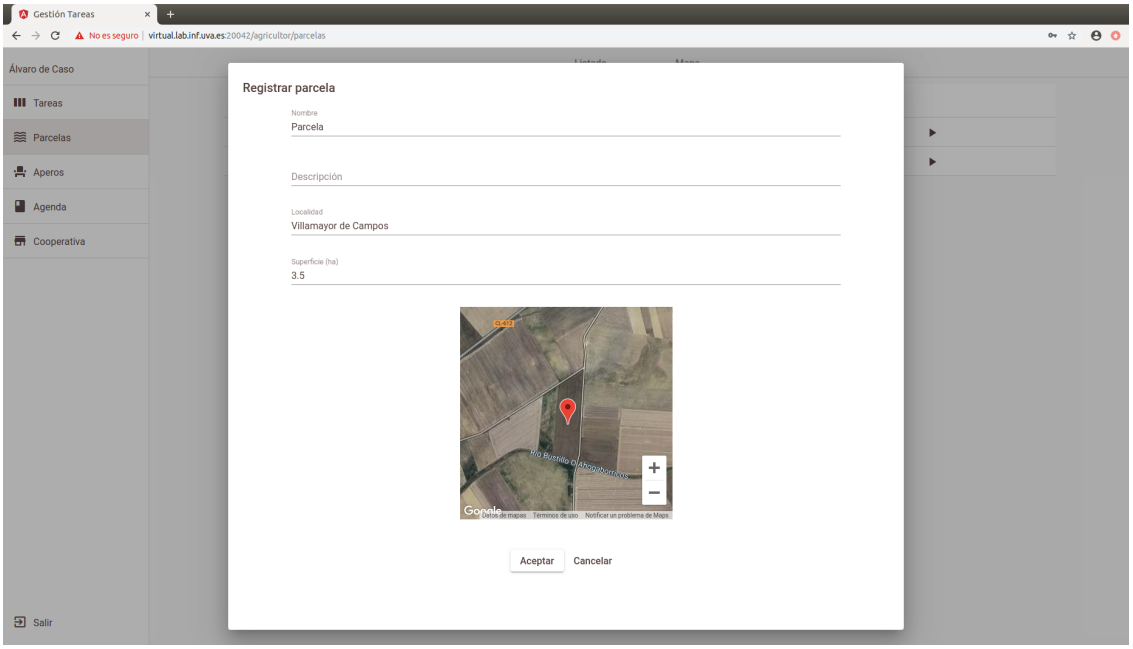

Figura A.23: Marcar ubicación parcela.

Ver aperos registrados

Para ver los aperos que un agricultor tiene registrados en el sistema sólo tiene que hacer click en botón Aperos de la parte izquierda de la pantalla lo que cargará la siguiente página.

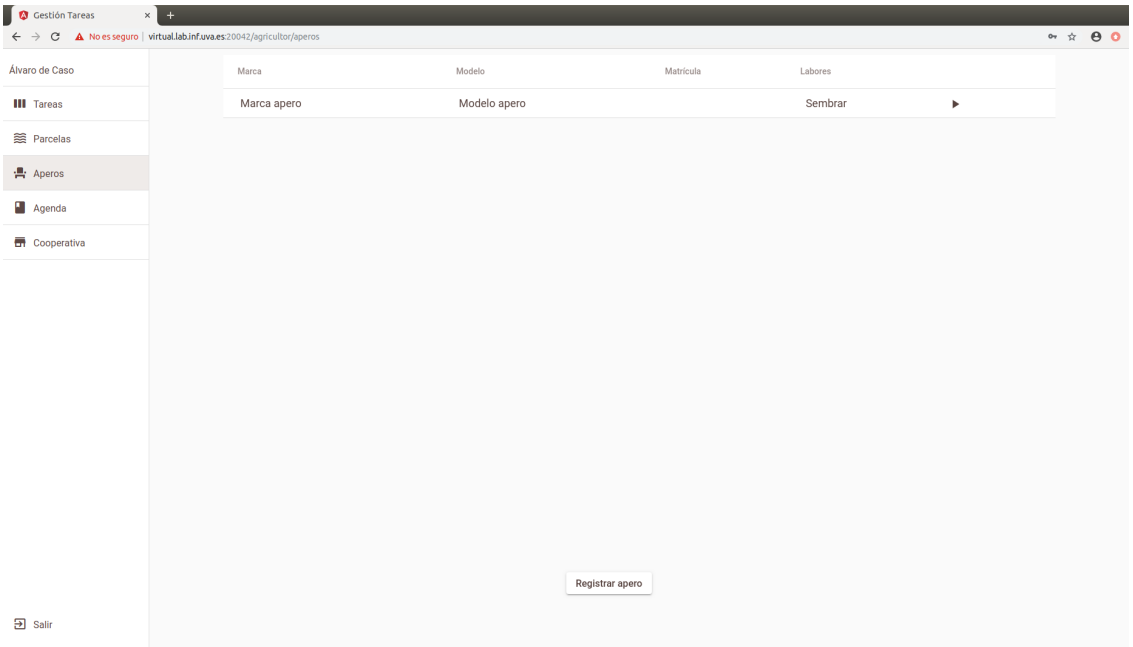

Figura A.24: Ver aperos registrados.

Ver detalles de un apero

Para ver los detalles de un apero, el usuario tendrá que hacer click en Aperos lo que cargará la lista de los aperos y en la flecha de la cuarta columna de la tabla lo que abrirá el *pop-up*.

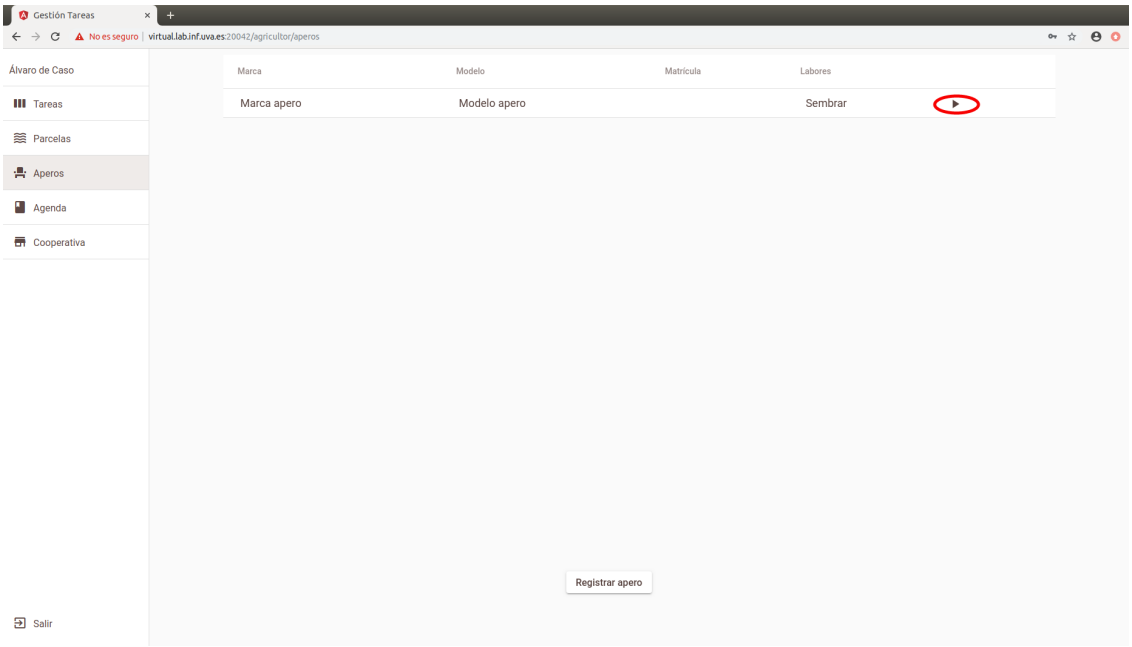

Figura A.25: Ver detalles apero.

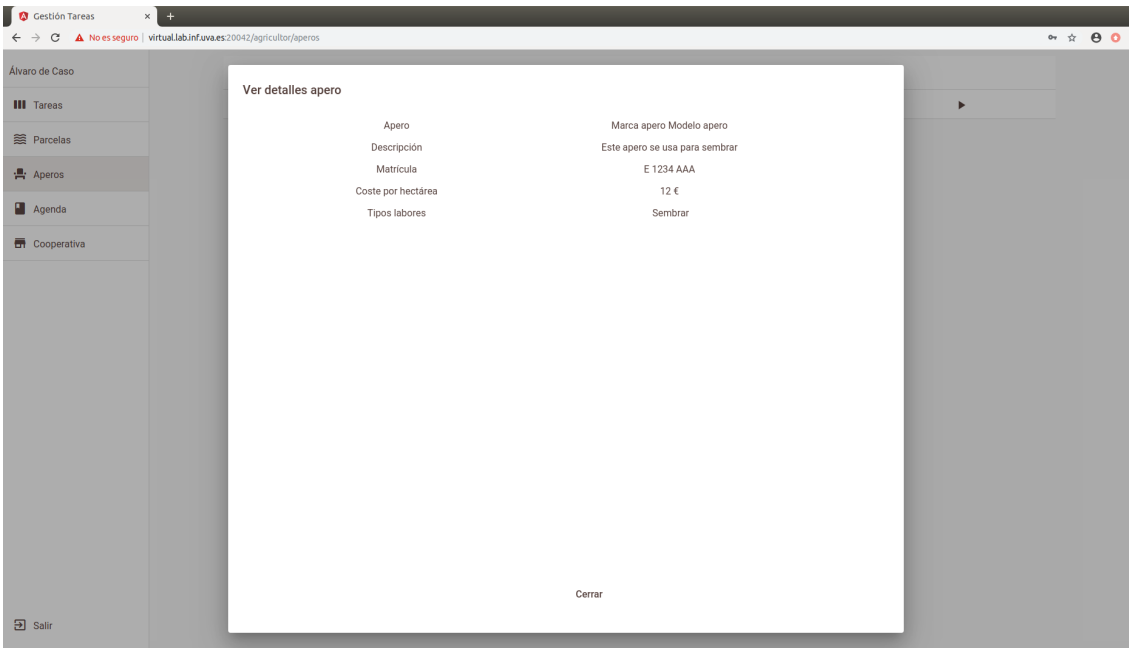

Figura A.26: Detalles apero.

Registrar un apero

De forma similar a como se registra una parcela, para registrar un apero hay que hacer click en el botón Aperos y en Registrar apero en la parte inferior de la pantalla. Esto abrirá el siguiente formulario para indicar que

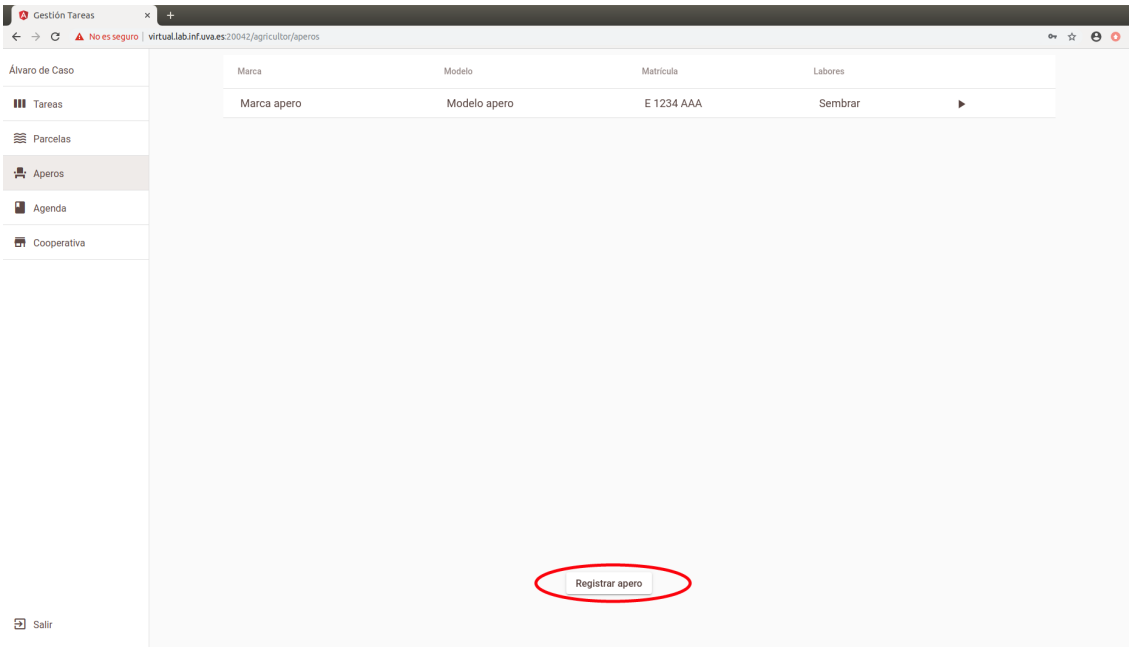

Figura A.27: Botón registrar apero.

características tiene el apero que se quiere registrar. Una vez se han completado los campos, habrá que hacer click en Aceptar para que el sistema lo registre.

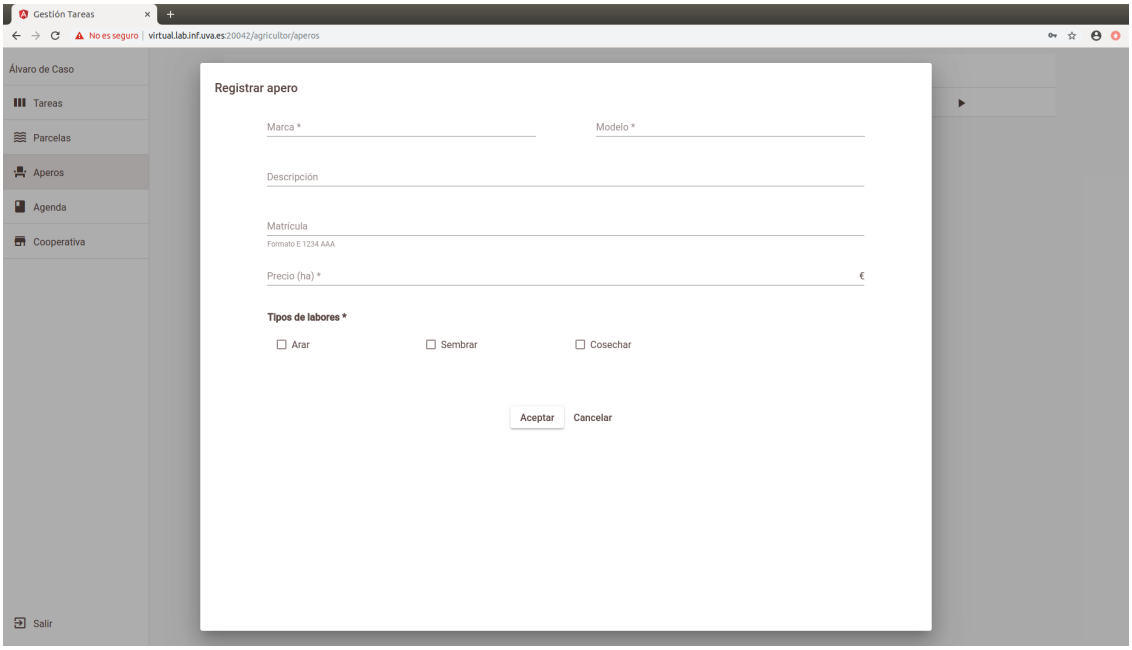

Figura A.28: Botón registrar apero.

Unirse a una cooperativa/ Ver productos de una cooperativa

Al hacer click en Cooperativa lo que mostrará la página dependerá si el usuario forma parte de alguna cooperativa o no. Si el usuario no está forma parte de una cooperativa la página que se carga es la siguiente.

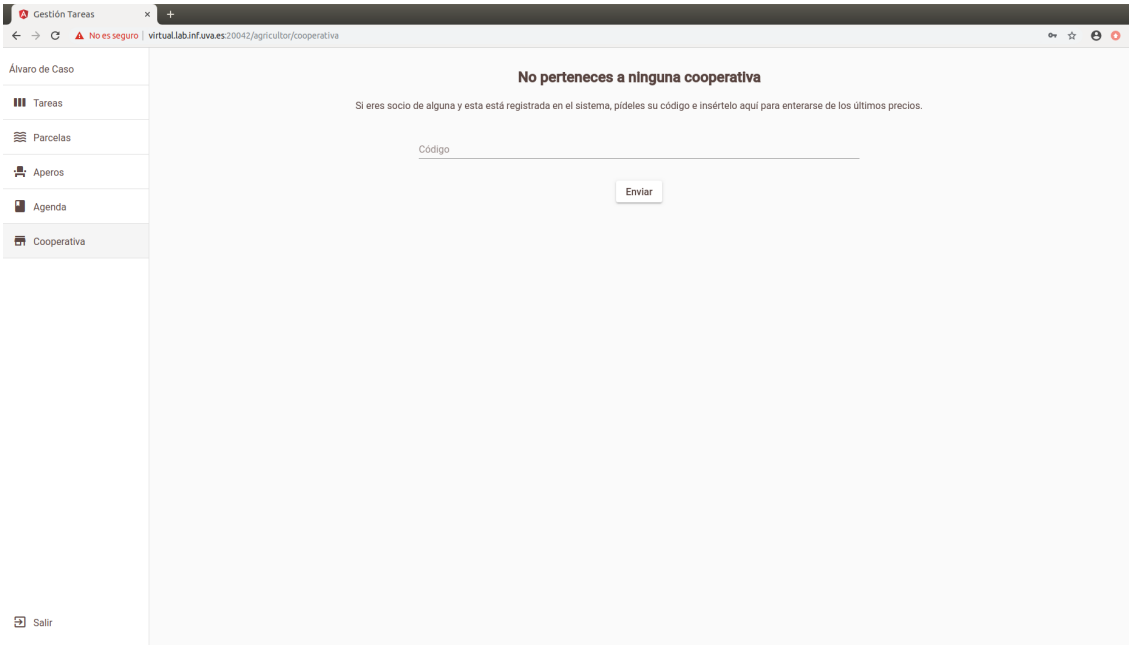

Figura A.29: Sin cooperativa.

En este campo el usuario podrá introducir el código de una cooperativa para poder ver los productos que oferta y sus precios.
Cuando el usuario forma parte de una cooperativa lo que carga el sistema es lo siguiente.

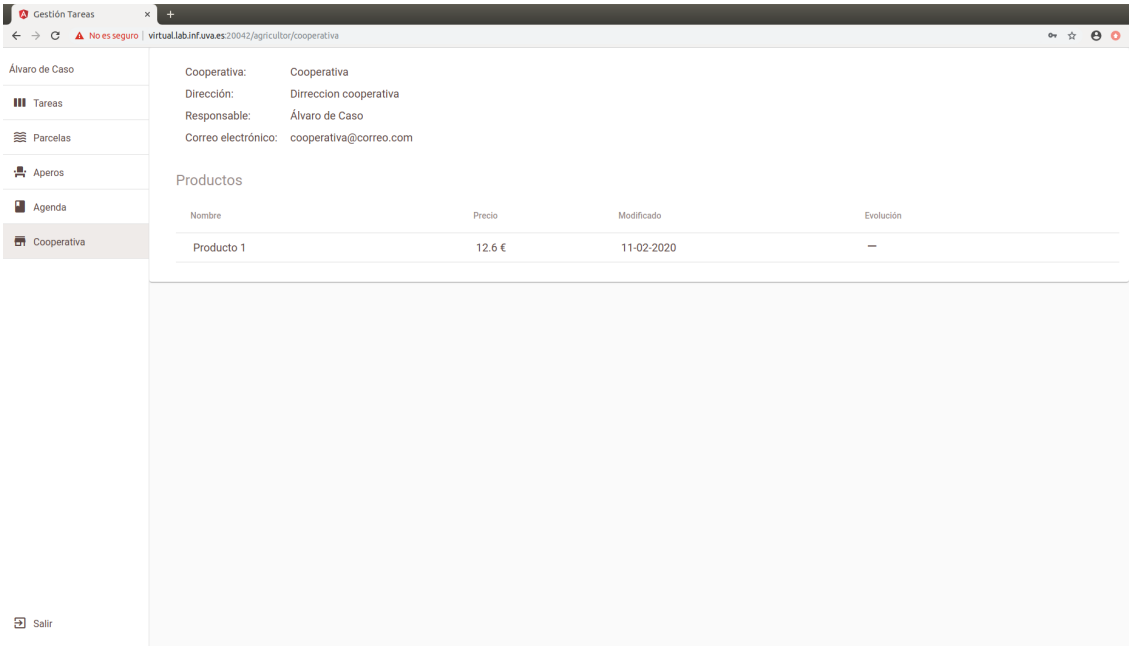

Figura A.30: Listado productos cooperativa.

En esta página se puede ver en la parte superior los datos básicos de la cooperativa y a continuación un listado de los productos que oferta. De izquierda a derecha se muestra el nombre del producto, el precio actual, la última fecha de modificación y la evolución del precio del producto (si subió, se mantuvo o bajo desde la última modificación).

Ver agricultores de la agenda

Para ver los agricultores que el usuario tiene registrados en la agenda sólo tiene que hacer click en el botón Agenda de la parte izquierda de la pantalla lo que mostrará una lista con estos.

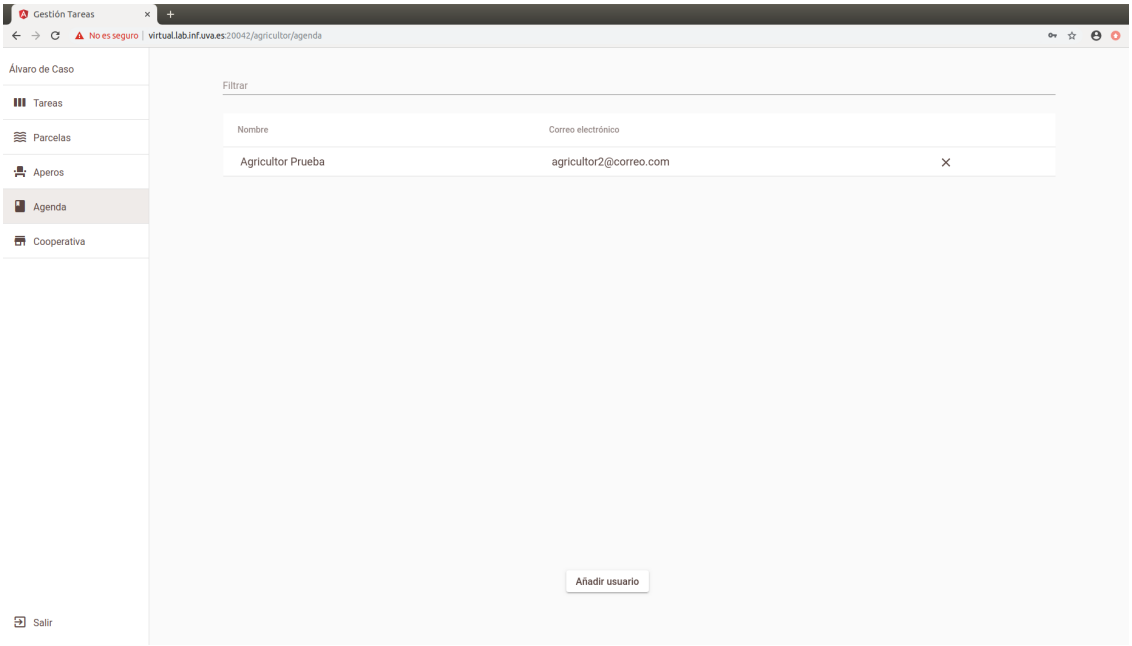

Figura A.31: Listado de usuarios.

Añadir un agricultor a la agenda

Para añadir un usuario a la agenda primero hay que hacer click en Agenda y a continuación en el botón Añadir usuario que se encuentra en la parte inferior de la pantalla lo que abrirá un formulario.

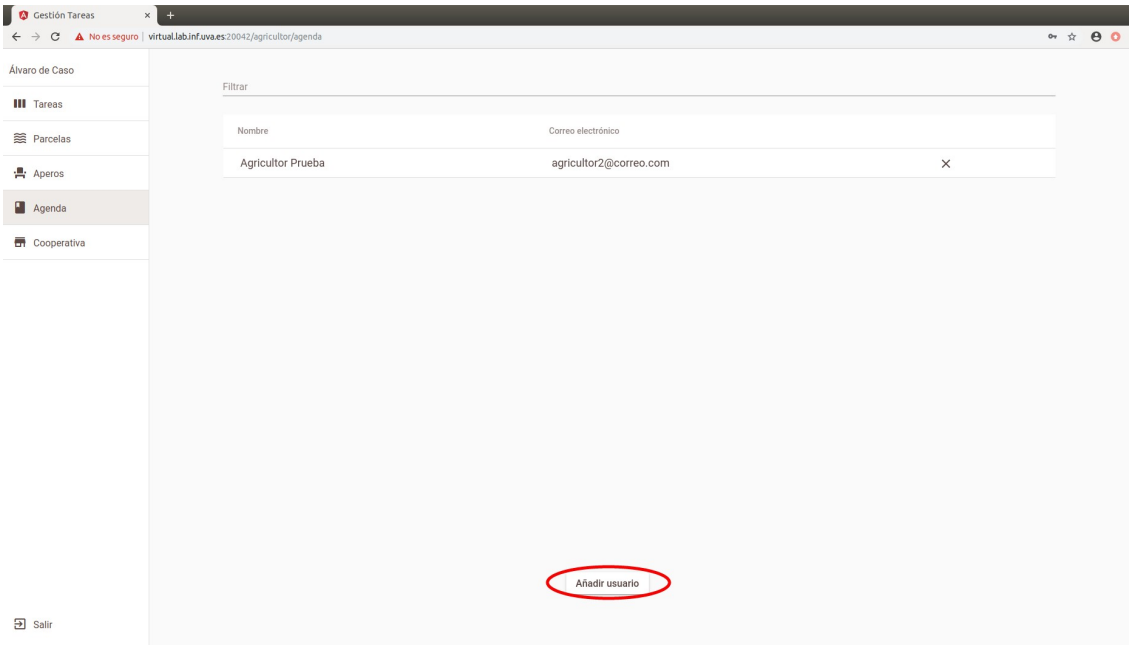

Figura A.32: Botón añadir usuario.

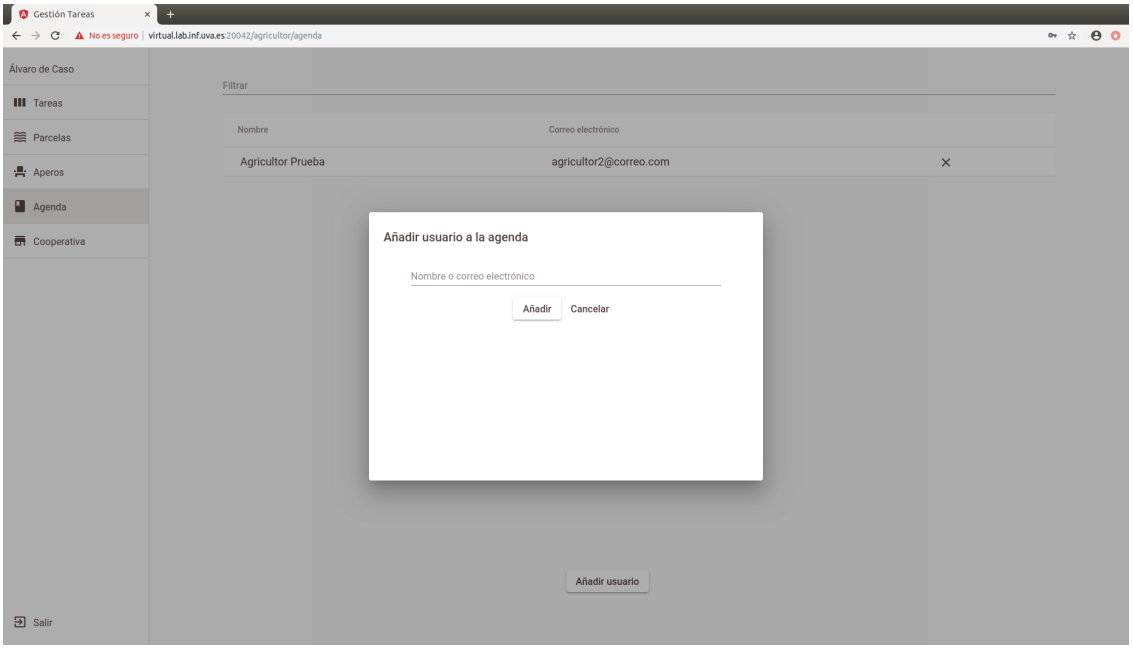

Figura A.33: Formulario añadir nuevo usuario.

En este formulario habrá que indicar el nombre del usuario que se quiere añadir a la agenda y seleccionar una de las opciones que muestra el sistema. Una vez hecho esto, pulsar Añadir.

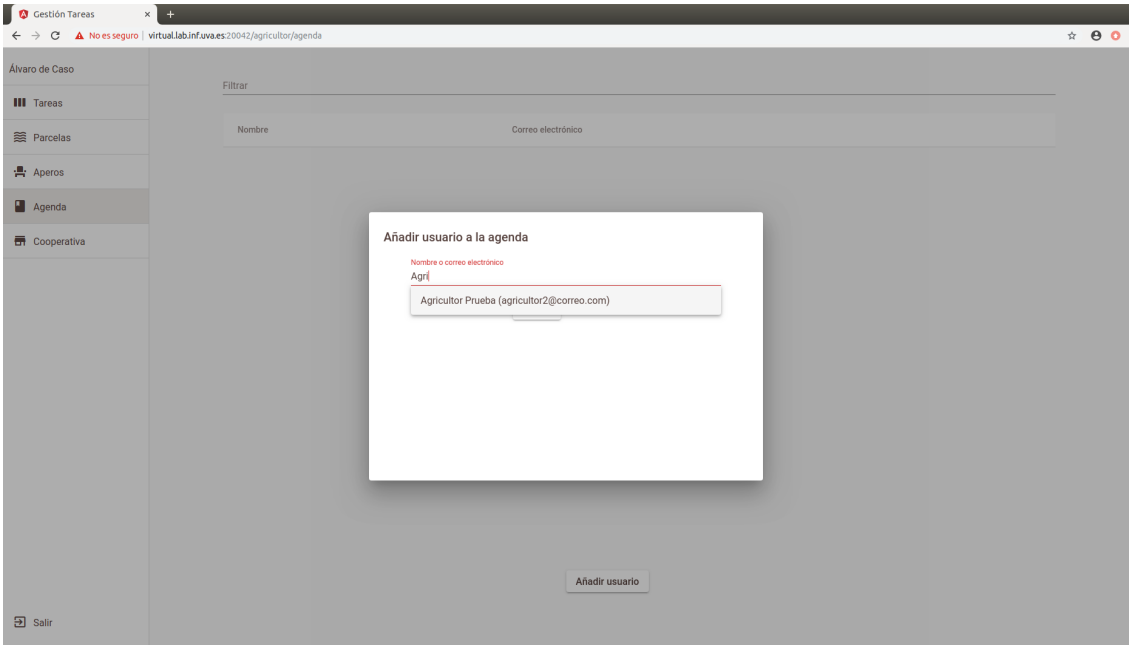

Figura A.34: Búsqueda del nuevo usuario.

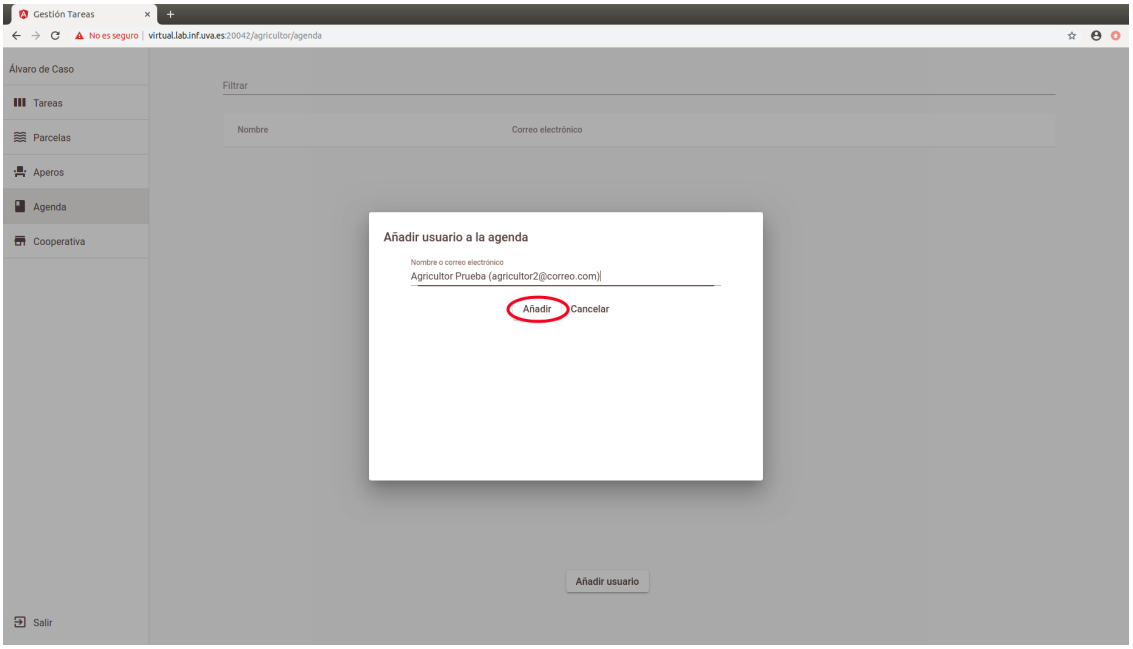

Figura A.35: Añadir usuario.

Eliminar un agricultor de la agenda

Para eliminar un usuario de la agenda el usuario tendrá que hacer click en el botón de Agenda para que el sistema le muestre el listado de los usuarios y hacer click en la X de la tercera columna.

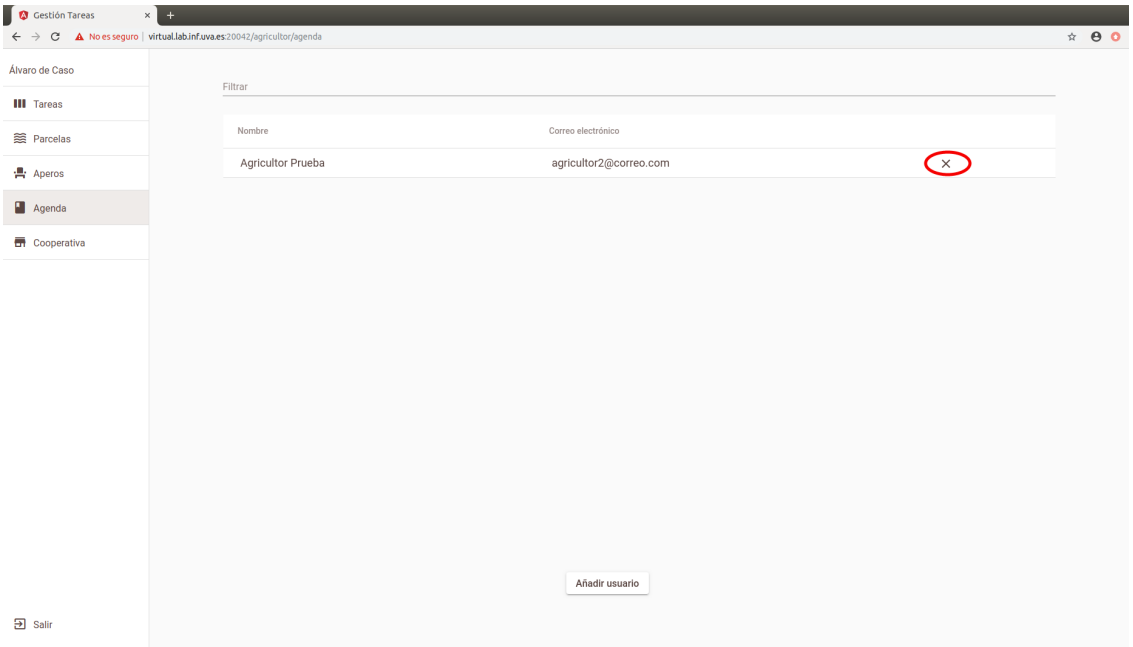

Figura A.36: Eliminar usuario agenda.

Ver tareas

Para ver las tareas que tienen como encargado el usuario que se encuentra identificado en el sistema. Las tareas se agruparán en tres columnas según el estado: estimadas, pendientes de estimación o completadas.

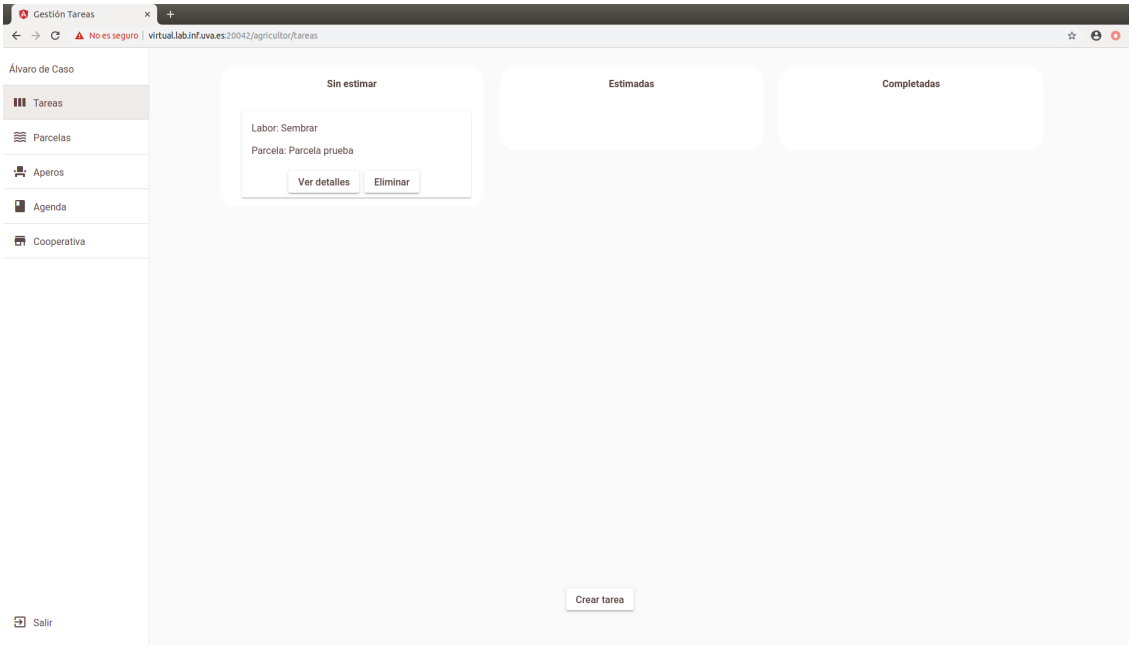

Figura A.37: Ver tareas.

Si en la esquina superior derecha de una tarea aparece una exclamación, quiere decir que el usuario que está identificado en el sistema ha creado, pero no es el encargado de la tarea (es una tarea delegada). Por lo tanto, no podrá actualizar el estado de esta.

Crear nueva tarea

Para crear una tarea hay que hacer click en primer lugar en Tareas y a continuación el botón que se encuentra en la parte inferior de la pantalla. Esto abrirá un formulario en el que habrá que indicar la parcela, el tipo de

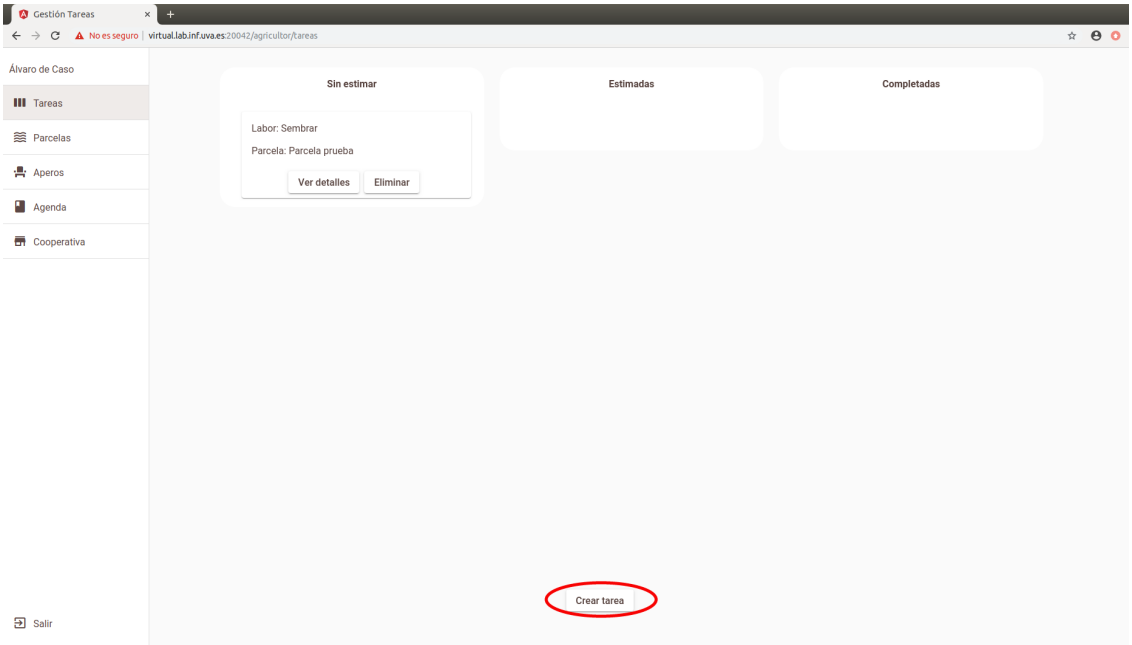

Figura A.38: Crear tarea.

labor que se quiere hacer y el encargado de la tarea. Además si la tarea necesita algún producto habrá que indicar también el nombre del producto. Se pueden indicar hasta tres productos haciendo click en el botón añadir producto. También, si se han añadido producto de más se pueden eliminar haciendo click en la papelera. Una vez se hayan indicado los campos, habrá que pulsar en Guarda y el sistema creará la tarea.

Figura A.39: Crear tarea sin producto.

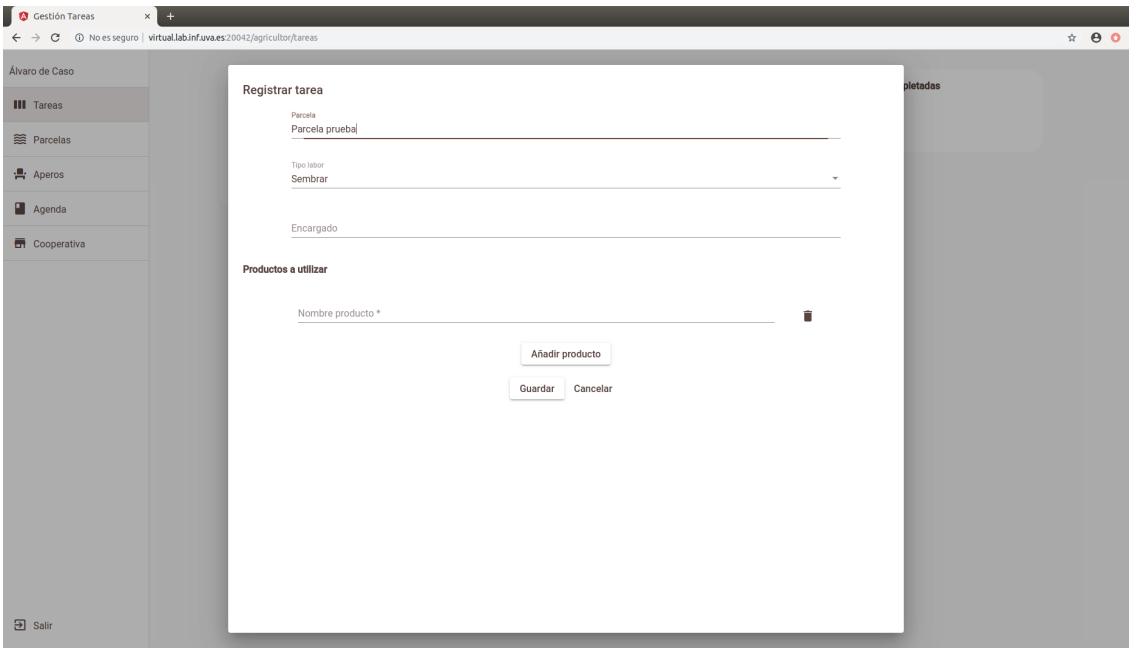

Figura A.40: Crear tarea con producto.

Eliminar una tarea

Para eliminar una tarea hay que hacer click en Tareas para que el sistema muestre las tareas que tenemos asignadas y hacer click en el botón eliminar. Si la tarea no puede ser eliminada, el sistema mostrará un mensaje de error.

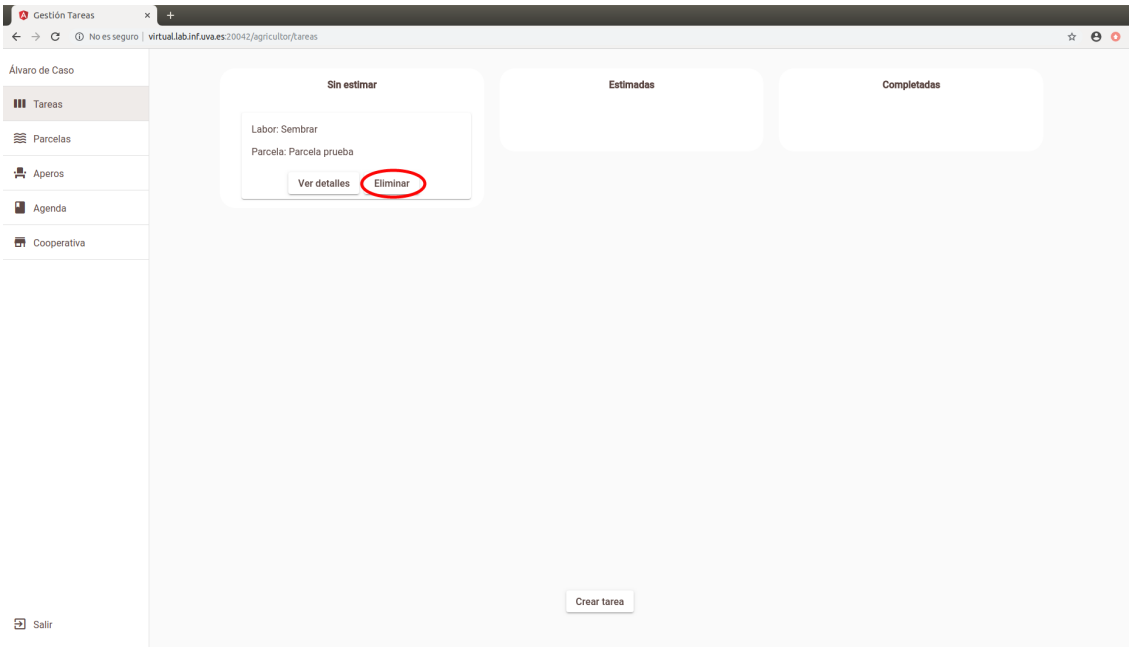

Figura A.41: Eliminar tarea.

Estimar una tarea

Para estimar una tarea el usuario tiene que hacer click “sostenido” en la tarea que se quiere estimar y arrastrarla y soltarla en la columna Estimadas lo que abrirá un formulario.

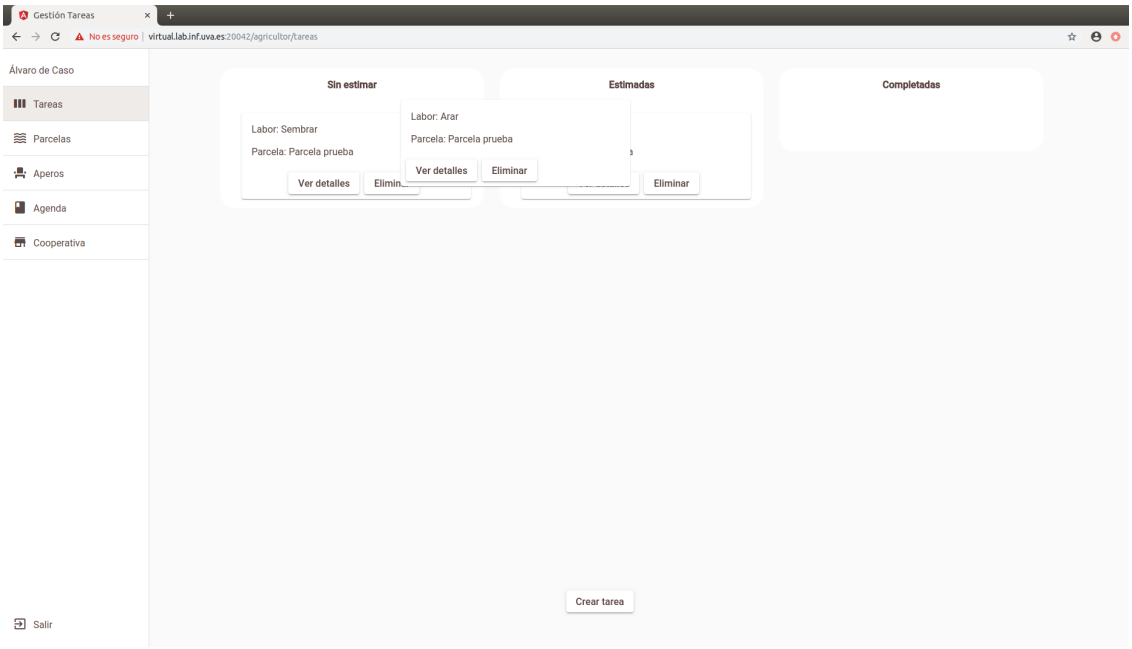

Figura A.42: Estimar tarea.

El formulario que se abrirá dependerá de si el tipo de labor necesita o no algún producto.

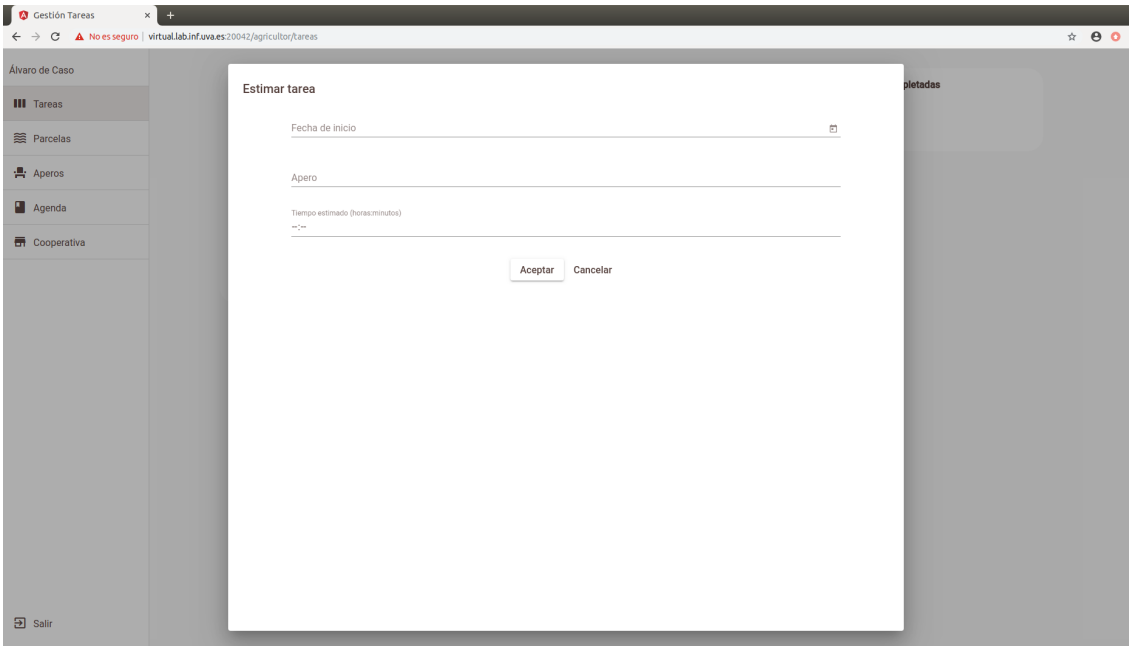

Figura A.43: Estimar tarea que no necesita producto.

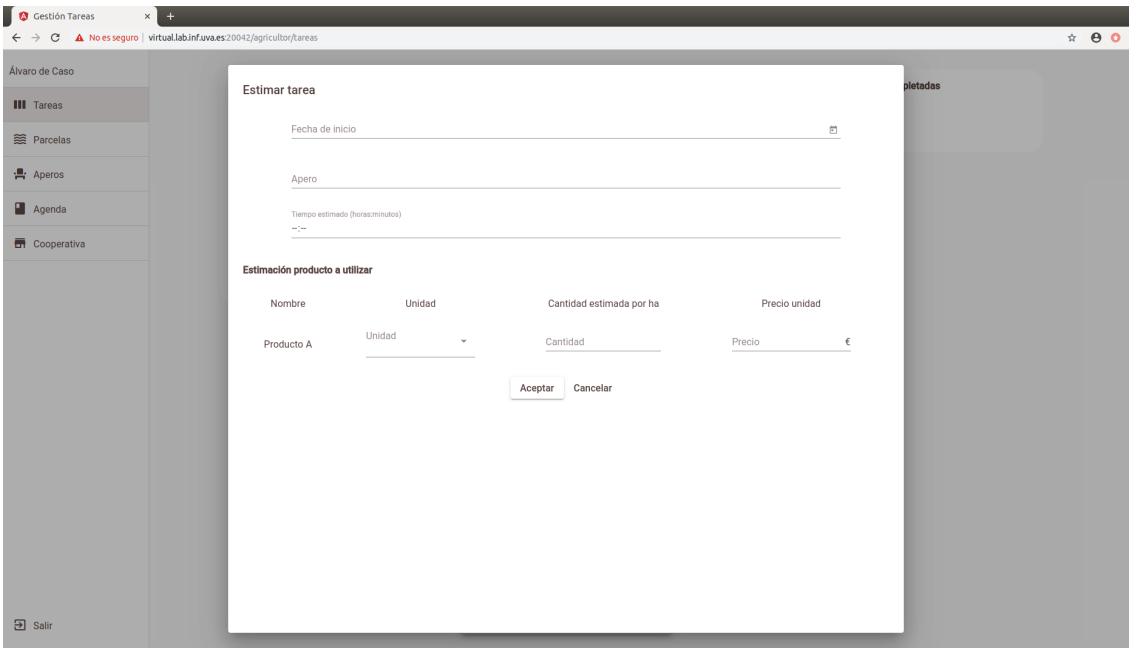

Figura A.44: Estimar tarea que necesita producto.

Una vez introducido los datos sólo habrá que hacer click en Aceptar para que se actualice el estado de la tarea.

Completar una tarea

De forma similar a como se estima una tarea, para completarla hay que hacer de nuevo pulsación "sostenida" en la tarea que queremos actualizar. En este caso se pueden abrir cuatro formularios diferentes: si la tarea ni necesita ni recoge ningún producto, si la tarea necesita algún producto, si la tarea recoge algún producto o las dos anteriores (cuyo formulario sería la combinación de los dos anteriores).

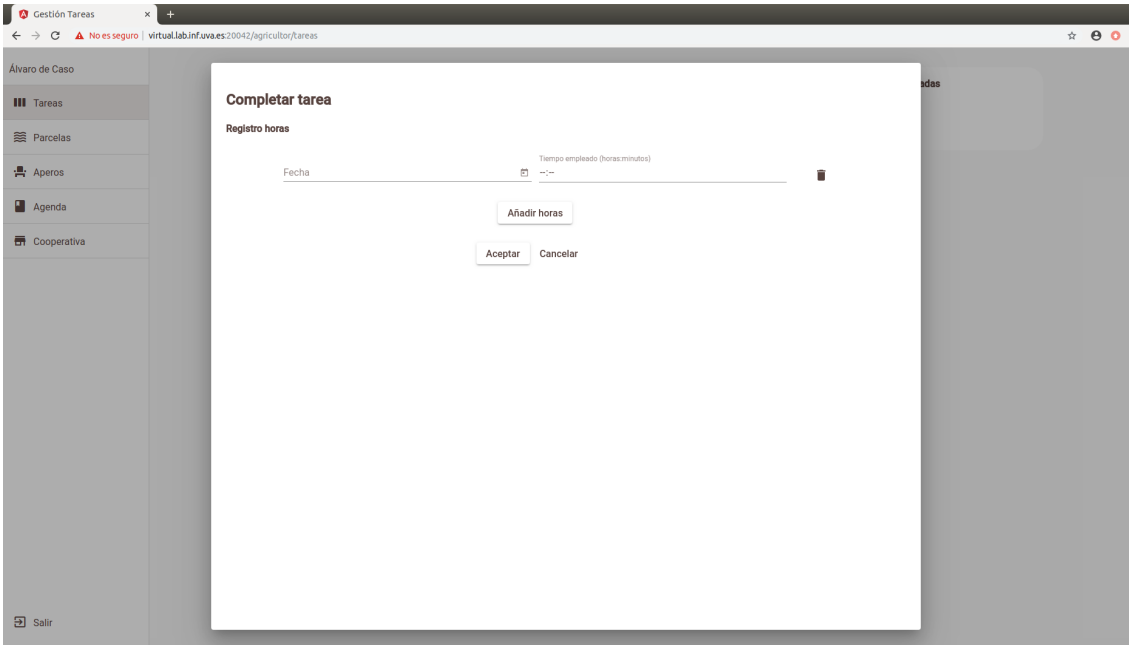

Figura A.45: Completar tarea.

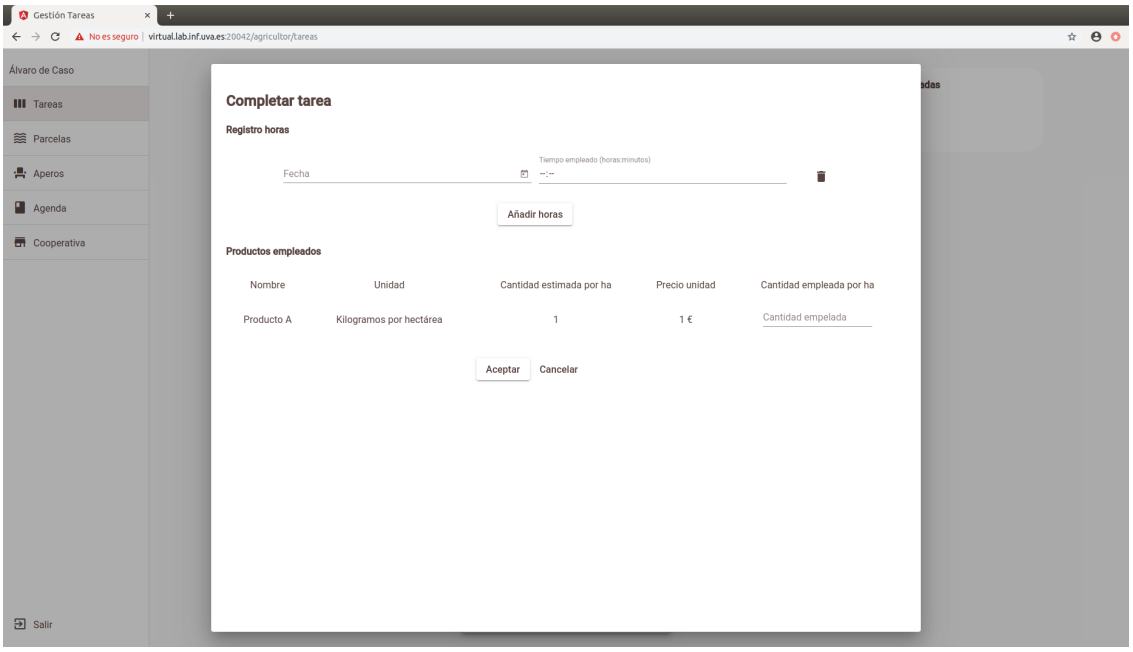

Figura A.46: Completar tarea que necesita producto.

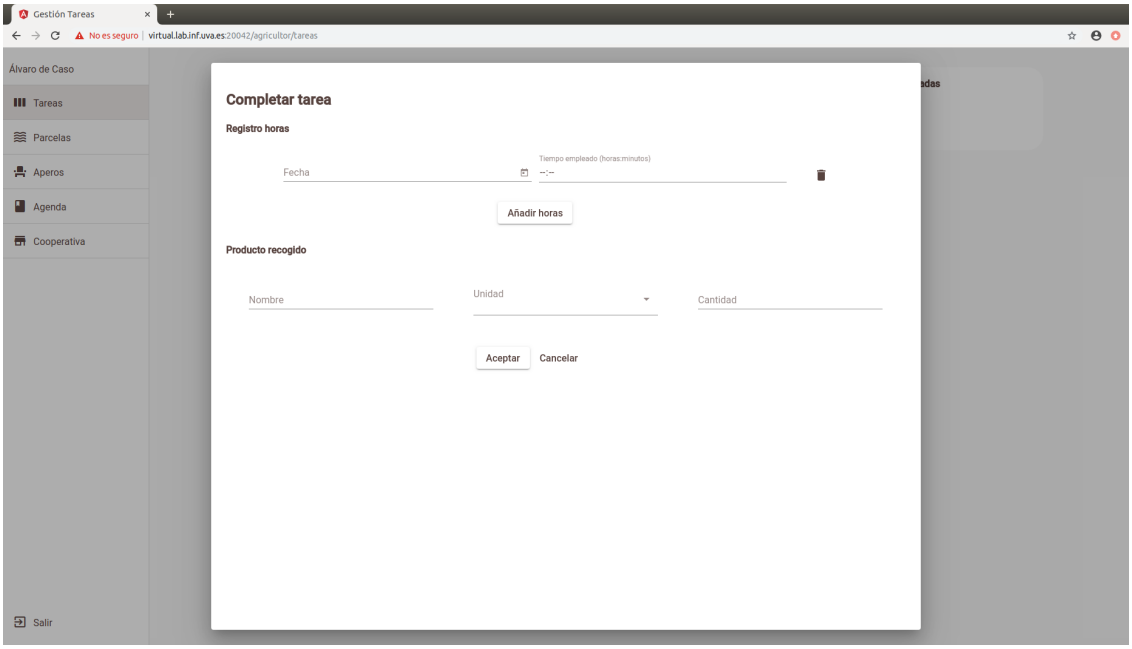

Figura A.47: Completar tarea que recoge producto.

Una vez se haya abierto el formulario,el usuario podrá añadir o eliminar los registros de horas que necesite haciendo click en Añadir horas o en la papelera a la derecha de cada registro y también tendrá que completar el resto de campos en función del tipo de labor. Una vez que haya completado los datos, el usuario tendrá que hacer click en Aceptar para que el sistema actualice la tarea.

Generar hoja de costes

Cuando una tarea es delegada, el encargado de la tarea puede ver un botón Generar hoja de costes. Haciendo click en este botón el navegador abrirá una pestaña donde se mostrarán los detalles de la tarea y el coste total de realizarla.

A.1.4. Operaciones relacionadas con los responsables de cooperativas

Página principal

Cuando un usuario de tipo responsable de cooperativa se identifica en el sistema, la página que se cargá es la siguiente.

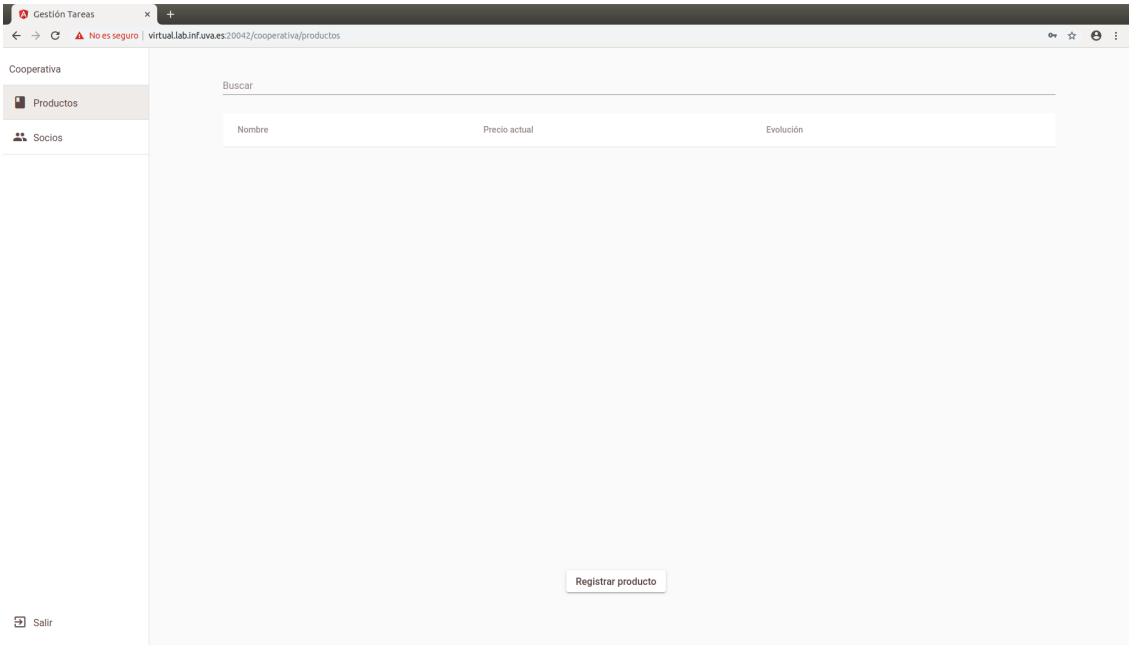

Figura A.48: Pantalla inicial cooperativa.

En el menú de la izquierda podemos observar los siguientes botones:

- *Productos* para cargar la página que permitirá la gestión de los productos que oferta la cooperativa. Esta es la opción por defecto que se muestra cuando un usuario de este tipo se identifica en el sistema.
- *Socios* para poder generar el código para que los agricultores se unan a la cooperativa.

Ver productos

Para ver los productos que está ofertando la cooperativa sólo hay que hacer click en el botón Productos del menú de la izquierda lo que cargará un listado de los productos.

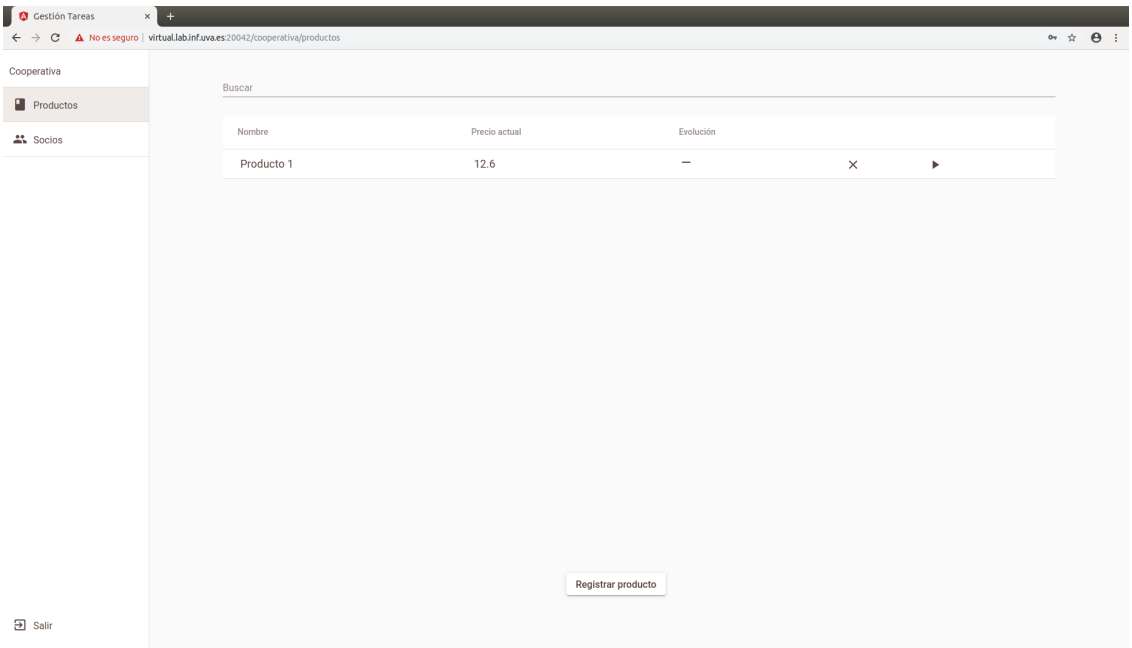

Figura A.49: Ver productos cooperativa.

De izquierda a derecha se muestra el nombre del producto, el precio actual, la evolución del precio, un botón para eliminarlo y un botón para ver los detalles.

Añadir producto

Para añadir un nuevo producto hay que hacer click en Productos y en el botón que se encuentra en la parte inferior de la pantalla. Esto abrirá un formulario donde habrá que indicar el nombre del nuevo producto y el precio inicial de este. Una vez que se hayan indicado estos campos, hay que hacer click en Aceptar.

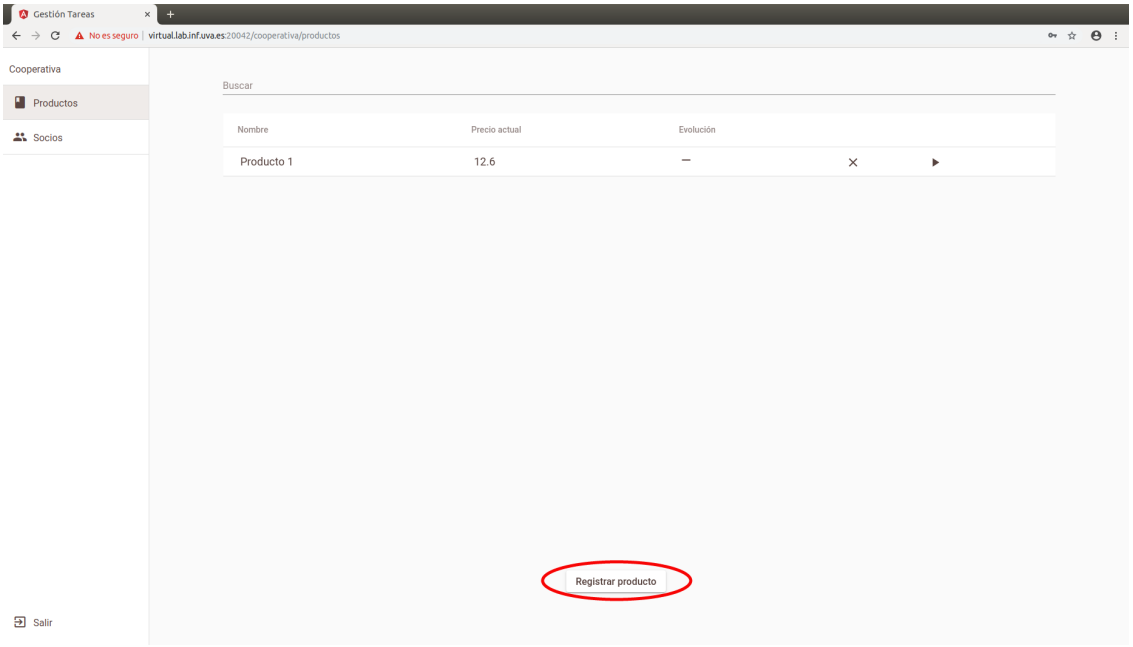

Figura A.50: Botón registrar producto.

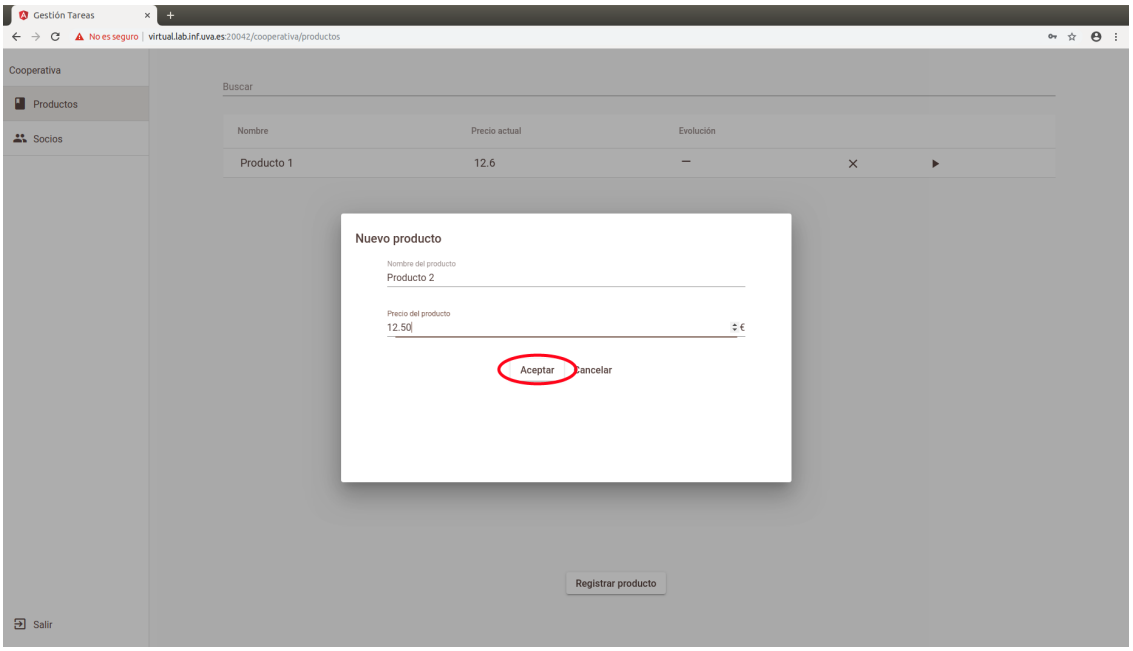

Figura A.51: Formulario nuevo producto.

Eliminar producto

Para eliminar un producto hay que, en primer lugar, hacer click en Productos y a continuación hacer click en la X que se encuentra en la cuarta columna de la tabla.

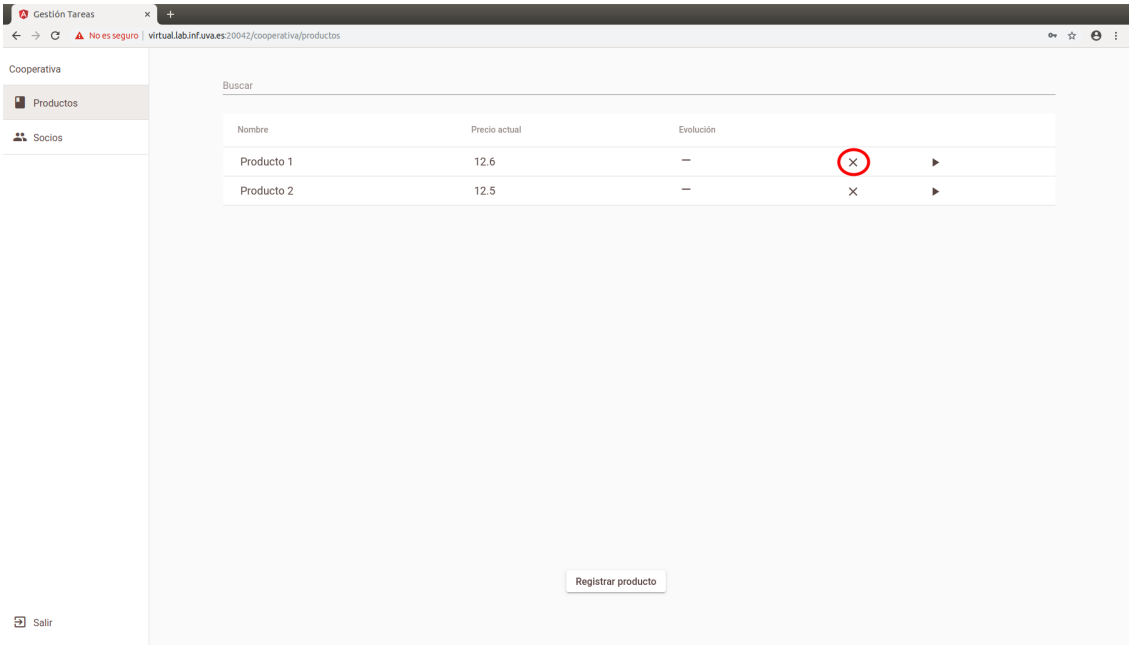

Figura A.52: Eliminar producto.

Modificar precio producto

Para modificar el precio de producto hay que hacer click en Productos en el menú de la izquierda y una vez que se ha cargado la lista de los productos, hacer click en la flecha de la cuarta columna. Esto abrirá una ventana que muestra los detalles del producto y el historial de precios.

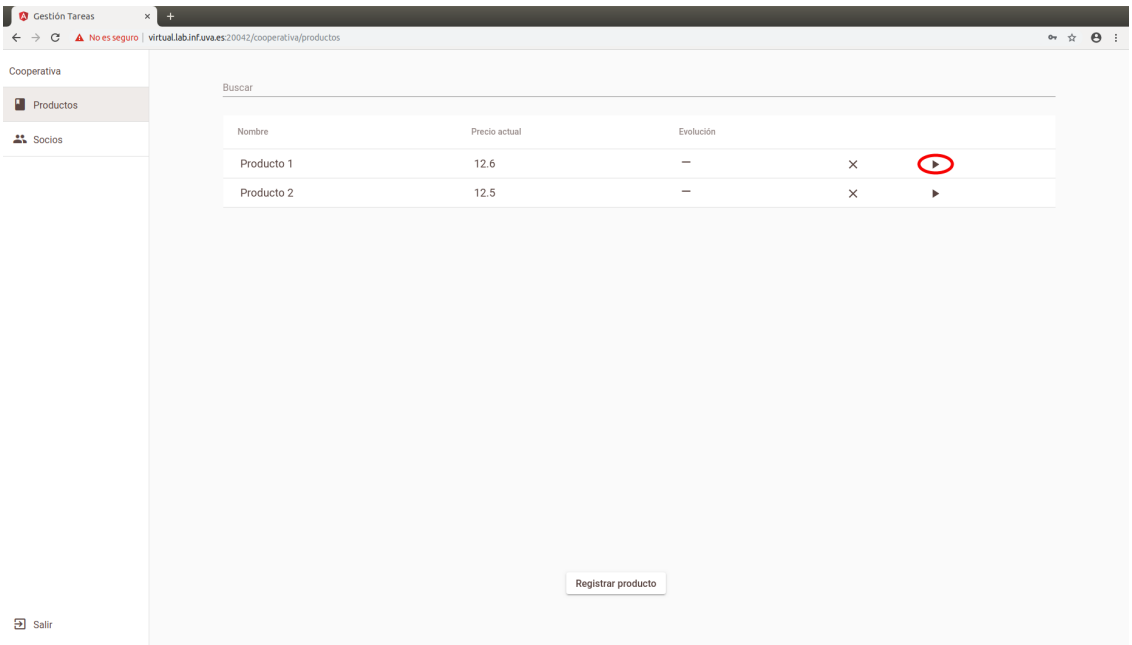

Figura A.53: Botón ver detalles producto.

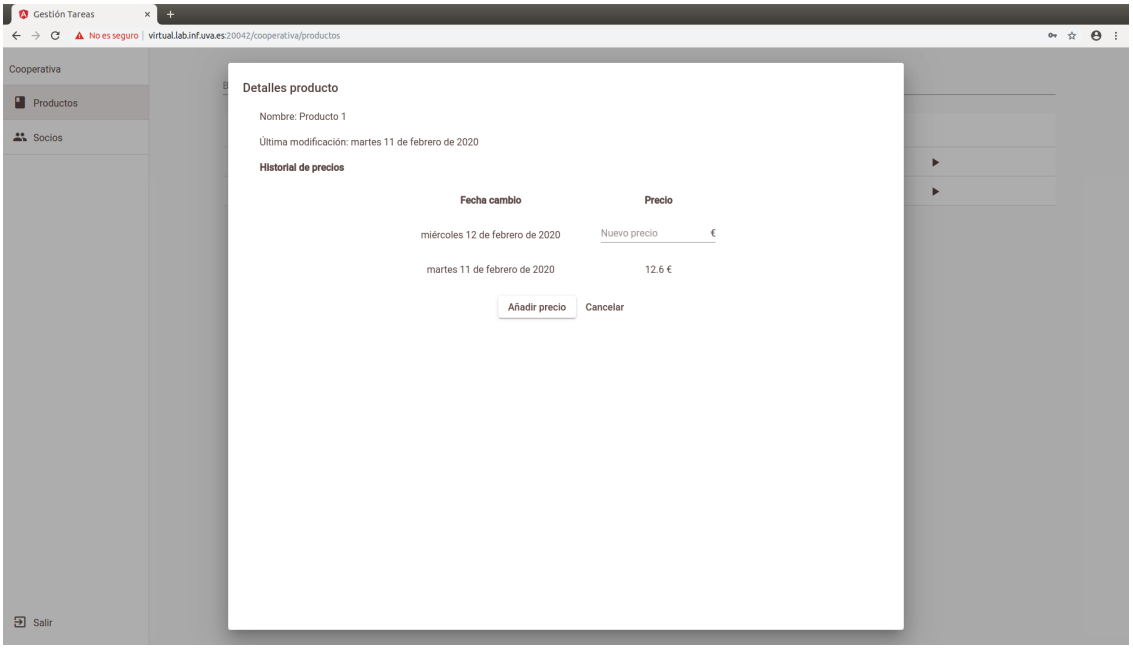

Figura A.54: Formulario nuevo precio.

Para cambiar el precio sólo hay que introducir el nuevo valor y pulsar Añadir precio.

Generar código

Para generar el código para que otros usuarios puedan unirse a la cooperativa hay que hacer click en Socios en el menú de la izquierda de la pantalla y en Generar código.

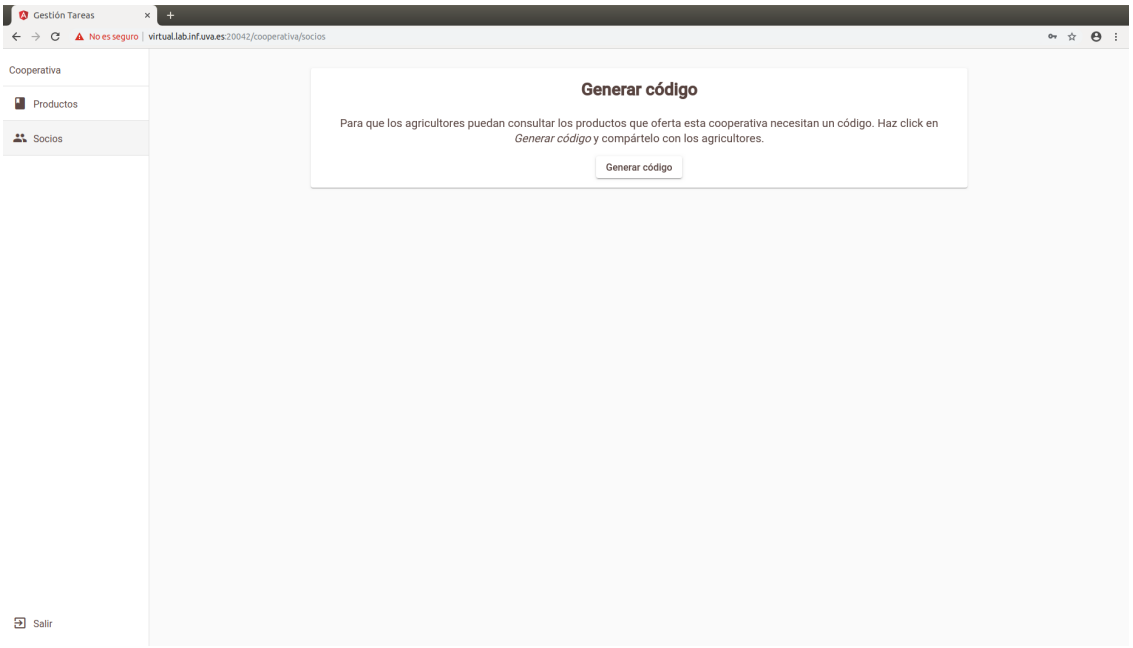

Figura A.55: Generar código cooperativa.

A.1.5. Operaciones relacionadas con los administradores

Ver listado de los usuarios

Cuando un administrador se identifica en el sistema esta es la página que se carga por defecto.

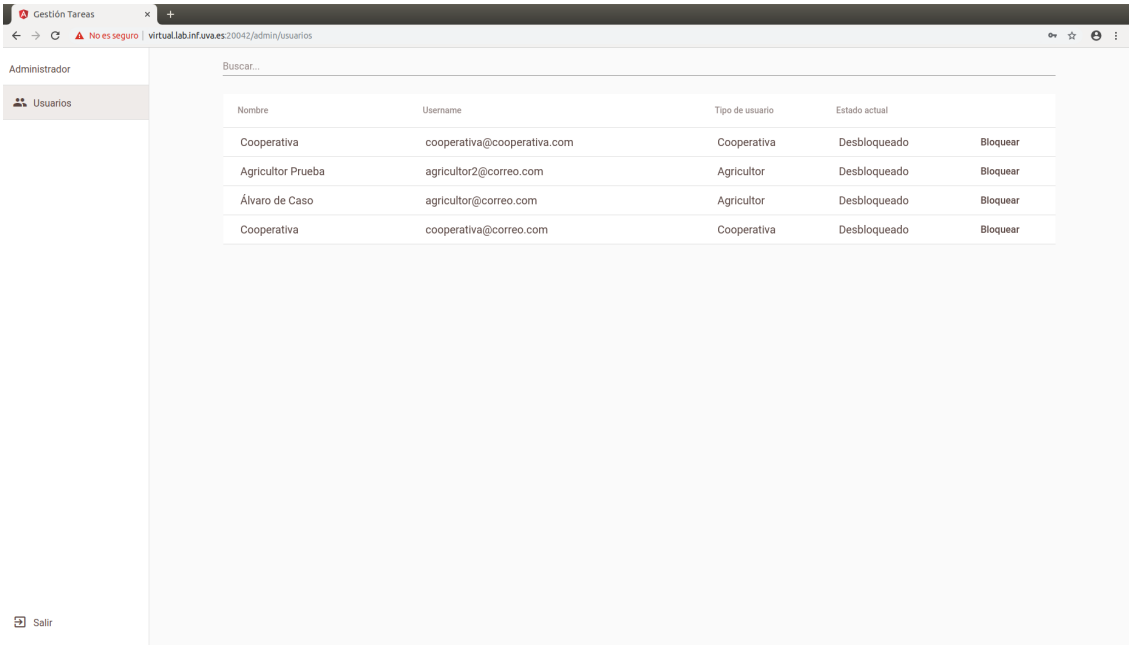

Figura A.56: Ver usuarios.

El sistema muestra el nombre del usuario, su correo electrónico y su estado actual.

Bloquear/Desbloquear usuario

Para bloquear/desbloquear un usuario sólo hay que hacer click en el botón de la cuarta columna.

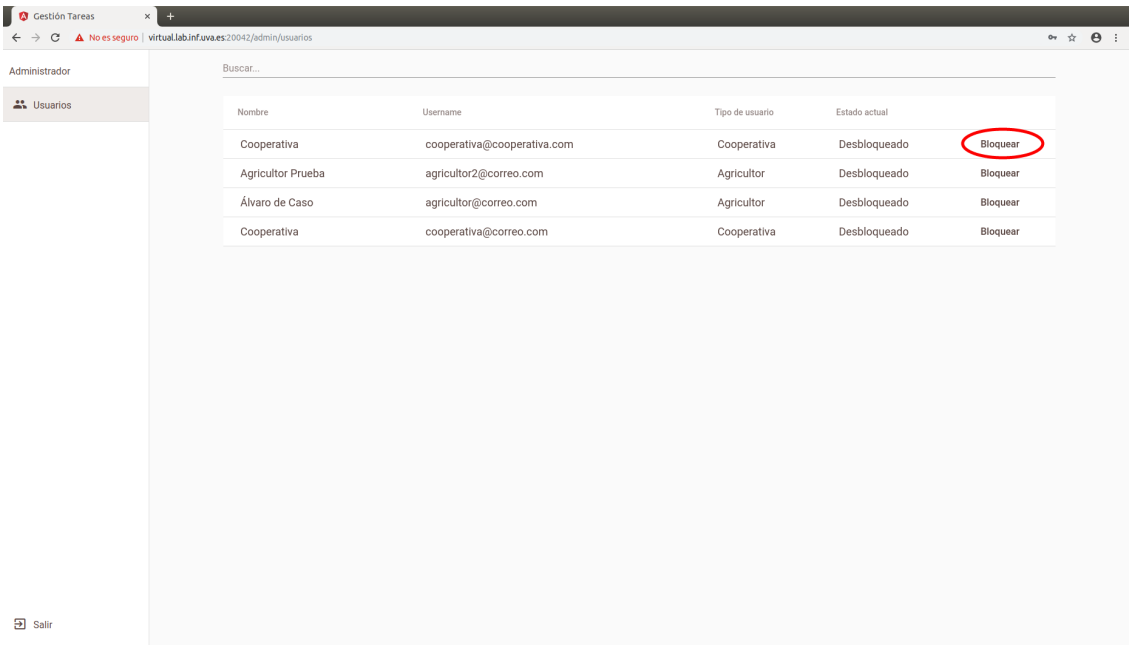

Figura A.57: Bloquear/desbloquear usuario.

A.2. Aplicación Android

Para usar la aplicación, abrir la aplicación con el siguiente icono.

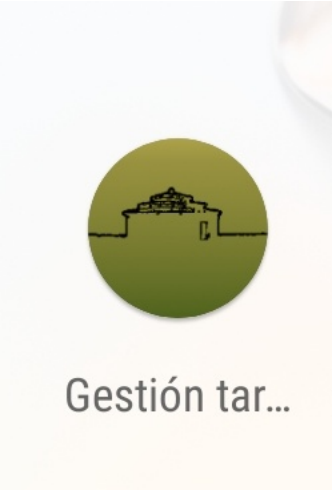

Figura A.58: Icono aplicación Android

Lo que mostrará la siguiente pantalla.

Figura A.59: Pantalla inicial Android

Al igual que la primera pantalla que aparece en el navegador, el objetivo de esta es que los agricultores de identifiquen y puedan acceder al sistema. Para ello sólo tendrán que indicar el correo electrónico y la contraseña con los que se registraron en el sistema. Cuando el usuario haga click en Entrar y si sus credenciales son válidas, el sistema mostrará la siguiente pantalla.

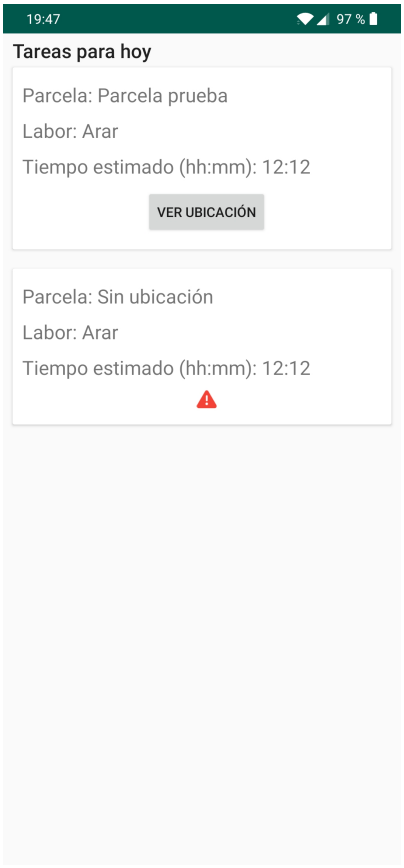

Figura A.60: Tareas estimadas para hoy.

En esta pantalla se muestran las tareas que están estimadas para realizar el día en el que se está haciendo la búsqueda. Si el usuario ha indicado la ubicación de la parcela en la que hay que realizar, la aplicación permitirá ver la ubicación de esta en la aplicación Google Maps haciendo click en Ver ubicación. Si no se ha indicado ubicación de la parcela donde se tiene que realizar la tarea, saldrá una exclamación como ocurre en el segundo caso del ejemplo.