

**UNIVERSIDAD MIGUEL HERNÁNDEZ DE
ELCHE**

ESCUELA POLITÉCNICA SUPERIOR DE ELCHE

**GRADO EN INGENIERÍA INFORMÁTICA EN
TECNOLOGÍAS DE LA INFORMACIÓN**



**"DIGITALIZACIÓN DEL CUADERNO
DE EXPLOTACIÓN PARA LA
PRODUCCIÓN AGRÍCOLA ECOLÓGICA"**

TRABAJO FIN DE GRADO

septiembre - 2020

**AUTOR: Juan Francisco Torres Martínez
DIRECTOR: Eloy Alarcón Ruiz**



A mis padres, a mis hermanas, a mis tíos y a mis amigos, sólo puedo expresar mi más sincero agradecimiento por apoyarme durante todos estos años que hoy culminan, decir que la meta que hoy consigo no es solo mía, sino mitad de todos vosotros y mitad mía.

Gracias a mi pareja Penélope por estar ahí apoyándome, animándome y acompañándome la cual ha sido un pilar fundamental.

Al director de este TFG, D. Eloy Alarcón, quiero agradecerle, todo el apoyo recibido no solo durante este proyecto, sino también durante toda la etapa académica y laboral. Gracias por ser un gran líder.



LISTA DE CONTENIDOS

LISTA DE FIGURAS	7
LISTADO DE TABLAS.....	10
CAPÍTULO 1: INTRODUCCIÓN	11
1.1. MOTIVACIÓN DE LA ELECCIÓN	12
1.2. OBJETIVOS	12
1.3. DESCRIPCIÓN DE LA APLICACIÓN A DESARROLLAR	13
1.4. LÍMITES DEL PROYECTO	14
CAPÍTULO 2: ESTADO DE LA CUESTIÓN.....	15
2.1. AGROPTIMA.....	15
2.2. AGRICOLUM.....	18
2.3. AGROSLAB	20
2.4. CULTIVAPP	22
CAPÍTULO 3: HIPÓTESIS DEL TRABAJO.....	24
3.1. LENGUAJES DE PROGRAMACIÓN	24
3.1.1. <i>REACT NATIVE</i>	24
3.2. SOFTWARE.....	25
3.2.1. <i>NODE</i>	25
3.2.2. <i>NPM</i>	26
3.2.3. <i>YARN</i>	27
3.3. SOFTWARE DE APOYO AL DESARROLLADOR.....	28
3.3.1. <i>GIT</i>	28
3.3.2. <i>GITHUB</i>	28
3.4. LIBRERIAS.....	29
3.4.2. <i>NATIVE BASE</i>	30
3.4.3. <i>COMPONENTES DE REACT NATIVE</i>	30
3.5. SERVIDORES	32
3.5.1. <i>XAMP</i>	32
3.6. HARDWARE.....	33
CAPÍTULO 4: METODOLOGÍA Y RESULTADOS	35
4.1. PLANIFICACIÓN DEL PROYECTO	35
4.2. DIAGRAMAS UML.....	38
4.2.1. <i>DIAGRAMAS DE CASOS DE USO</i>	38
4.2.2. <i>DIAGRAMAS DE SECUENCIA</i>	45
4.2.3. <i>DIAGRAMAS DE ACTIVIDAD</i>	47

4.3.	PROTOTIPADO	49
4.4.	IMPLEMENTACIÓN	53
4.4.1.	<i>COMPONENTES DE REACT NATIVE</i>	53
4.4.2.	<i>CICLO DE VIDA DE UN COMPONENTE</i>	53
4.4.3.	<i>API FETCH</i>	56
4.4.4.	<i>REACT NAVIGATION</i>	62
4.4.5.	<i>EXPO ICONS</i>	64
4.5.	GUÍAS DE ESTILO	64
CAPÍTULO 5: CONCLUSIONES Y TRABAJOS FUTUROS		66
5.1.	CONCLUSIONES	66
5.2.	TRABAJOS FUTUROS	66
5.2.1.	<i>APLICACIÓN WEB</i>	67
5.2.2.	<i>ESTUDIO DE USABILIDAD CON USUARIOS</i>	67
5.2.3.	<i>SEGURIDAD</i>	67
5.2.4.	<i>API MAPS</i>	67
5.2.5.	<i>SISTEMA SCADA</i>	67
CAPÍTULO 6: BIBLIOGRAFÍA		69
ANEXOS		71



LISTA DE FIGURAS

Figura 1. Menú actividades de Agroptima mostrando límites de diseño	16
Figura 2. Pantalla filtrar actividades mostrando los límites de diseño.....	16
Figura 3. Pantalla que lista los terrenos introducidos mostrando los límites de diseño	17
Figura 4. Menú desplegable mostrando los límites de diseño	17
Figura 5. Pantalla de acceso de Agricolium mostrando los límites de diseño	18
Figura 6. Menú cajón mostrando los límites de diseño	19
Figura 7. Pantalla escoge actividad mostrando los límites de diseño	19
Figura 8. Datos que muestra la aplicación Wappalyzer	20
Figura 9. Menú inicio de la aplicación Agroslab.....	21
Figura 10. Pantalla mis recintos Agroslab	21
Figura 11. Pantalla nueva prescripción.....	22
Figura 12. Pantalla inicio de Cultivapp mostrando los límites de diseño	23
Figura 13. Logo React Native	24
Figura 14. Esquema VirtualDOM.....	25
Figura 15. Logo NodeJS	25
Figura 16. Funcionamiento asíncrono NodeJS.....	26
Figura 17. Logo NPM	26
Figura 18. Logo Yarn	27
Figura 19. Logo Git.....	28
Figura 20. Logo GitHub.....	28
Figura 21. Logo Expo.....	29
Figura 22. Logo Native Base.....	30
Figura 23. Componente Date Timer Picker	30
Figura 24. Ejemplo Table Componet	31
Figura 25. Ejemplo de un Componente Chart Kit	31
Figura 26. Logo Xampp.....	32
Figura 27. Interfaz XAMPP en SO Windows	33
Figura 28. Planificación del proyecto.....	36
Figura 29. Diagrama de Gantt	37
Figura 30. Diagrama de casos de uso	39
Figura 31. Diagrama de secuencia. Inicio de sesión	46

Figura 32. Diagrama de secuencia. Registro.....	46
Figura 33. Diagrama de secuencia. Crear terreno	47
Figura 34. Diagrama de secuencia. Mostrar gráficas	47
Figura 35. Diagrama de actividad. Iniciar sesión	48
Figura 36. Diagrama de actividad. Registro de usuario.....	48
Figura 37. Diagrama de actividad. Crear productor	49
Figura 38. Diagrama de actividad. Crear práctica cultural.....	49
Figura 39. Prototipado de la vista crear propietario.....	51
Figura 40. Prototipado de la vista editar productor	51
Figura 41. Prototipado de la vista menú lateral.....	52
Figura 42. Prototipado de la vista listar productores	52
Figura 43. Ciclo de vida con componentes desaconsejados	54
Figura 44. Ciclo de vida con los componentes aconsejados	55
Figura 45. API Fetch crearTerrno.....	57
Figura 46. Constante que indica la url del servidor.....	57
Figura 47. Código php crearTerreno	60
Figura 48. Fucnión registrar componente de clase crear_terreno.js	61
Figura 49. Comandos de npm para instalar React Navigation	62
Figura 50. Comandos de Yarn para instalar React Navigation	62
Figura 51. Constates creadas para agilizar código.....	62
Figura 52. Código del núcleo de la navegación	63
Figura 53. Importación del componente FontAwesome	64
Figura 54. Componente FontAwesome icono user.....	64
Figura 55. Pantalla inicio de sesión	71
Figura 56. Pantalla registro de usuario.....	72
Figura 57. Pantalla Inicio o home del administrador.....	73
Figura 58. Pantalla de inicio o home	73
Figura 59. Pantalla de crear productor	75
Figura 60. Pantalla listar productores.....	76
Figura 61. Pantalla editar productor.....	77
Figura 62. Pantalla crear terreno	78
Figura 63. Pantalla crear práctica cultural.....	79
Figura 64. Pantalla crear abonados.....	80
Figura 65. Pantalla crear control de plagas.....	81

Figura 66. Pantalla crear recolección y venta..... 82
Figura 67. Pantalla donde se muestran las gráficas..... 83
Figura 68. Pantalla donde muestran las tablas de los datos que tiene el productor
..... 84



LISTADO DE TABLAS

Tabla 1. Usuario del sistema	40
Tabla 2. Administrador del sistema	40
Tabla 3. Caso de uso 1. Inicio de sesión	40
Tabla 4. Caso de uso 2. Registro de usuario.....	41
Tabla 5. Caso de uso 3. Crear Productor	41
Tabla 6. Caso de uso 4. Crear terreno	42
Tabla 7. Caso de uso 5. Introducir práctica cultural	42
Tabla 8. Caso de uso 6. Introducir abonado o fertilización	42
Tabla 9. Caso de uso 7. Introducir control de plagas.	43
Tabla 10. Caso de uso 8. Introducir recolección y venta.....	43
Tabla 11. Caso de uso 9. Mostrar gráficas.....	44
Tabla 12. Caso de uso 10. Mostrar tablas.....	44
Tabla 13. Caso de uso 11. Ver productores	44
Tabla 14. Caso de uso 12. Editar productores.	45



CAPÍTULO 1: INTRODUCCIÓN

Actualmente vivimos en una era tecnológica dentro de la Cuarta Revolución Industrial, la cual está acuñada con el nombre de Industria 4.0, este término viene dado por el avance de la digitalización en las empresas. Cada vez son más las empresas que no dudan en ir incorporándose a la Industria 4.0, modificando procesos, métodos, añadiendo el IoT a su industria, por nombrar algunos de esos cambios.

Con este trabajo se quiere poner nuestro granito de arena a esa transformación industrial, escogiendo el campo de las aplicaciones móviles híbridas, las cuales está teniendo una buena acogida en el mundo de las aplicaciones móviles. Cada vez más son las empresas que están optando por soluciones de aplicaciones híbridas, en vez de aplicaciones nativas. Si una empresa tiene que crear una aplicación y pretende abarcar los dos mercados actuales de software, que son:

- iOS [11]: para los sistemas operativos móviles de la multinacional Apple Inc.
- Android [10]: que es el sistema operativo desarrollado por la conocida Google.

Entonces una empresa, debe tener como mínimo dos grupos de desarrolladores o los desarrolladores que tenga la empresa deben tener conocimiento de ambas plataformas. Bien de una forma u otra, el coste de esa aplicación móvil siempre va a ser más cara que la de una híbrida, porque la híbrida solamente necesita de una persona o un grupo de desarrolladores que conozca un lenguaje solo para poder crearla.

Según van pasando los años, las aplicaciones híbridas le van echando un pulso más fuerte a las nativas y esto es gracias a la potencia del hardware, que permite la compilación de un código intermedio y son ejecutadas por una máquina virtual.

Por estos motivos, se ha elegido la opción híbrida para el desarrollo de este proyecto.

1.1. MOTIVACIÓN DE LA ELECCIÓN

Desde la niñez hasta el día de hoy mi relación con el mundo de la agricultura es bastante estrecho. Mi padre posee unos trozos de tierra donde tiene algunos árboles frutales, como almendros, olivos y naranjos. Desde bien pequeño he estado junto a él, realizando todo tipo de tareas agrícolas para el cultivo de esas plantaciones.

Por todas estas circunstancias, vi que la implementación del cuaderno de explotación para la producción agrícola ecológica podía pasar de ser un elemento físico a ser un elemento digital, y así tener esos datos digitalizados, de esa forma se aprovecharían mejor las ventajas de la industria 4.0., ya que uno de sus fuertes son los datos y su tratamiento. No obstante, si se necesitasen los datos físicos, se tendrían a un clic de ratón.

Por todo esto no podía dejar pasar la oportunidad de aportar los conocimientos que me ha brindado el grado de Ingeniería Informática en Tecnologías de la Información y aplicarlos al mundo de la agricultura que me toca de cerca, pese a la importancia que tiene en nuestras vidas, enfrenta a grandes amenazas que abarcan un amplio espectro que van desde los daños naturales, las condiciones laborales de los trabajadores, muchas veces de la falta de ayudas por parte del estado y la falta o nula digitalización en muchos de sus procesos que podrían optimizar recursos y maximizar beneficios.

1.2. OBJETIVOS

Como objetivo principal estamos persiguiendo el poder digitalizar un proceso que pueda facilitar la gestión o administración del cuaderno de campo de explotación para la producción agrícola ecológica, que tras mi experiencia vital, lo hago con el firme convencimiento de que puedo ayudar a muchos agricultores, empezando por mi interés familiar y conociendo la falta de digitalización que tiene este sector.

Y como objetivo secundario, se quiere llevar a cabo la realización de una aplicación móvil híbrida, empleando uno de los framework más potentes y con una de las comunidades más activas y a su vez grande. Esto es debido a que fue desarrollado por la empresa Facebook.

Como meta personal se aspira a ir más allá de una app con simples pantallas y sin tratamiento de datos, sino que se intentará adecuar a un desarrollo lo más profesional posible, aportando valor en todo lo referente a la usabilidad del producto.

Para aportarle más valor al producto, en un futuro se implementaría una aplicación web para que formase el conglomerado digital.

1.3. DESCRIPCIÓN DE LA APLICACIÓN A DESARROLLAR

En este proyecto se va a desarrollar una aplicación móvil híbrida encargada de llevar al día el cuaderno de explotación para la producción agrícola ecológica, de esta forma los datos estarán almacenados digitalmente. Con los datos guardados digitalmente su tratamiento será más eficiente, evitando un consumo desmesurado de papel y ayudando a convertir la gestión en un proceso más claro y fácil de gestionar.

De nuevo comentar, que el proyecto va a utilizar el framework React Native que es uno de los más extendidos entre la comunidad y con proyectos muy grandes a sus espaldas.

La aplicación se estructura de la siguiente forma, donde detallamos por encima que funciones va a tener:

- Registro de usuario, donde el usuario introducirá los datos para crearse una cuenta.
- Acceso individual a la aplicación o Login. En esta pantalla es donde se controla el acceso a la aplicación.
- Home o principal, donde se encuentran todos los enlaces a las acciones pertinentes.
- Alta de terreno.
- Alta práctica cultural.
- Añadir la acción de abono o fertilización.
- Alta del control de plaga que se lleve a cabo.
- Introducción de los datos referidos a la recolección y venta.
- Muestra de gráficas resumiendo los parámetros más importantes.
- Tablas con información detallada de los datos introducidos.

- Todas las pantallas dispondrán de un menú de cajón.

1.4. LÍMITES DEL PROYECTO

Culminando la descripción de lo que se va a llevar a cabo en este proyecto, tenemos que mencionar cual va a ser su alcance. Este proyecto va a ser un proyecto piloto. No se va a poder tocar en la parte de seguridad de la aplicación, el entorno va a ser simulado y no se va a llegar a la fase de producción/monetización donde se subirá la aplicación a las diferentes tiendas de aplicaciones que existen en el mercado y los usuarios podrían disfrutar de ella.



CAPÍTULO 2: ESTADO DE LA CUESTIÓN

En este capítulo nos vamos a encargar de hablar un poco sobre las ofertas que hay en el mercado, relacionadas con aplicaciones que hagan algo similar. Todo el análisis siguiente se ha ido recabando para ver como este proyecto puede aportar un valor diferente.

2.1. AGROPTIMA

Agroptima [1] es una web diseñada para la gestión eficiente de una explotación agrícola. Agroptima tiene desarrollada una APP móvil y una web. La APP cuenta con diferentes apartados donde se van anotando las actividades que se ejecutan, por ejemplo: abonar, prácticas culturales, recolección, etc. Tiene también chat de ayuda, videotutoriales donde se explica cómo manejar los diferentes apartados de la aplicación. Y por medio de la web, una vez que el usuario tiene una cuenta, puede ir viendo y gestionando todos los datos que va introduciendo.

Algunos detalles de software son:

- Las versiones mínimas para la APP son: Android 4.0.3 o iOS 8.
- La información se guarda en la nube donde se accede desde el navegador con la cuenta de usuario correspondiente.

La APP de Agroptima no es una aplicación nativa, ya que al activar en nuestro móvil la opción de desarrollador "mostrar límites de diseño", se puede observar que las supuestas vistas no están delimitadas.

- En su web utiliza una base de datos MySQL.
- Es una solución de pago, por medio de planes.

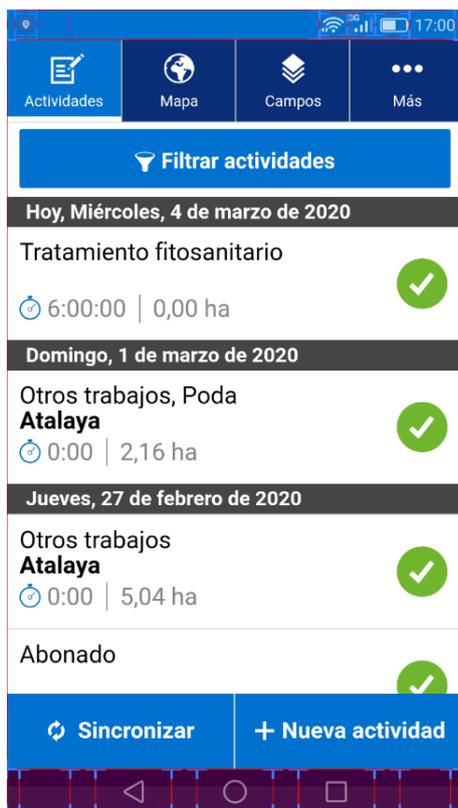


Figura 1. Menú actividades de Agroptima mostrando límites de diseño

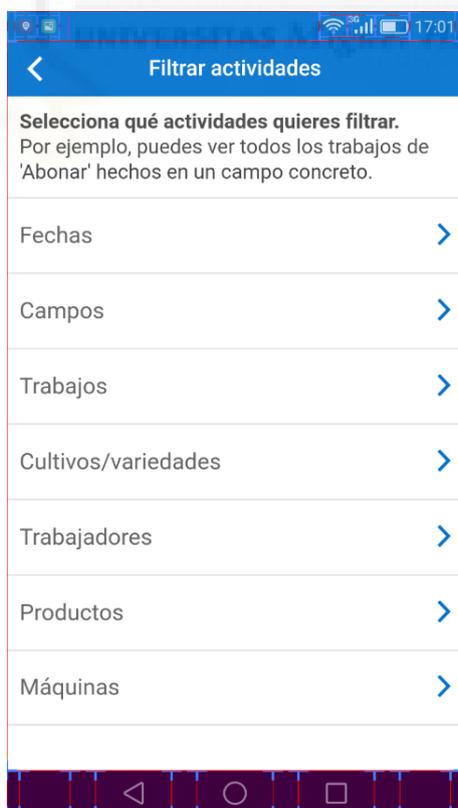


Figura 2. Pantalla filtrar actividades mostrando los límites de diseño

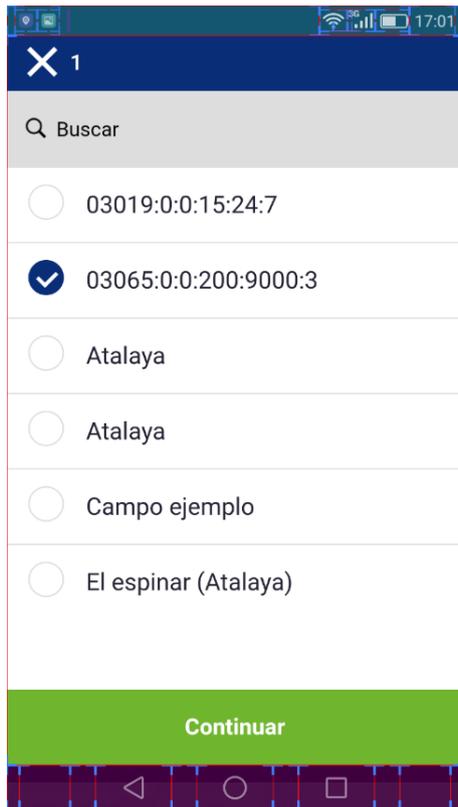


Figura 3. Pantalla que lista los terrenos introducidos mostrando los límites de diseño

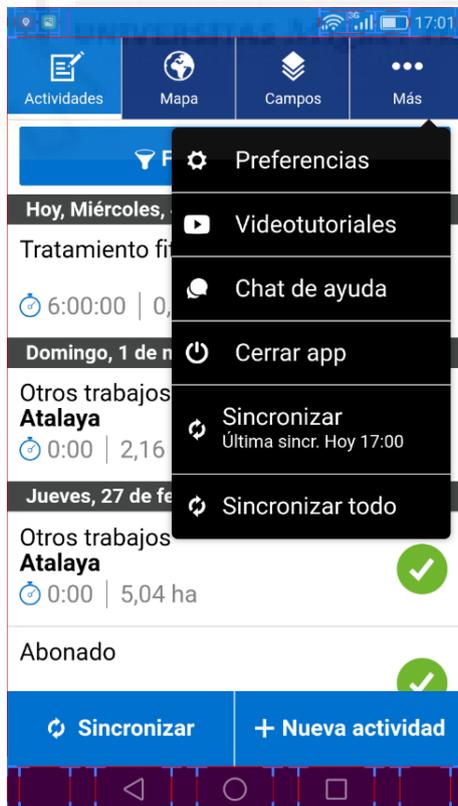


Figura 4. Menú desplegable mostrando los límites de diseño

2.2. AGRICOLUM

Agricolum [2] es una web para la gestión de las explotaciones agrarias. En este caso Agricolum está dotada de una APP en donde se van introduciendo todo tipo de datos relacionado con la explotación y su gestión. Los datos introducidos en la APP se sincronizan y se pueden acceder a ellos desde cualquier sitio, bien con la APP o con la web, claro está siempre logeados.

Agricolum dispone APP para Android e iOS. Solo se ha podido probar la APP Android ya que solo disponemos de un dispositivo Android. La APP en este caso es nativa, para verificarlo, se ha habilitado la opción de "mostrar límites de diseño", así se puede apreciar en las siguientes imágenes que estos límites sí coinciden con las vistas.



Figura 5. Pantalla de acceso de Agricolum mostrando los límites de diseño

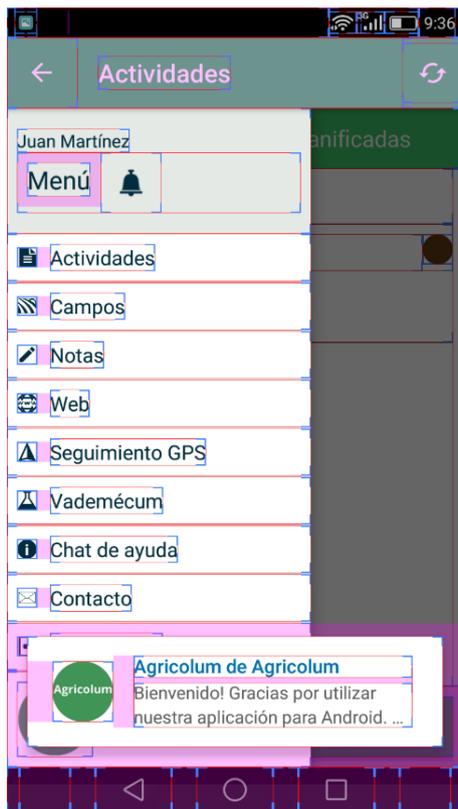


Figura 6. Menú cajón mostrando los límites de diseño



Figura 7. Pantalla escoge actividad mostrando los límites de diseño

Agricolum no utiliza base de datos, gestiona el contenido mediante tablas HTML y utiliza una librería de JavaScript para agregarle controles de interacción avanzados



Figura 8. Datos que muestra la aplicación Wappalyzer

2.3. AGROSLAB

Agroslab [3] es otra plataforma web, la cual tiene implementado numerosas aplicaciones informáticas de gestión orientada a técnicos y asesores, contando con unas herramientas informáticas diseñadas para poder dar respuesta a las necesidades del sector de la agricultura. Dispone de APPs tanto para Android como iOS.



Figura 9. Menú inicio de la aplicación Agroslab

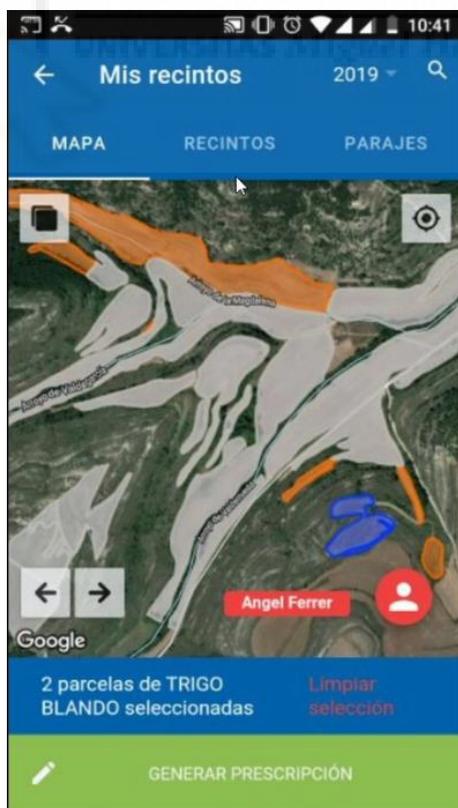


Figura 10. Pantalla mis recintos Agroslab



Figura 11. Pantalla nueva prescripción

2.4. CULTIVAPP

Cultivapp [4] es una aplicación para registrar y monitorizar las actividades que a diario se llevan a cabo en una explotación agraria. Esta aplicación de nuevo es una aplicación híbrida, ya que al habilitar en el móvil que muestre los límites de diseño, estos no coinciden con los límites de cada elemento que nos muestra. Tras la revisión de la aplicación, la encontramos intuitiva, pero según se va interactuando con ella, vemos que se nos hace un poco liosos el poder llegar donde queremos. Dentro de la versión gratuita de CultivAPP que está orientada a los agricultores, nos incorpora publicidad, la cual ocupa toda la pantalla y cuesta un poco quitarla, dado que la "x" de salida es muy pequeña y parece que está deshabilitada unos segundos para que no funcione mientras duren estos.



Figura 12. Pantalla inicio de Cultivapp mostrando los límites de diseño

Tras el testeo de estas aplicaciones he comprobado que la realización de sencillas tareas como la de dar de alta un terreno, pese a mis conocimientos en el mundo de la agricultura y mis conocimientos tecnológicos, se convierte en una tarea ardua y difícil de realizar. Por ello, una de mis principales motivaciones es la de proveer de una aplicación que facilite las acciones más habituales que se realizan en la gestión de un cuaderno de explotación para la producción agrícola ecológica.

Por otro lado, la evidente falta de digitalización en la ingente gestión de papel también es suficiente motivo para proponer una solución informática para un dispositivo móvil en pos de perseguir una medida ecoeficiente que ayude a la conservación de recursos naturales del planeta.

CAPÍTULO 3: HIPÓTESIS DEL TRABAJO

Para el desarrollo de este trabajo se han empleado diversos recursos, entre los que tenemos lenguajes de programación, servidores, software de simulación, librerías y software de apoyo al desarrollo, además, de elementos de hardware. A continuación, vamos a ver para que han sido utilizados cada uno de estos recursos.

3.1. LENGUAJES DE PROGRAMACIÓN

3.1.1. REACT NATIVE



Figura 13. Logo React Native

React Native [9] es una de las tecnologías para el desarrollo de aplicaciones móviles para Android, iOS, Web y UWP con una comunidad muy grande y activa. Esta tecnología fue creada por Facebook y lanzada por primera vez el 26 de marzo de 2015. Este marco de código abierto está escrito en: JavaScript, Java, C++, Objective-C, Objective-C++ y Python.

React Native tiene casi los mismos principios operativos que React, excepto que React Native no maneja el DOM por medio del DOM virtual. Efectúa un procesamiento en segundo plano directamente en el terminal y se comunica con la plataforma raíz a través de un puente intermedio, asíncrono y por lotes.

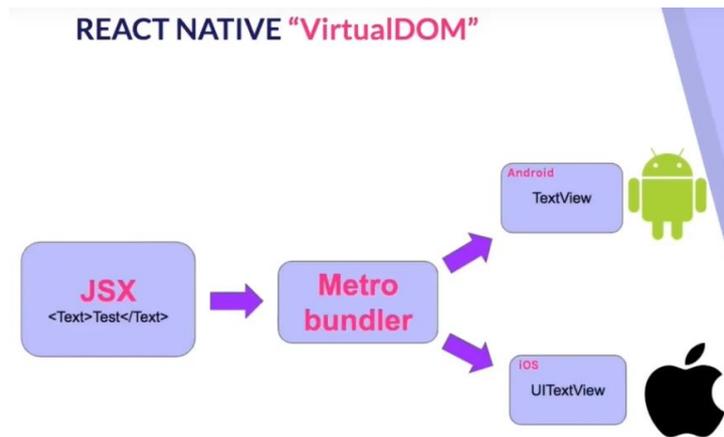


Figura 14. Esquema VirtualDOM

React utiliza componentes que encapsulan el código nativo e interactúan con la API nativa por medio del modelo declarativo de interfaz de usuario de React y Javascript. Con esto el desarrollo de aplicaciones para múltiples plataformas es más ágil.

3.2. SOFTWARE

3.2.1. NODE



Figura 15. Logo NodeJS

NodeJS es un entorno de ejecución para Javascript, ¿y esto qué quiere decir? Para entenderlo es necesario saber que Javascript era un lenguaje de programación del lado del cliente que solo podía ser ejecutado sobre un navegador web, sin embargo, esto dejó de ser así en 2009, año en el que Ryan Dhal presentó un entorno con todo lo necesario para ejecutar Javascript fuera de un navegador web, dejando el concepto de este original lenguaje de programación obsoleto.

Es llevado a cabo gracias a V8, que es un motor de JavaScript de código abierto desarrollado por Google para su navegador Chrome, este poderoso motor puede ejecutar JavaScript a una velocidad impresionante, además la ejecución es independiente de los navegadores, una ventaja que Ryan Dhal aprovechó incorporándolo al núcleo de Node. También es importante destacar que Node no

usa un modelo síncrono, el cuál retrasaría o bloquearía procesos, si no que utiliza un sistema de operaciones asíncrono y orientado a eventos, en consecuencia, se pueden ejecutar a la vez sin esperar a que termine la tarea predecesora. Encima NodeJS cuenta con su propio gestor de paquetes llamado NPM (Node Package Manager) que lo explicaremos más adelante.

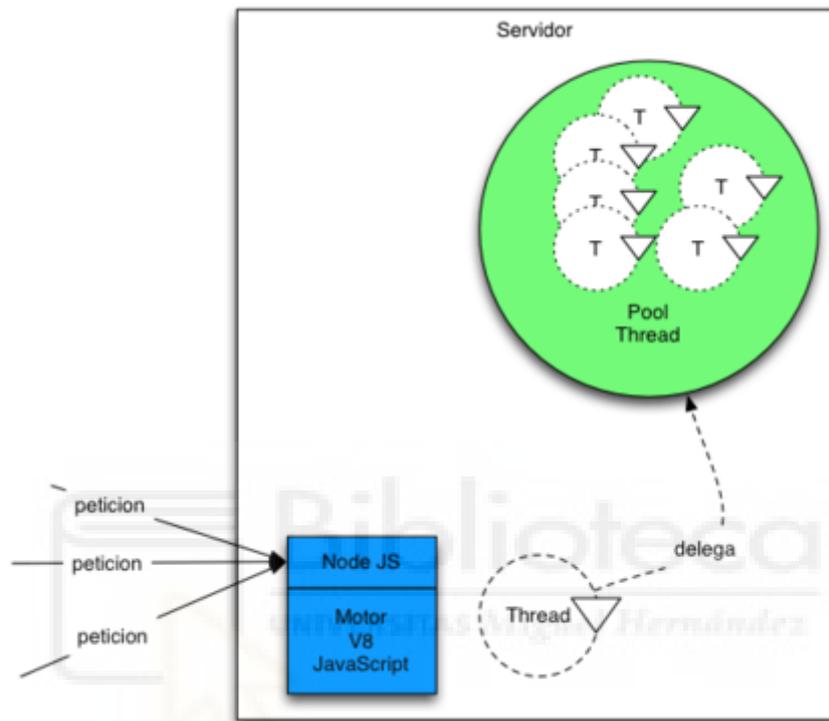


Figura 16. Funcionamiento asíncrono NodeJS

3.2.2. NPM



Figura 17. Logo NPM

NPM es un gestor de dependencias para Javascript. Es el gestor por defecto en el ecosistema de Nodejs y viene empaquetado con el. Con el tiempo ha conseguido ser la herramienta más conocida popular para manejar paquetes de Javascript.

El gestor de dependencias está compuesto por tres componentes principales:

- La página web: Desde la web se pueden descubrir nuevos paquetes y alojar los tuyos propios de forma pública o privada (mediante pago).
- La interfaz de línea de comandos: La forma en que los desarrolladores interactúan con NPM.
- El registro: Considerado el ecosistema de librerías más grande del mundo.

Los principales usos de NPM son:

- Agregar dependencias a nuestro proyecto en el sitio correcto para que nodejs pueda encontrarlos.
- Manejar diferentes versiones de dependencias de código.
- Actualizar dependencias de forma sencilla.
- Descargar Standalone tools.
- Crear nuestros propios paquetes y compartirlos con cualquier usuario de NPM.
- Descubrir a otros desarrolladores que estén abordando nuestros mismos problemas.
- No reinventar la rueda.

3.2.3. YARN



Figura 18. Logo Yarn

Yarn es un gestor de paquetes de Javascript, este fue lanzado por Facebook y Google, Exponent y Tilde. Su creación fue debido al crecimiento de Facebook ya que este utilizaba el famoso gestor de paquetes NPM, pero al ir creciendo el volumen de código de la compañía y a su vez se fue incrementando el número de ingenieros y programadores que participaban en ese código, los problemas de consistencia, rendimiento y seguridad también se fueron incrementando en paralelo. Y por ello decidieron crear su propio gestor de paquetes, para que así se gestionara sus dependencias con mayor fiabilidad.

3.3. SOFTWARE DE APOYO AL DESARROLLADOR

3.3.1. GIT



Figura 19. Logo Git

Una herramienta fundamental [14] que no le puede faltar a un desarrollador tanto principiante como profesional, es un software de control de versiones. Con un software de control de versiones un desarrollador esta al tanto de todos sus cambios a lo largo del desarrollo de su código. Y aún se hace más esencial, si hay más de un desarrollador desarrollando en el mismo código. En este proyecto se ha utilizado el software de control de versiones Git, conjuntamente con la plataforma GitHub.

Git es una herramienta que realiza las funciones de control de versiones de código de forma distribuida, como hemos comentado anteriormente, permite tener el control de todos los cambios realizados en el código, además permite, la creación de ramas y retroceder a un estado anterior del proyecto. Todos los cambios son almacenados en plataformas como GitLab o GitHub que bien pueden ser públicos o privados, hasta hace poco, en GitHub para almacenar proyectos privados se necesitaba una licencia.

3.3.2. GITHUB



Figura 20. Logo GitHub

GitHub es una de las plataformas para que los desarrolladores suban el código de sus aplicaciones y herramientas, vamos un repositorio de código. El código que no sea privado se puede descargar y revisar por cualquier usuario que desee.

Otra cosa que se puede hacer es ver la documentación del proyecto, ver las colaboraciones, fechas, etc.

3.4. LIBRERIAS

3.4.1. EXPO

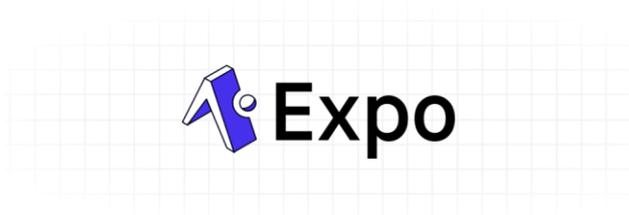


Figura 21. Logo Expo

Expo sdk [18] es una librería que nos permite construir aplicaciones de React Native pero sin cambiar código nativo. Nos referimos al código nativo como código de iOS y código Android. Con lo cual solo se desarrolla en Javascript y React Native.

Con Expo se pueden generar muy rápido los bundles que van a producción y con ello subir la app a las diferentes plataformas de descarga. También cuenta con una característica llamada Snack la cual te permite desarrollar código a través de la web, los snack son pedazos de código o proyectos que subes donde puedes ejecutar en Android, iOS o en el navegador.

Expo cuenta con una app, mediante la cual podemos ejecutar e ir depurando la aplicación que estemos desarrollando, según vamos escribiendo código y guardando los cambios, estos se van actualizando y mostrándolos.

3.4.2. NATIVE BASE

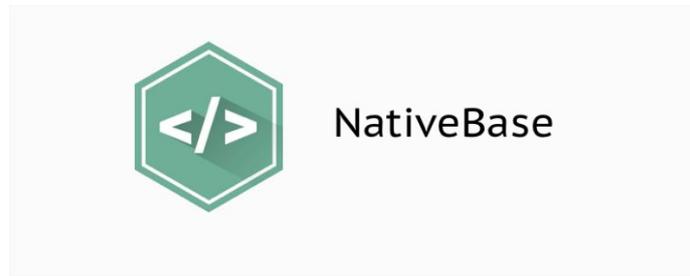


Figura 22. Logo Native Base

Native Base [19] es una librería UI de código abierto y gratuita para React Native y Vue Native. Con ella se construyen aplicaciones nativas para las plataformas de iOS y Android. Native Base soporta webs desde su versión 2.4.1.

3.4.3. COMPONENTES DE REACT NATIVE

React Native [17] provee de multitud de componentes que son utilizados para el desarrollo de las aplicaciones, a pesar de ello siempre uno necesita otros componentes que pueden no estar en el núcleo de React native, por eso la comunidad, según va teniendo necesidades de ellos los va creando.

En este proyecto se han necesitado algunos, los cuales se describen más adelante.

3.4.3.1. React Native Date Timer Picker

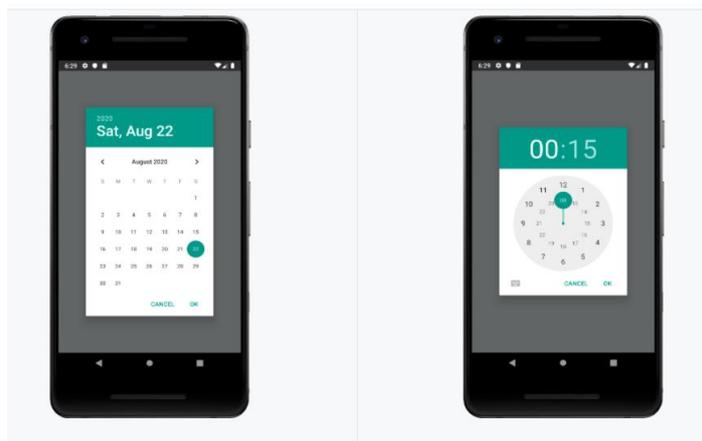
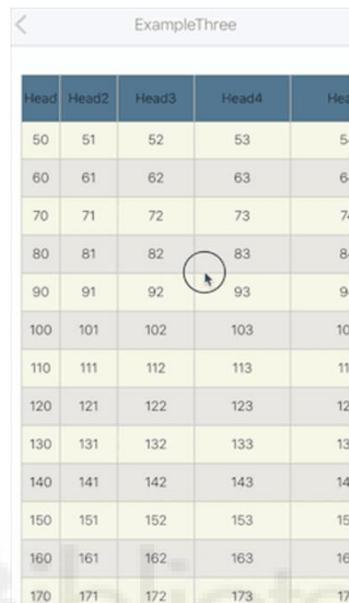


Figura 23. Componente Date Timer Picker

Este componente es un selector de fecha, el cual se puede instalar bien con el gestor de paquetes npm o bien con yarn. En su repositorio de Github muestra

toda la información, como las entradas arbitrarias llamas "props" y demás información necesaria para su adaptación en proyectos.

3.4.3.2. React Native Table Component



Head	Head2	Head3	Head4	Head5
50	51	52	53	54
60	61	62	63	64
70	71	72	73	74
80	81	82	83	84
90	91	92	93	94
100	101	102	103	104
110	111	112	113	114
120	121	122	123	124
130	131	132	133	134
140	141	142	143	144
150	151	152	153	154
160	161	162	163	164
170	171	172	173	174

Figura 24. Ejemplo Table Component

Table Component [15] es un componente para crear tablas, de nuevo, en su plataforma de Github tenemos una amplia información para poder implementar dicho componente en proyectos. Dentro de la documentación, podemos encontrar el proceso de instalación, que de nuevo es con el gestor de paquetes npm, ejemplos de tablas, las propiedades de estas, que licencia tiene ese componente y algunos datos más.

3.4.3.3. React Native Chart Kit

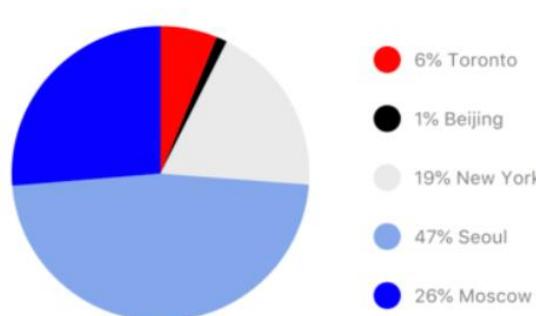


Figura 25. Ejemplo de un Componente Chart Kit

Chart Kit [12] es un componente que tiene implementados diferentes formatos de gráficos, desde gráficos de barras hasta gráficos circulares. Alojado en el repositorio de Github, uno puede ver la documentación e implementar el o los componentes que se adapten a su proyecto. Chart Kit detalla entre otras cosas el modo de instalación, configuraciones de los diferentes gráficos, enlaces para más información y contribuidores.

3.5. SERVIDORES

3.5.1. XAMP



Figura 26. Logo Xampp

XAMPP [20] es un conjunto de software libre con licencia GNU, que simulan un entorno online sin acceso a internet, en el que un desarrollador puede probar su proyecto web, app, etc, sin la necesidad de adquirir servidores, dominios, bases de datos, etc. En resumen, ejerce las funciones de un servidor web libre, con un uso sencillo y pudiendo mostrar en el navegador páginas dinámicas.

El nombre de XAMPP es un acrónimo de los componentes que lo forman; Apache, MySQL/MariaDB, PHP, Perl y la X viene a representar los sistemas operativos Linux, Windows y Mac OS X.

1. Apache: es un servidor web de código abierto cuyo su principal uso es para enviar páginas web estáticas y dinámicas en la World Wide Web.
2. MySQL/MariaDB: es un sistema relacional de gestión de bases de datos. En las últimas versiones de XAMPP ha sido sustituida por MariaDB.
3. PHP: es el lenguaje más usado en el lado del servidor o back end, este código es de código abierto y con el se pueden construir páginas webs o aplicaciones dinámicas.

es un lenguaje de programación de código abierto en el lado del servidor, el cual consigue crear páginas webs o aplicaciones dinámicas.

- Perl: es un lenguaje de programación usado en la gestión del sistema, en el desarrollo web y también en la programación de la red. Este lenguaje también permite la programación de webs dinámicas.

A parte de estos componentes, XAMPP incluye, dependiendo del sistema operativo que se esté usando, herramientas como servidores de correo, herramientas para la gestión de las bases de datos, software de analítica web, herramientas para ayudar al sistema a implementar el Secure Socket Layer (SSL), también incluyen servidores con el protocolo FTP y algo más.

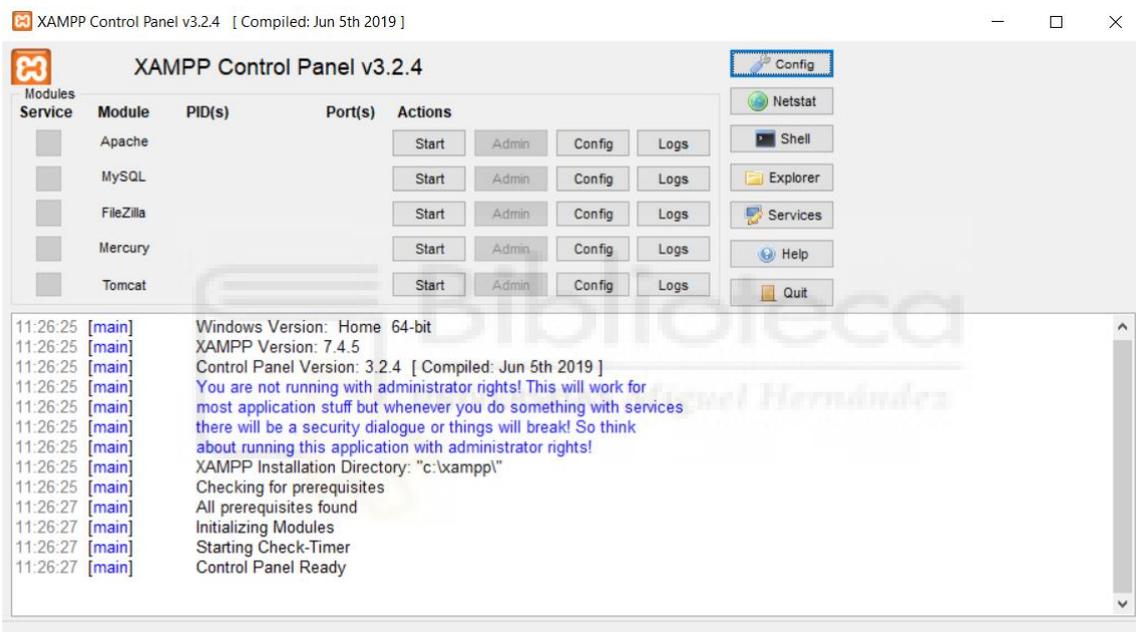


Figura 27. Interfaz XAMPP en SO Windows

3.6. HARDWARE

En el desarrollo de este proyecto se han utilizado un ordenador portátil, monitor, teclado, ratón y un móvil. A continuación, se detallan las especificaciones del hardware:

Ordenador Portátil	
Marca	Acer
Modelo	Aspire7 A715-71G-727N
Procesador	Intel® Core™ i7-7700HQ CPU @ 2.80GHz 2.81 GHz
Memoria instalada	8,00 GB (7,89 GB utilizable)
Tipo de Sistema	Sistema operativo de 64 bits, procesador x64

Sistema operativo	Windows 10 Home
-------------------	-----------------

Monitor	
Marca	LG
Modelo	22MP58VQ – PB.AEUKEVN

Teclado	
Marca	Logitech
Modelo	k-120

Ratón	
Marca	Logitech
Modelo	B100

Teléfono Movil	
Marca	Huawei
Modelo	VNS
Número de compilación	VNS-L31C900B161
Versión de EMUI	EMUI 4.1.1
CPU	Kirin 650
RAM	3,0 GB
Memoria del teléfono	16,00 GB



CAPÍTULO 4: METODOLOGÍA Y RESULTADOS

4.1. PLANIFICACIÓN DEL PROYECTO

Llegados a este punto del presente Trabajo Final de Grado, nos encontramos frente a un desarrollo de una aplicación móvil híbrida, la cual está cargada de una complejidad elevada, dado la cantidad de componentes de núcleo y de componentes externos, la parte cliente y la parte servidor encadenando una conexión a través de sockets para consumir datos de la API que conecta con la base de datos, necesaria para el tratamiento de los datos y su funcionamiento lógico. Por todo esto es necesario la planificación del proyecto para que el desarrollo sea ágil y eficiente.

En cuanto a la planificación del proyecto existen multitud de programas y métodos que se pueden aplicar, aunque el más utilizado y que nos brinda mejores resultados sea el diagrama de Gantt.

Un diagrama de Gantt es un cronograma que se vale para planear y subsecuentemente informar del progreso dentro del entorno del proyecto, cuantificando tiempos y recursos. La finalidad de la planificación de los proyectos es crear un plan de proyecto y que un gestor de proyectos pueda usar para acompañar el progreso de su equipo.

Nombre	Fecha de inicio	Fecha de fin
• Elección y Documentación de la propuesta	17/02/20	28/02/20
• Analizar APPs similares	2/03/20	20/03/20
• Realización de un curso de React Native	23/03/20	27/03/20
• Realización de un curso de JavaScript	30/03/20	2/04/20
• Configuración del entorno	3/04/20	10/04/20
• Instalación de Expo	3/04/20	3/04/20
• Instalación Git	7/04/20	7/04/20
• Instalación y Configuración Visual Studio Code	6/04/20	6/04/20
• Instalación y Configuración de XAMPP	9/04/20	10/04/20
• Prototipado de la aplicación	13/04/20	14/04/20
• Desarrollo parte de logueo y registro	15/04/20	21/04/20
• Codificación del componente Login	15/04/20	21/04/20
• Codificación del componente Registro	17/04/20	21/04/20
• Codificación de la navegación entre el componente Login y Re...	22/04/20	27/04/20
• Diseño del modelo Entidad-Relación	28/04/20	28/04/20
• Creación de la base de datos	29/04/20	29/04/20
• Establecer la comunicación entre XAMPP y la API de Registro	30/04/20	30/04/20
• Desarrollo una vez logueado el usuario	1/05/20	20/05/20
• Codificación del componente Prácticas Culturales	1/05/20	4/05/20
• Codificación del componente Productores	5/05/20	6/05/20
• Codificación alta de usuarios	7/05/20	8/05/20
• Codificación del componente Inicio	11/05/20	13/05/20
• Codificación del componente Abono y Fertilizantes	13/05/20	14/05/20
• Codificación del componente Control de plagas	15/05/20	18/05/20
• Codificación del componente Recolección y Venta	19/05/20	20/05/20
• Modificación lógica al dar de alta un terreno	21/05/20	22/05/20
• Modificación del modelo Entidad-Relación	25/05/20	25/05/20
• Modificación de la base de datos	26/05/20	27/05/20
• Codificación del componente Drawer Layout Android	28/05/20	1/06/20
• Codificación del componente Gráficas	2/06/20	8/06/20
• Sincronización del componente Gráficas con la API	2/06/20	8/06/20
• Codificación del componente Tablas	9/06/20	16/06/20
• Sincronización del componente Tablas con la API	9/06/20	16/06/20
• Preparación guía visual	17/06/20	22/06/20
• Desarrollo del estilo	23/06/20	29/06/20
• Documentación	17/02/20	29/06/20

Figura 28. Planificación del proyecto

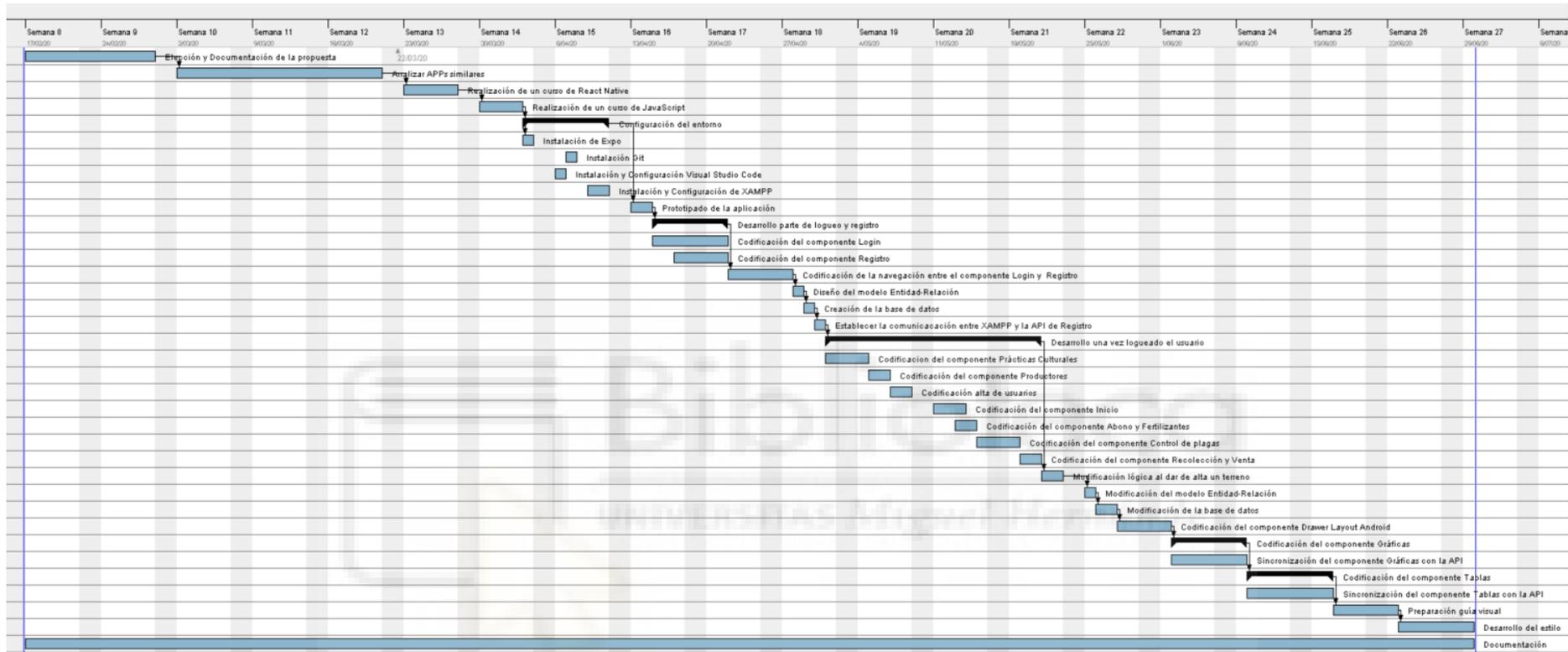


Figura 29. Diagrama de Gantt

4.2. DIAGRAMAS UML

Los diagramas UML (lenguaje unificado de modelado) [5] [], están integrados en las materias de la Ingeniería y se encargan de describir un sistema como una representación formalizada del mismo. Las personas no pueden procesar toda la información que se desee en un momento dado, ya que cuando se trabaja en un problema complejo es necesario dividirlo en porciones más pequeñas y manejables, por ello se construyen los modelos, para cubrir esa necesidad.

En este proyecto, como es de suponer, se van a usar diferentes tipos de diagramas que nos van a ayudar a entender mejor todos los aspectos del sistema.

4.2.1. DIAGRAMAS DE CASOS DE USO

Los casos de uso [8] nos brindan una descripción detallada de los pasos que van a seguir los usuarios a la hora de llevar correctamente una tarea determinada.

Antes de desarrollar las tareas, se explicará quien son los actores que intervienen en el sistema y después le seguirán los correspondientes casos de uso que estén definidos.



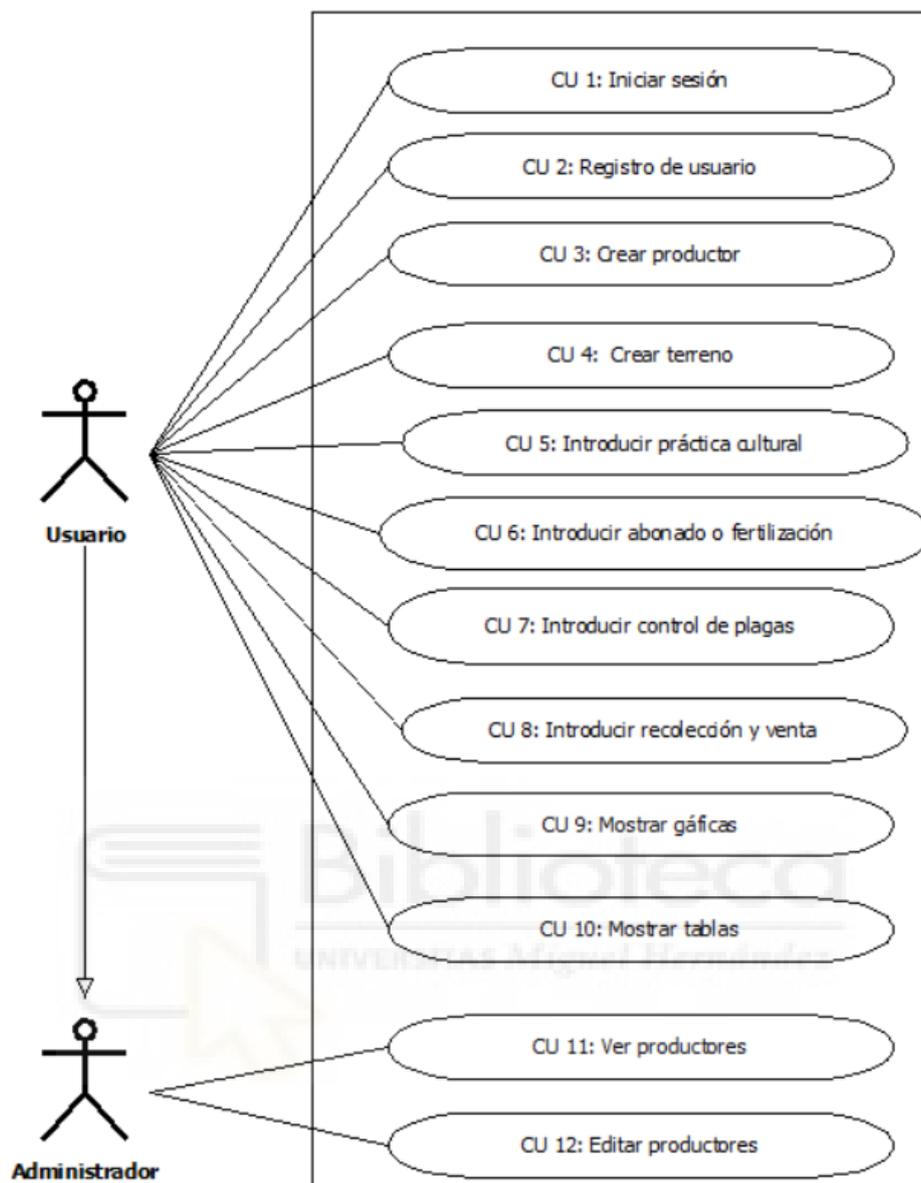


Figura 30. Diagrama de casos de uso

Actor	Usuario
Descripción	El usuario es cualquier persona que tenga o este encargado de la producción agrícola ecológica y quiera digitalizar el cuaderno de explotación para la producción agrícola ecológica.
Casos de uso relacionados	<ul style="list-style-type: none"> • CU 1: Iniciar sesión • CU 2: Registro de usuario • CU 3: Crear Productor • CU 4: Crear terreno • CU 5: Introducir práctica cultural • CU 6: Introducir abonado o fertilización

	<ul style="list-style-type: none"> • CU 7: Introducir control de plagas • CU 8: Introducir recolección y venta • CU 9: Mostrar gráficas • CU 10: Mostrar tablas
--	---

Tabla 1. Usuario del sistema

Actor	Administrador
Descripción	El administrador tiene todos los permisos en la aplicación y es el encargado de diagnosticar problemas de software, realizar copias de seguridad de datos y gestionar a los usuarios.
Casos de uso relacionados	<ul style="list-style-type: none"> • CU 1: Iniciar sesión • CU 2: Registro de usuario • CU 3: Crear Productor • CU 4: Crear terreno • CU 5: Introducir práctica cultural • CU 6: Introducir abonado o fertilización • CU 7: Introducir control de plagas • CU 8: Introducir recolección y venta • CU 9: Mostrar gráficas • CU 10: Mostrar tablas • CU 11: Ver productores • CU 12: Editar productores

Tabla 2. Administrador del sistema

C.U. 1	Iniciar sesión
Actores	Usuario
Descripción	Acceder dentro de la aplicación
Precondición	El usuario debe de haberse registrado anteriormente.
Secuencia normal	<p>P1- Tocamos donde pone usuario e introducimos él nombre de usuario con el que nos hayamos registrado.</p> <p>P2- Tocamos donde pone contraseña e introducimos nuestra contraseña.</p> <p>P3- Pulsamos el botón de "Entrar".</p>
Poscondición	
Excepciones	Si en P3 no se rellena todos los espacios o algún dato de los introducidos no es válido, vuelven a quedarse los campos usuario y contraseña en blanco para intentarlo de nuevo.

Tabla 3. Caso de uso 1. Inicio de sesión

C.U. 2	Registro de usuario
Actores	Usuario
Descripción	Proceso donde un usuario se da de alta en la aplicación

Precondición	
Secuencia normal	<p>P1- Tocamos donde pone correo electrónico e introducimos el correo electrónico que queremos asociar a dicha aplicación.</p> <p>P2- Tocamos donde pone usuario e introducimos el nombre de usuario con el que nos queramos registrar.</p> <p>P3- Tocamos donde pone contraseña e introducimos la contraseña que queremos usar en esta aplicación.</p> <p>P4- Pulsamos el botón de "Guardar".</p>
Excepciones	Si en P4 algún campo está vacío, no se guarda y le redirecciona a la pantalla de inicio de sesión.

Tabla 4. Caso de uso 2. Registro de usuario

C.U. 3	Crear productor
Actores	Usuario
Descripción	Proceso donde el usuario introduce los datos para darse de alta como productor.
Precondición	El usuario debe de haberse registrado anteriormente y ser la primera vez que se loguea.
Secuencia normal	<p>P1- Tocamos donde pone usuario e introducimos el nombre de usuario con el que nos hayamos registrado.</p> <p>P2- Tocamos donde pone contraseña e introducimos nuestra contraseña.</p> <p>P3- Pulsamos el botón de "Entrar".</p> <p>P4- Introducimos los datos que nos piden y pulsamos el botón de "Guardar".</p> <p>P5- Entra en la página principal o home de la aplicación.</p>
Excepciones	Si P3 no es la primera vez, pasa directamente a P5

Tabla 5. Caso de uso 3. Crear Productor

C.U. 4	Crear terreno
Actores	Usuario
Descripción	Dar de alta un terreno nuevo para que se almacene en la base de datos.
Precondición	El usuario debe de haberse registrado anteriormente y haberse logueado.
Secuencia normal	<p>P1- Tocamos donde pone usuario e introducimos el nombre de usuario con el que nos hayamos registrado.</p> <p>P2- Tocamos donde pone contraseña e introducimos nuestra contraseña.</p> <p>P3- Pulsamos el botón de "Entrar".</p> <p>P4- Dentro de la pantalla principal pulsamos en "CREAR TERRENO".</p> <p>P5- Introducimos los datos que nos piden para dar de alta un terreno.</p>

	P6- Una vez que están los datos introducidos, pulsamos el botón de "Guardar".
Excepciones	Si P6 falla en la inserción de los datos es redirigido a P4.

Tabla 6. Caso de uso 4. Crear terreno

C.U. 5	Introducir práctica cultural
Actores	Usuario
Descripción	Dar de alta una práctica cultural nueva para que se almacene en la base de datos.
Precondición	El usuario debe de haberse registrado anteriormente y haberse logueado.
Secuencia normal	P1- Tocamos donde pone usuario e introducimos él nombre de usuario con el que nos hayamos registrado. P2- Tocamos donde pone contraseña e introducimos nuestra contraseña. P3- Pulsamos el botón de "Entrar". P4- Dentro de la pantalla principal pulsamos en "PRÁCTICAS CULTURALES". P5- Introducimos los datos que nos piden para dar de alta una práctica cultural nueva. P6- Una vez que están los datos introducidos, pulsamos el botón de "Guardar".
Excepciones	Si P6 falla en la inserción de los datos es redirigido a P4.

Tabla 7. Caso de uso 5. Introducir práctica cultural

C.U. 6	Introducir abonado o fertilización
Actores	Usuario
Descripción	Dar de alta una práctica de abonado o fertilización nueva para que se almacene en la base de datos.
Precondición	El usuario debe de haberse registrado anteriormente y haberse logueado.
Secuencia normal	P1- Tocamos donde pone usuario e introducimos él nombre de usuario con el que nos hayamos registrado. P2- Tocamos donde pone contraseña e introducimos nuestra contraseña. P3- Pulsamos el botón de "Entrar". P4- Dentro de la pantalla principal pulsamos en "ABONO Y FERTILIZANTES". P5- Introducimos los datos que nos piden para dar de alta un nuevo abonado. P6- Una vez que están los datos introducidos, pulsamos el botón de "Guardar".
Excepciones	Si P6 falla en la inserción de los datos es redirigido a P4.

Tabla 8. Caso de uso 6. Introducir abonado o fertilización

C.U. 7	Introducir control de plagas
Actores	Usuario
Descripción	Dar de alta una práctica de control de plagas nueva para que se almacene en la base de datos.
Precondición	El usuario debe de haberse registrado anteriormente y haberse logueado.
Secuencia normal	P1- Tocamos donde pone usuario e introducimos él nombre de usuario con el que nos hayamos registrado. P2- Tocamos donde pone contraseña e introducimos nuestra contraseña. P3- Pulsamos el botón de "Entrar". P4- Dentro de la pantalla principal pulsamos en "CONTROL DE PLAGAS". P5- Introducimos los datos que nos piden para dar de alta un nuevo control de plaga. P6- Una vez que están los datos introducidos, pulsamos el botón de "Guardar".
Excepciones	Si P6 falla en la inserción de los datos es redirigido a P4.

Tabla 9. Caso de uso 7. Introducir control de plagas.

C.U. 8	Introducir recolección y venta
Actores	Usuario
Descripción	Dar de alta una recolección y venta nueva para que se almacene en la base de datos.
Precondición	El usuario debe de haberse registrado anteriormente y haberse logueado.
Secuencia normal	P1- Tocamos donde pone usuario e introducimos él nombre de usuario con el que nos hayamos registrado. P2- Tocamos donde pone contraseña e introducimos nuestra contraseña. P3- Pulsamos el botón de "Entrar". P4- Dentro de la pantalla principal pulsamos en "RECOLECCIÓN Y VENTA". P5- Introducimos los datos que nos piden para dar de alta la recolección y venta. P6- Una vez que están los datos introducidos, pulsamos el botón de "Guardar".
Excepciones	Si P6 falla en la inserción de los datos es redirigido a P4.

Tabla 10. Caso de uso 8. Introducir recolección y venta.

C.U. 9	Mostrar gráficas
Actores	Usuario

Descripción	Mostrar los datos de unas determinadas tablas de la base de datos en gráficas para una visualización de los datos rápida.
Precondición	El usuario debe de haberse registrado anteriormente y haberse logueado.
Secuencia normal	P1- Tocamos donde pone usuario e introducimos él nombre de usuario con el que nos hayamos registrado. P2- Tocamos donde pone contraseña e introducimos nuestra contraseña. P3- Pulsamos el botón de "Entrar". P4- Dentro de la pantalla principal pulsamos en "GRÁFICAS". P5- Se ven las gráficas.

Tabla 11. Caso de uso 9. Mostrar gráficas.

C.U. 10	Mostrar tablas
Actores	Usuario
Descripción	Muestra de los datos que están almacenados en la base de datos en formato de tabla.
Precondición	El usuario debe de haberse registrado anteriormente y haberse logueado.
Secuencia normal	P1- Tocamos donde pone usuario e introducimos él nombre de usuario con el que nos hayamos registrado. P2- Tocamos donde pone contraseña e introducimos nuestra contraseña. P3- Pulsamos el botón de "Entrar". P4- Dentro de la pantalla principal pulsamos en "TABLAS". P5- Se ven las tablas.

Tabla 12. Caso de uso 10. Mostrar tablas.

C.U. 11	Ver productores
Actores	Administrador
Descripción	Mostrar todos los productores que hay registrados en la aplicación.
Precondición	El usuario debe de tener los permisos de administrador y haberse logueado.
Secuencia normal	P1- Tocamos donde pone usuario e introducimos él nombre del usuario administrador. P2- Tocamos donde pone contraseña e introducimos la contraseña de administrador. P3- Pulsamos el botón de "Entrar". P4- Dentro de la pantalla principal pulsamos en el botón de "PRODUCTORES". P5- Vemos los productores que hay en la base de datos.

Tabla 13. Caso de uso 11. Ver productores

C.U. 12	Editar productores
Actores	Administrador
Descripción	Mostrar todos los productores que hay registrados en la aplicación.
Precondición	El usuario debe de tener los permisos de administrador y haberse logueado.
Secuencia normal	<p>P1- Tocamos donde pone usuario e introducimos él nombre del usuario administrador.</p> <p>P2- Tocamos donde pone contraseña e introducimos la contraseña de administrador.</p> <p>P3- Pulsamos el botón de "Entrar".</p> <p>P4- Dentro de la pantalla principal pulsamos en el botón de "PRODUCTORES".</p> <p>P5- Vemos los productores que hay en la base de datos y pulsamos sobre el que queramos editar.</p> <p>P6- Editamos los datos que necesitamos y pulsamos el botón de "Guardar".</p>

Tabla 14. Caso de uso 12. Editar productores.

Con estas tablas podemos observar la descripción de los principales casos de uso de la aplicación para llevar una idea del desarrollo que debemos llevar, aunque también tiene un gran peso la realización del modelado de la información que es enviada y recibida por los sockets y API de React Native.

4.2.2. DIAGRAMAS DE SECUENCIA

Un diagrama de secuencia sirve para modelar la interacción entre los objetos/actores de nuestro sistema, de este modo identifica la comunicación (mensajes) entre los mismos y las operaciones (métodos) de las clases para resolver un servicio.

La notación UML que es la que vamos a utilizar en el diagrama de secuencia, encontramos al actor con el símbolo de la persona, que es el que inicia el diagrama de secuencia, los objetos, en vertical van las líneas de vida, en horizontal están los mensajes y las cajas de activación que nos indican la resolución de un servicio.

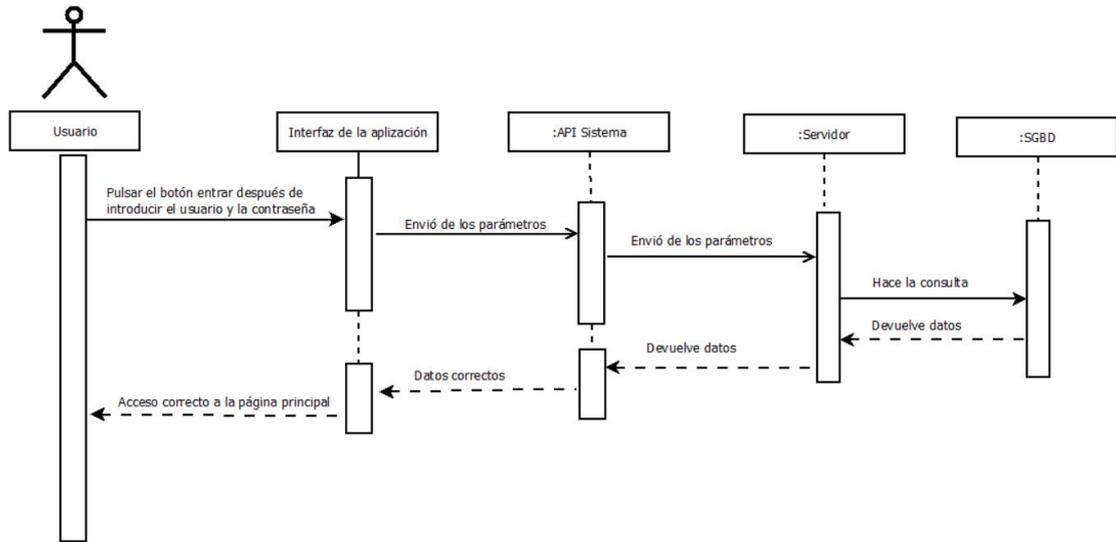


Figura 31. Diagrama de secuencia. Inicio de sesión

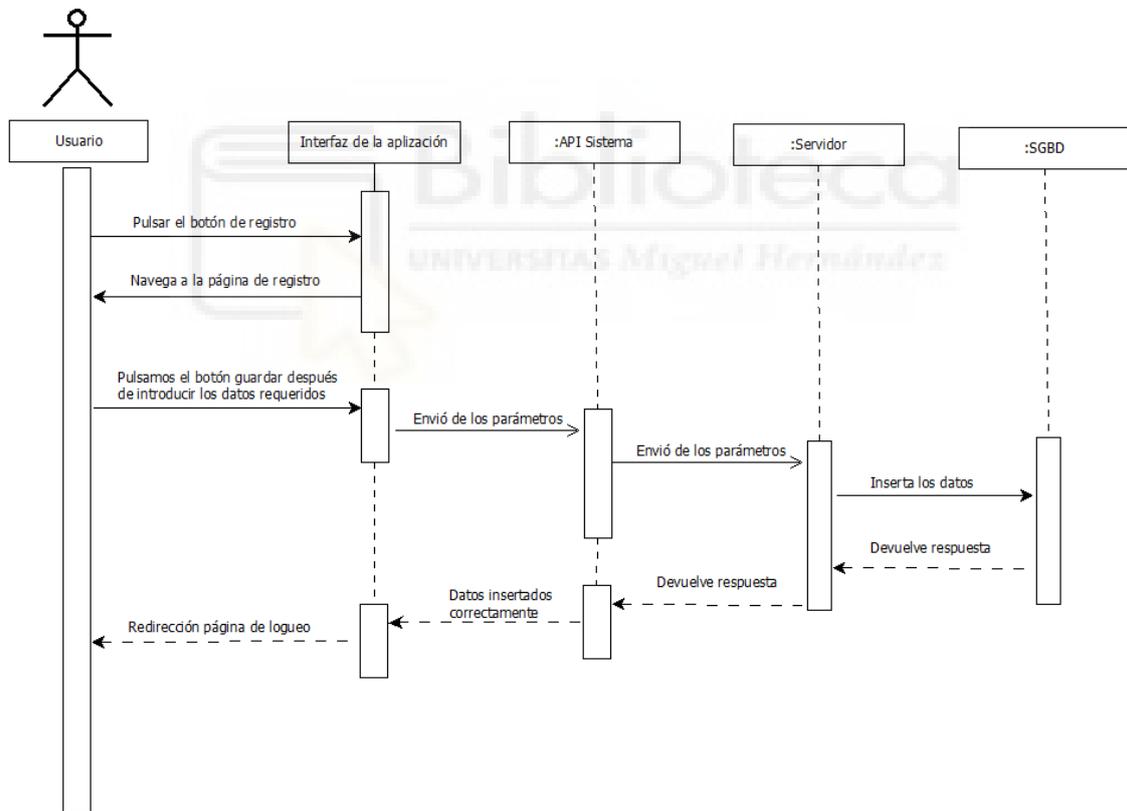


Figura 32. Diagrama de secuencia. Registro

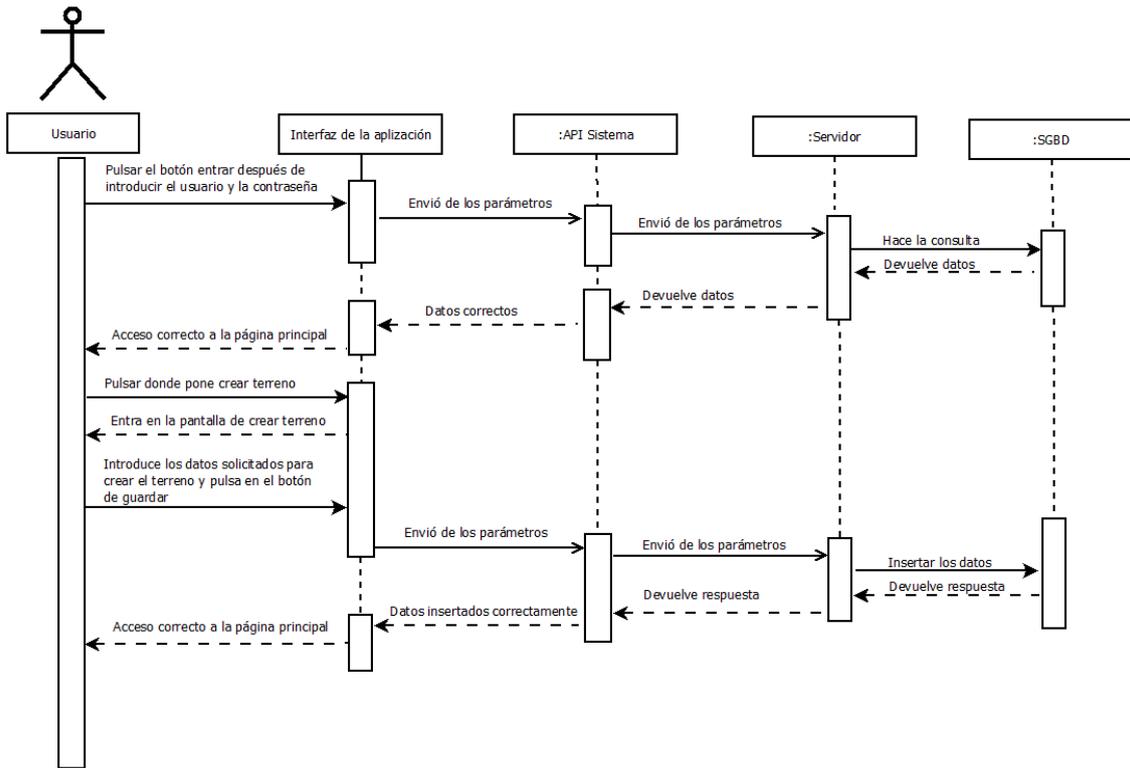


Figura 33. Diagrama de secuencia. Crear terreno

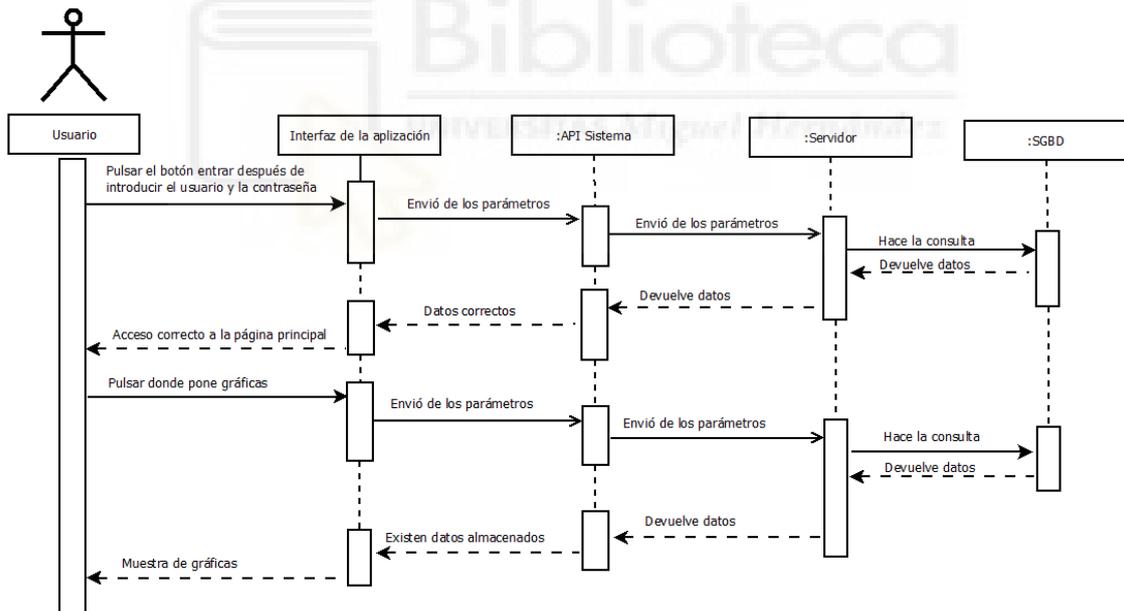


Figura 34. Diagrama de secuencia. Mostrar gráficas

4.2.3. DIAGRAMAS DE ACTIVIDAD

Los diagramas de actividad son parecidos a los diagramas de flujo, son usados para modelar el comportamiento del sistema. Nosotros los vamos a usar como un añadido a una descripción textual del caso de uso.

Para este proyecto se ha necesitado el uso de ellos, ya que son una herramienta gráfica muy clara y ayuda a entender mejor el proceso.

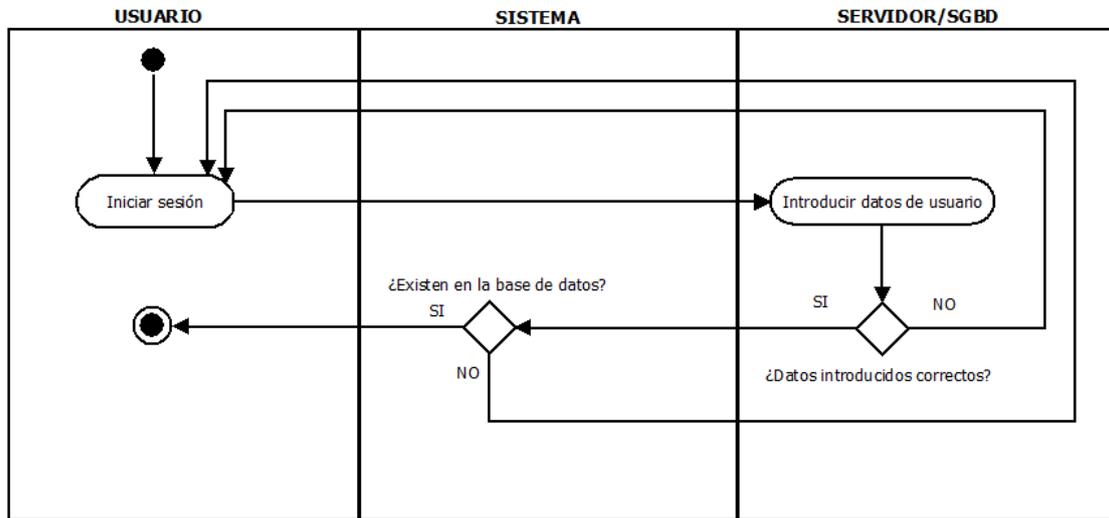


Figura 35. Diagrama de actividad. Iniciar sesión

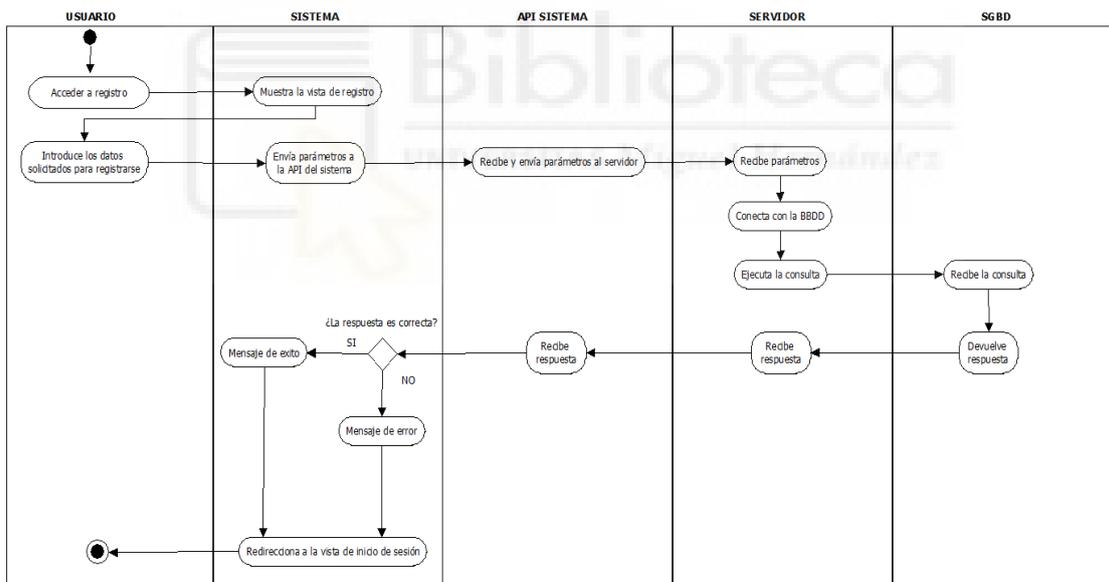


Figura 36. Diagrama de actividad. Registro de usuario

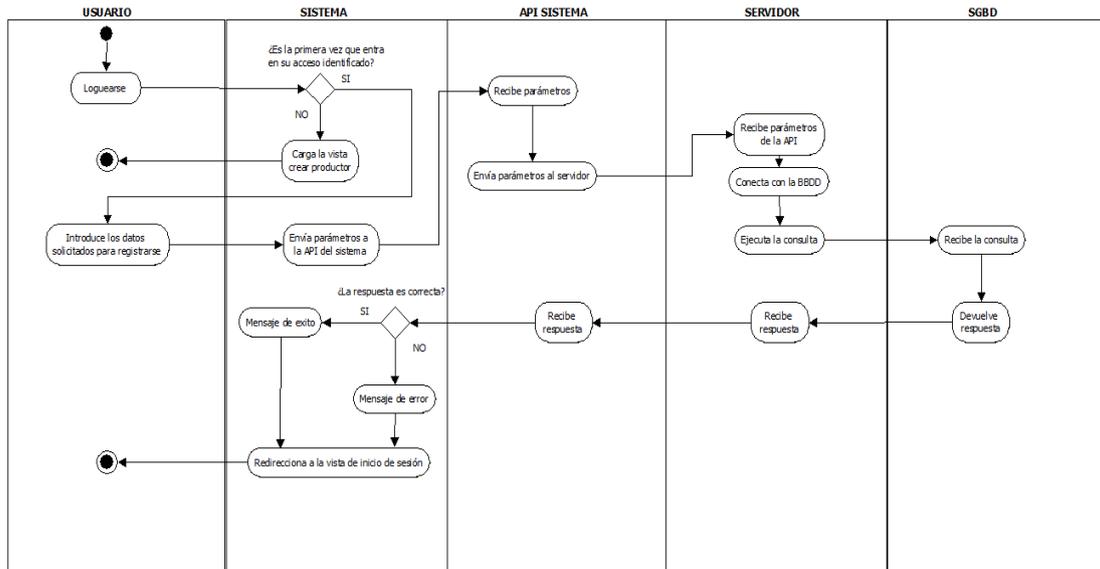


Figura 37. Diagrama de actividad. Crear productor

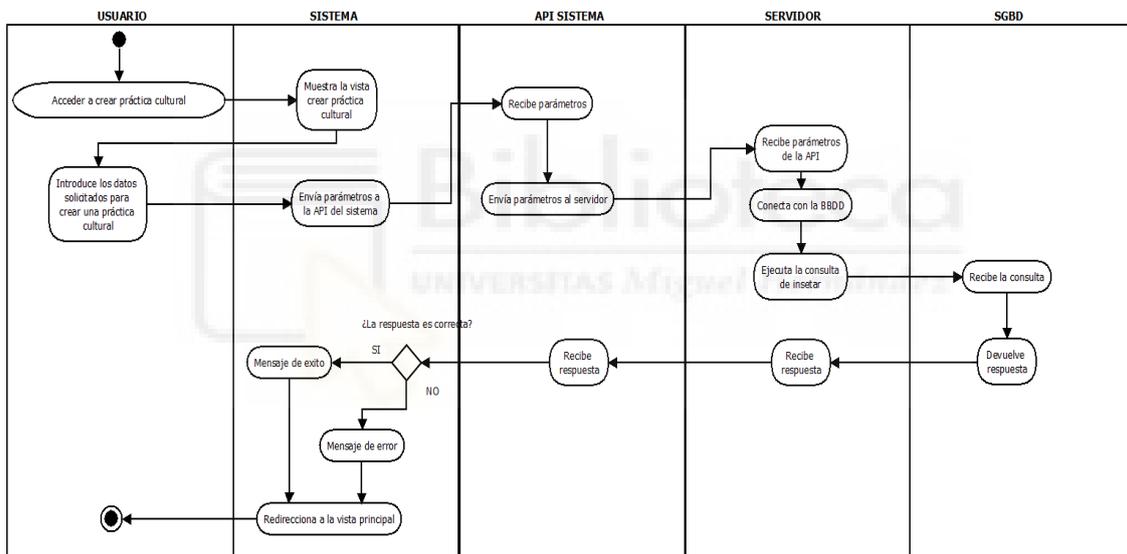


Figura 38. Diagrama de actividad. Crear práctica cultural

4.3. PROTOTIPADO

A la hora de empezar con el desarrollo de este proyecto, no solo se hace hincapié en lo que se refiere a la Ingeniería del Software [7], hemos de tener en una idea de cómo va a ser esa interfaz, para ello es de notoria importancia la realización de un proceso de prototipado efectivo [6].

En este proyecto se han seguido las diferentes fases para un proceso de prototipado efectivo que se han estudiado en la asignatura de Diseño de Sistemas

Interactivos del Grado en Ingeniería Informática en Tecnologías de la Información.

De una forma resumida, presentamos las fases y recursos conseguidos en dicho proceso:

- Fase1: Planificación.
 - Verificar requerimientos: Documentos que datan las reuniones de validación o invalidación de suposiciones y requisitos mostrados en una hoja de trabajo, tras las reuniones que se realizan entre el equipo de desarrollo y con el cliente.
 - Diseñar los flujos de tareas: Documento de flujo de tareas.
 - Definir el contenido y la fidelidad: Documento en donde se recopila la serie de parámetros que determina el grado de fidelidad que debe tener el prototipo y el contenido de la información que se mostrará.
- Fase 2: Especificaciones.
 - Determinar las características
 - Elegir los métodos: Analizar los métodos de prototipado que pueden encajar con nuestra aplicación a desarrollar.
 - Escoger las herramientas: Analizar las herramientas de diseño en las que el equipo se encuentra con mayor experiencia a la hora de trabajar.
- Fase 3: Diseño
 - Seleccionar las reglas del diseño
 - Crear el diseño
- Fase 4: Resultados
 - Revisar el diseño
 - Validar el diseño
 - Implementar el diseño

Se obtiene un esbozo a mano del prototipo, como se muestran en las siguientes capturas:

Crear Propietario

DNI/NIF

Nombre y apellidos / Nombre de Empresa

Fecha de la primera inscripción en CAERM.

Identificador del responsable técnico

Cooperativa o S.A.T. de la que forma parte

ACEPTAR

Figura 39. Prototipado de la vista crear propietario

Juan Francisco

DNI 334448882

Nombre Juan Francisco Torres Martinez

Fecha 27/02/2007

Cooperativa Coato S.C.L.

GUARDAR ELIMINAR

Figura 40. Prototipado de la vista editar productor

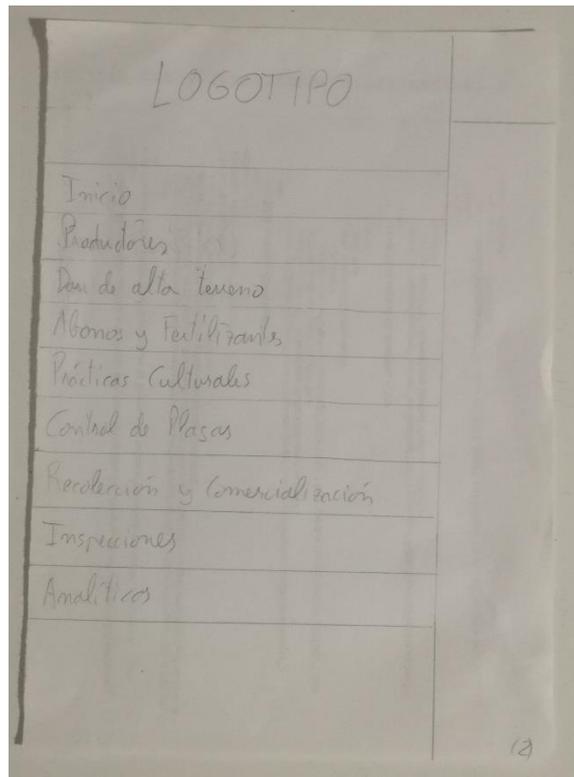


Figura 41. Prototipado de la vista menú lateral

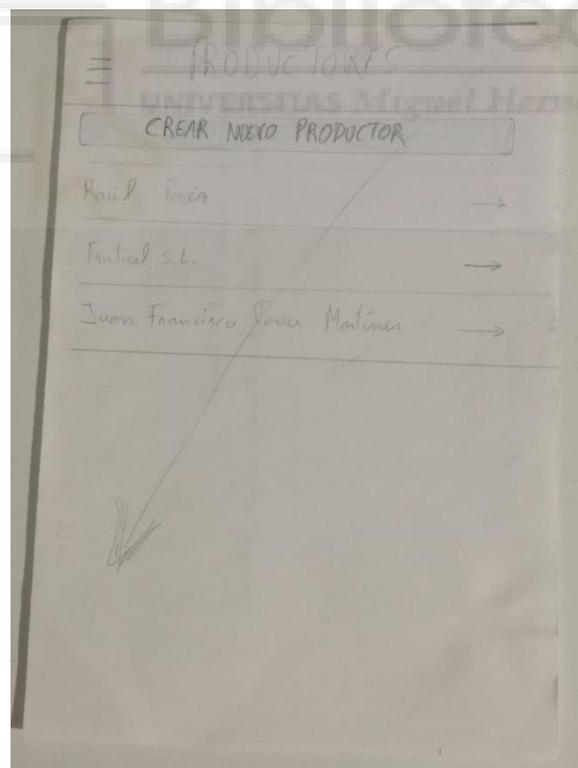


Figura 42. Prototipado de la vista listar productores

4.4. IMPLEMENTACIÓN

En lo referente a este capítulo vamos a explicar con más detalle el funcionamiento de React Native, para que se tengan unas nociones más claras de este framework, seguidamente se detallarán las partes de nuestro código que entrañan mayor complejidad.

4.4.1. COMPONENTES DE REACT NATIVE

Un componente de React Native es similar a las funciones de JavaScript, las cuales reciben entradas que son llamadas "props" y lo que devuelve son elementos cuya función es describir lo que tiene que mostrar en la pantalla.

Los componentes están pensados para dividir la interfaz de usuario en porciones independientes y reutilizables.

4.4.2. CICLO DE VIDA DE UN COMPONENTE

Conocer el ciclo de vida de React Native es sin duda uno de los requisitos de los que debemos saber, porque el ciclo de vida es quien se encarga de gestionar como los componentes serán montados, actualizados y eliminados.

El no conocer el ciclo de vida correctamente nos puede causar problemas cuando estemos creando una aplicación.

Cuando hablamos de problemas, nos referimos a código con bug. Todo esto es debido a que no se estarán manejando los estados del componente y no sabremos cuando está creando, cuando está actualizando y cuando está eliminando.

Un ejemplo sería, cuando creamos un componente que vaya a tener un listener (escuchador de eventos) y este componente lo eliminamos, pero se nos olvida eliminar el listener. Lo que pasaría, es que nos mandaría un mensaje el cual nos diría que tenemos una pérdida de memoria en la aplicación. Con lo cual esto sería un bug en nuestro código y así existen muchos ejemplos que se podría mencionar.

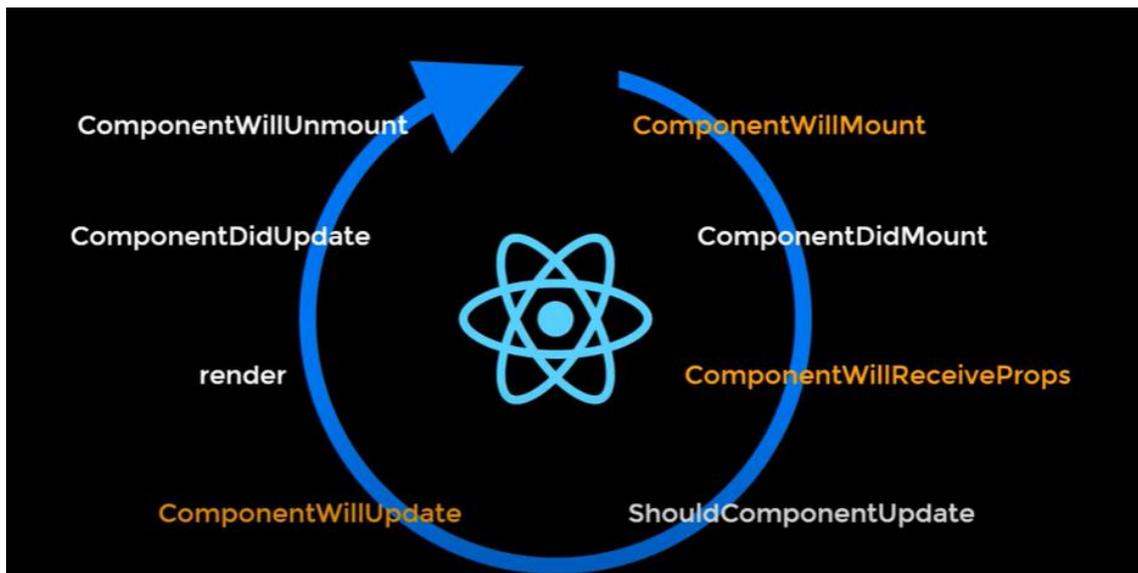


Figura 43. Ciclo de vida con componentes desaconsejados

En la figura 43 nos encontramos con unos componentes de color amarillo, estos componentes están desaconsejados por la comunidad desde la versión 16 y su uso estaría ligado a una mala práctica según la documentación.

Vamos a explicar los componentes que están desaconsejados:

- **ComponentWillMount**: este método se ejecuta antes de que el componente sea montado.
- **ComponentWillRecieveProp**: este método sirve para realizar una acción, una vez que las propiedades hayan sido recibidas por el componente.
- **ComponentWillUpdate**: este método se ejecuta una vez que el componente ha sido actualizado.

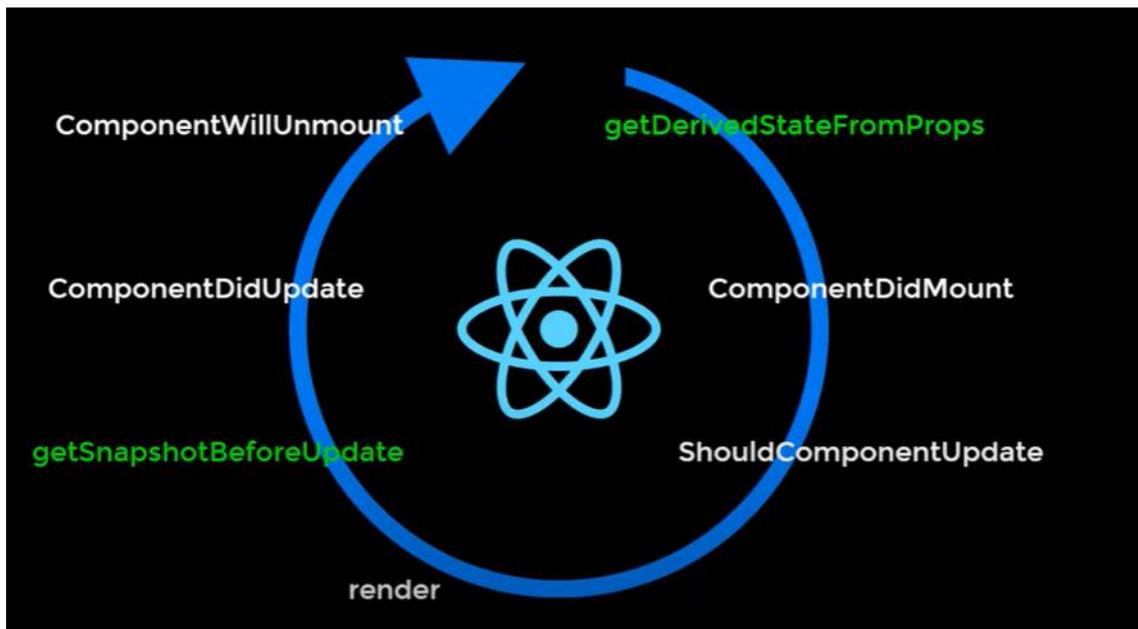


Figura 44. Ciclo de vida con los componentes aconsejados

Ahora explicamos todos los componentes que están aconsejados:

- `getDerivedStateFromProps`: este componente existe solamente con un propósito, permite a un componente actualizar su estado interno según el resultado de cambios en las props. Básicamente este método se ejecutará cuando exista algún cambio en las propiedades.
- `ComponentDidMount`: Este método se ejecuta cuando el componente ha sido montado en el DOM.
- `ShouldComponentUpdate`: Este método nos retorna un tipo de dato lógico o booleano, indicándonos si el componente debe de ser actualizado o no. Este componente recibe el estado y las propiedades, es así como nosotros podremos validar o determinar, si queremos actualizar el componente. (Este método apenas es utilizado, salvo que uno quiera una lógica más personalizada).
- `render`: este método es el más importante, porque es el encargado de mostrar la interfaz de usuario. También podemos explicar que el método `render`, es un método obligatorio, siempre que nosotros utilicemos un componente del tipo clase.

- `getSnapshotBeforeUpdate`: este método se ejecuta justo antes de que el componente sea actualizado. Aquí podemos comparar el estado y las propiedades para saber que cambios se han dado y poder así realizar algún tipo de acción, como por ejemplo alguna animación.
- `ComponentDidUpdate`: este método nos sirve para realizar acciones una vez que el componente ya ha sido actualizado.
- `ComponentWillUnmount`: este método se ejecuta justamente antes de que el componente sea destruido o mejor dicho desmontado. Aquí se podría ejecutar algún tipo de acción especial que se utiliza para eliminar. Por ejemplo, escuchadores de eventos que ya hayamos declarado en el método `ComponentDidMount`. También podríamos limpiar el storage en el caso de estar utilizando Redux, etc.

4.4.3. API FETCH

Antes de que abarcar lo que es el API FETCH y donde se ha utilizado en nuestro proyecto, vamos a explicar que es una API.

Bien, una API por sus siglas del inglés Application Programming Interfaces, que traducido al español se entiende por interfaz de programación de aplicaciones, no es más que una serie procesos y protocolos cuyo objetivo es permitir la comunicación entre dos aplicaciones de software por medio de un conjunto de reglas.

Google maps es un ejemplo de una API, esta nos permite utilizar el servicio de mapas de Google en tu aplicación por medio de JavaScript. Dentro de las APIs, las podemos encontrar publicas a las que todo el mundo tiene acceso y luego las privadas que irán acompañadas de un pago o suscripción según determine el propietario.

Entonces ya podemos explicar que es API Fetch, está API nos permite el acceso y el tratamiento a unas secciones del canal HTTP, bien sean peticiones o respuestas. API Fetch cuenta con un método global llamado `fetch()` que provee de una forma sencilla y lógica la obtención de recursos de forma asíncrona por la red.

```

102     async crearTerreno(
103         dni,
104         municipio,
105         poligono,
106         parcela,
107         recinto,
108         superficieSigpac,
109         superficieCultivada,
110         calificacion,
111         fechaIniPracticass,
112         cultivo,
113         sr,
114         distribucion){
115         return fetch(`${END_POINT}crearTerreno.php`,{
116             method: 'POST',
117             headers: {
118                 'Accept': 'application/json',
119                 'Content-Type' : 'application/json',
120             },
121             body: JSON.stringify({
122                 pDni: dni,
123                 pMunicipio : municipio,
124                 pPoligono : poligono,
125                 pParcela : parcela,
126                 pRecinto : recinto,
127                 pSuperficieSigpac : superficieSigpac,
128                 pSuperficieCultivada : superficieCultivada,
129                 pCalificacion : calificacion,
130                 pFechaIniPracticass : fechaIniPracticass,
131                 pCultivo : cultivo,
132                 pSr : sr,
133                 pDistribucion : distribucion
134             })
135         }).then((res) => res.json())
136         .catch(error => console.error('Error capturado:', error))
137         .then((response) => response.status);
138     } //Fin crearTerreno

```

Figura 45. API Fetch crearTerreno

En la figura 45 se puede ver al función asíncrona crearTerreno que utiliza el método fetch(), donde su primer argumento es la ruta, que está compuesta de una cadena de texto (template string) la cual son literales de texto que habilitan el uso de expresiones incrustadas, lo que quiere decir, es que utilizando las comillas invertidas, pueden coexistir cadenas de texto con marcadores, siempre identificados por el signo dólar seguido de corchetes `${expresión}` y así de esta forma no hace falta la combinación de comillas dobles o simples para las cadenas de texto y el signo más para concatenar expresiones, que es otra forma de hacer lo mismo, pero con una claridad menor y un coste de desarrollo mayor.

```
const END_POINT = `http://192.168.1.42:19000/EcoCuadernoCampo/`
```

Figura 46. Constante que indica la url del servidor

Al final, lo que se consigue utilizando las plantillas de cadena de texto (template strings) es una sintaxis con un código más sencillo a la hora de interpretarlo y un desarrollo más ágil.

Como en esta función lo que se está llevando a cabo es una consulta a una base de datos que está alojada en un servidor, dentro del método `fetch` se le indica, como se van a enviar los datos si por `POST` o por `GET` y dependiendo de esto el método tendrá una estructura u otra. Como se envía por el método `POST`, debemos incluir las cabeceras y después utilizamos el método `JSON.stringify` para convertir el objeto donde van los parámetros que queremos enviar a texto `JSON`.

Seguidamente utilizamos el método `then()` que nos devuelve una promesa o un objeto que indica la finalización correcta o el error en esa finalización eventual de una operación asíncrona. En el caso de un error en esa finalización, se va a manejar ese error con un `catch`, donde enviamos un `console.error()` y con el segundo `then()` controlamos el resultado si este ha tenido una finalización correcta.

En lo que a la parte del servidor se refiere, tenemos la figura 47 que corresponde con el código `crearTerreno` en la parte del servidor.

En la línea cinco muestra la llamada a la función donde se conecta la base de datos, a continuación en la línea nueve y once, primero recibimos los datos del método `fetch` por medio de la función `file_get_contents()` de `php` y se los asignamos a la variable `$json` y después esa variable es decodificada por la función `json_decode()` de `php`. El siguiente paso es asignar las variables recibidas del objeto `JSON` a una variable `php`. Una vez conseguido esto en la línea 26 creamos la sentencia de insertar datos en la base de datos y se la asignamos a una variable, acto seguido se procesa esa inserción de datos.

Una vez acabados los procesos anteriores, se evalúa la variable `$res`, la cual tendrá almacenado un valor de verdadero si la consulta de insertar datos se ha llevado a cabo correctamente o falso si ha habido algún error. Para el caso en el

que `$res` tenga almacenado verdadero, se le asigna un el valor de uno a la posición llamada `status` del array `$response` o un cero en cualquier otro caso.

Por último devolvemos el valor de la variable `$response` por medio de la función `json_encode()`.



```

1  <?php
2
3  require("connect_bd.php");
4
5  $result = connect();
6  $response = array();
7
8  //Recibimos los datos de fetch
9  $json = file_get_contents('php://input');
10 //Estamos decodificando y guardandolos en una variable php
11 $jsonObj = json_decode($json, true);
12
13 $dni = $jsonObj["pDni"];
14 $municipio = $jsonObj["pMunicipio"];
15 $poligono = $jsonObj["pPoligono"];
16 $parcela = $jsonObj["pParcela"];
17 $recinto = $jsonObj["pRecinto"];
18 $superficieSigpac = $jsonObj["pSuperficieSigpac"];
19 $superficieCultivada = $jsonObj["pSuperficieCultivada"];
20 $calificacion = $jsonObj["pCalificacion"];
21 $fechaIniPracticas = $jsonObj["pFechaIniPracticas"];
22 $cultivo = $jsonObj["pCultivo"];
23 $sr = $jsonObj["pSr"];
24 $distribucion = $jsonObj["pDistribucion"];
25
26 $sql = "INSERT INTO terreno(municipio,poligono,parcela,recinto,superficie_SIGPAC,superficie_cultivada,calificacion,fecha_inicio_practicas,cultivo_variedad,S_o_R,distribucion_cultivo,nif_o_dni)
VALUES (:municipio,:poligono,:parcela,:recinto,:superficieSigpac,:superficieCultivada,:calificacion,:fechaIniPracticas,:cultivo,:sr,:distribucion,:dni)";
27 $query = $result->prepare($sql);
28 $res = $query->execute([
29     "municipio" => $municipio,
30     "poligono" => $poligono,
31     "parcela" => $parcela,
32     "recinto" => $recinto,
33     "superficieSigpac" => $superficieSigpac,
34     "superficieCultivada" => $superficieCultivada,
35     "calificacion" => $calificacion,
36     "fechaIniPracticas" => $fechaIniPracticas,
37     "cultivo" => $cultivo,
38     "sr" => $sr,
39     "distribucion" => $distribucion,
40     "dni" => $dni
41 ]);
42
43 if($res){
44     $response["status"] = 1;
45 }else{
46     $response["status"] = 0;
47 }
48 //Devolvemos el valor
49 echo json_encode($response);
50
51

```

Figura 47. Código php crearTerreno

De nuevo en la parte de React Native, nos vamos a la función que inicializó todo el proceso. Esta se encuentra dentro del componente de clase Crear_terreno. La figura 48 muestra el trozo de código referente a esta función. En la línea 62 evaluamos que valor tiene response.status que ha sido previamente asignada a la variable data, en caso de que el valor sea uno (caso de éxito), se mostrará un mensaje en la interfaz del móvil con el texto de "Registro exitoso" y se redireccionara hacia la pantalla de inicio. En caso contrario mostrará un mensaje dentro de texto en el que pondrá "Error en el registro" y se pondrán los campos de entrada de datos en blanco.

```
45     register = async () =>{
46         //console.log(this.state.id)
47         let data = await api.crearTerreno(
48             this.state.dni,
49             this.state.municipio,
50             this.state.poligono,
51             this.state.parcela,
52             this.state.recinto,
53             this.state.superficieSigpac,
54             this.state.superficieCultivada,
55             this.state.calificacion,
56             this.state.fechaIniPracticadas,
57             this.state.cultivo,
58             this.state.sr,
59             this.state.distribucion)
60
61         console.log(data)
62         if(data == 1){
63             Alert.alert("Registro exitoso")
64             //Navegamos al Inicio
65             this.props.navigation.navigate('Inicio')
66         }else{
67             Alert.alert("Error en el registro")
68         }
69     }
```

Figura 48. Función registrar componente de clase crear_terreno.js

4.4.4. REACT NAVIGATION

En toda aplicación móvil se necesita la navegación entre pantallas y en React Native no iba a ser menos. La API encargada de hacer posible esa navegación es React Navigation.

En nuestro proyecto, se ha usado para hacer dicha navegación entre pantallas posible. Para poder utilizar esta API, es necesario la instalación de sus paquetes. Para ello se puede utilizar el gestor de paquetes npm o Yarn.

```
npm install @react-navigation/native
```

Figura 49. Comandos de npm para instalar React Navigation

```
yarn add @react-navigation/native
```

Figura 50. Comandos de Yarn para instalar React Navigation

Para este proyecto es necesario la instalación de dependencias, dado que este proyecto está usando Expo, a continuación, se muestra dicha secuencia de instalación.

Se abre una ventana de símbolo de sistema y se va al directorio donde se encuentre el proyecto, y una vez en dicho directorio se ejecuta la siguiente línea:

```
expo install react-native-gesture-handler react-native-reanimated react-native-screens react-native-safe-area-context @react-native-community/masked-view
```

En este proyecto se han utilizado la combinación de las funciones `createStackNavigator()` y `createDrawerNavigator()`, esta última es necesaria cuando se implementa un menú de cajón.

```
27 const Drawer = createDrawerNavigator();  
28 const Stack = createStackNavigator();
```

Figura 51. Constates creadas para agilizar código

```

112 function Root() {
113   return (
114     <Stack.Navigator>
115       <Stack.Screen name="Login" component={Login}
116         options={{
117           //title: 'CUADERNO DE CAMPO',
118           headerTitle: props => <LogoTitle {...props} />
119         }}/>
120       <Stack.Screen name="Registro" component={Register}
121         options={{
122           //title: 'REGISTRO'
123           //headerTitle: props => <RegistroTitle {...props}/>,
124           headerShown:false
125         }} />
126     </Stack.Navigator>
127   );
128 }
129
130 function App() {
131   return (
132     <NavigationContainer>
133       <Drawer.Navigator initialRouteName="Root" drawerContent={({props})=> <Menu {...props}/>>
134         <Drawer.Screen name="Login" component={Root} options={{
135           //Con swipeEnabled en false desabilitamos el drawer en las páginas de Root
136           swipeEnabled: false,
137         }}/>
138         <Drawer.Screen name="Inicio" component={Inicio} options={{
139           //Con headerShow en false ocultamos el header
140           headerShown: false,
141           // headerTitle: props => <LogoTitle {...props} />
142         }} />
143         <Drawer.Screen name="Productores" component={Productores} />
144         <Drawer.Screen name="CrearProductor" component={Crear_productor}/>
145         <Drawer.Screen name="EditarProductor" component={Editar_productor}/>
146         <Drawer.Screen name="CrearTerreno" component={Crear_terreno}/>
147         <Drawer.Screen name="CrearPracticas" component={Crear_practicas}/>
148         <Drawer.Screen name="CrearAbonado" component={Crear_abonado}/>
149         <Drawer.Screen name="CrearControlPlaga" component={Crear_control_plaga}/>
150         <Drawer.Screen name="CrearRecoleccion" component={Crear_recoleccion}/>
151         <Drawer.Screen name="Graficas" component={Graficas}/>
152         <Drawer.Screen name="Tablas" component={Tablas}/>
153       </Drawer.Navigator>
154     </NavigationContainer>
155   );
156 }
157
158 export default App;

```

Figura 52. Código del núcleo de la navegación

Para que se pueda navegar de una pantalla a otra, los componentes de las distintas pantallas o Screen deben de estar dentro del componente NavigationContainer. Para poder anidar los componentes createStackNavigator() y createDrawerNavigator(), necesitamos crear un componente, en este caso Root() y dentro de este crear la navegación Stack y posteriormente implementarla dentro del componente Drawer.

Si no se hace de esta forma, la navegación no es posible entre estos dos tipos de componentes de navegación.

4.4.5. EXPO ICONS

Una cosa muy interesante, es que al instalar Expo, este lleva un paquete el cual tiene un componente llamado icon. Este componente sirve para utilizar iconos que tiene Native Base. Pero hay otra forma de adquirir iconos y es por medio de los conjuntos de herramientas de iconos como: Font Awesome, Glyphicons o Ionicons. Estos o bien son de código libre o ceden unos iconos gratis.

Para poder utilizarlos en nuestro proyecto, primero se importa el componente y después se añade el componente que hayamos elegido.

```
3 import {FontAwesome} from '@expo/vector-icons';
```

Figura 53. Importación del componente FontAwesome

```
58 <FontAwesome name='user' size={20}></FontAwesome>
```

Figura 54. Componente FontAwesome icono user

4.5. GUÍAS DE ESTILO

Como no se le iba a dar importancia a la interacción persona-móvil en un proyecto móvil. En este proyecto se ha hecho todo lo posible por hacer una interfaz cuya usabilidad sea alta. Por eso se han tenido algunas consideraciones como, por ejemplo:

- Darle al usuario la capacidad de moverse libremente por la aplicación
- Mantener la consistencia y los estándares, no intentar innovar en este ecosistema.
- Mantener la claridad de los elementos, casi siempre menos es más en estos casos.
- No abrumar con los textos.

En cuanto a la identidad corporativa, se han elegido dos colores primarios y dos secundarios:

Como primarios se han elegido el naranja HEX(#FF7101), RGB(255,113,1) y azul celeste HEX(#B3E2FC) RGB(179,226,252).

Como secundarios el negro HEX(#000000), RGB(0,0,0) y blanco HEX(#FBFBFB), RGB(251,251,251).

En cuanto a las guías de estilo utilizadas como referentes, se ha seguido la guía de estilo Material Design [21] mantenida por el equipo de Google y la Human Interface Guidelines [22] de Apple.



CAPÍTULO 5: CONCLUSIONES Y TRABAJOS FUTUROS

5.1. CONCLUSIONES

Con la finalización de este proyecto, los objetivos que nos proponíamos al comienzo de este TFG quedan cubiertos, con lo que se ha logrado una aplicación con un aspecto claro y atractivo más una lógica completa.

Tras el transcurso de este TFG se han adquirido nuevos conocimientos relacionados con el framework React Native, ampliar los conocimientos de JavaScript ES6, poner en práctica la codificación en PHP, trabajar con bases de datos, etc.

Además, nos queda la satisfacción de haber superado lo que habíamos establecido como metas personales, estar relacionado con las nuevas tecnologías que actualmente existen en el mercado y poder trabajar con ellas, ver las nuevas formas de crear aplicaciones móviles y dentro de estas, más concretamente las híbridas, consiguiendo de esta forma, ampliar el abanico de conocimientos adquiridos tras terminar el Grado de Ingeniería Informática en Tecnologías de la Información y obtener de esta forma la oportunidad de cubrir una gama más amplia de puestos de trabajo.

No podemos dejar pasar la experiencia que hemos tenido a la hora de desarrollar en Expo, ya que a la hora de desarrollar solo hay que preocuparse por el código de JavaScript, ni por el de Android, ni por el de iOS. Aunque dependiendo de las necesidades del proyecto si necesitásemos alguna utilidad que solo esté en código nativo, se podrá pasar de Expo.io a React Native CLI. Teniendo en cuenta que una vez que se pase de Expo.io a React Native CLI no habrá vuelta atrás.

5.2. TRABAJOS FUTUROS

Según pasaban los días y se iba avanzando en el proyecto, no paran de surgir posibles nuevas características y mejoras que poder implementar, proceso normal, ya que dentro de los proyectos de desarrollo, tanto móviles como web, estos suelen ser proyectos vivos, los cuales no paran nunca de crecer.

5.2.1. APLICACIÓN WEB

La aplicación web sería la segunda fase del proyecto para aportarle más valor y terminase con su cometido. La funcionalidad de la aplicación web sería, todo lo que el usuario puede hacer en la aplicación móvil y la descarga de documentos con extensiones específicas como .csv, .pdf.

5.2.2. ESTUDIO DE USABILIDAD CON USUARIOS

Con el estudio de usabilidad con usuarios se pretende detectar posibles problemas de usabilidad dentro de las aplicaciones. Estos estudios se componen de unos cuestionarios tipo test, los cuales unos se realizan antes de la prueba física con el usuario, luego están los cuestionarios que van después de cada tarea y por último el cuestionario que se hace al terminar la tarea.

Cada vez más se va apostando por estos tipos de estudios de usabilidad, dado que es la mejor forma de saber si la aplicación construida es usable por el usuario.

5.2.3. SEGURIDAD

En cuanto a la parte de seguridad, hay que hacer un estudio de seguridad para ver qué tipo de sistemas y metodologías se adecuan a nuestro proyecto e implantarlas.

5.2.4. API MAPS

Otra de las características que se podía implementar es la utilización de la API que provee el ministerio de agricultura pesca y alimentación la cual permite identificar geográficamente las parcelas declaradas por los agricultores y ganaderos llamada SIGPAC (Sistema de Información Geográfica de Parcelas Agrícolas). Esto se utilizaría a la hora de dar los terrenos de alta, por medio del mapa los seleccionaríamos y con este proceso ya quedarían registrados los datos referentes a las parcelas.

5.2.5. SISTEMA SCADA

Otra implementación ambiciosa sería la implantación de un sistema SCADA (Supervisión, Control y Adquisición de Datos). Un ejemplo de donde se podría implantar es en el sistema de riego y abonado, para poder regar y abonar sin

tener que estar in situ en las parcelas, por medio de la aplicación y desde casi cualquier lugar.



CAPÍTULO 6: BIBLIOGRAFÍA

- [1] Agroptima. <https://www.agroptima.com/es/>
- [2] Agricolium. <https://agricolum.com/>
- [3] Agroslab. <https://www.agroslab.com/>
- [4] CultivApp. <https://www.cultivapp.com/v2.8.7/>
- [5] El lenguaje unificado de modelado: guía del usuario. (2006). G. Booch, J. Rubaugh y I. Jacobson. Pearson Addison Wesley
- [6] Effective prototyping for software makers. Jonathan Arnowitz, Michael Arent y Nevin Berger. ProQuest Ebook Central.
- [7] Ingeniería del software. (2005). Ian Sommerville. Pearson Addison Wesley
- [8] UML Gota a gota. (1999). Martin Fowler y Kendall Scott. Pearson Addison Wesley
- [9] Hybrid mobile app development (React Native). (2019). Information & Communication Technology (ICT).
- [10] Desarrollo de aplicaciones móviles con Android. (2015). Jorge Santiago Nolasco Valenzuela.
- [11] Desarrollo de aplicaciones IOS con Swift. (20169). Enrique Blasco Blanquer.
- [12] React Native Chart kit. <https://www.npmjs.com/package/react-native-chart-kit>
- [13] React Native Date TimePicker. <https://github.com/react-native-community/datetimepicker>
- [14] Git. <https://git-scm.com/>
- [15] React Native Table Component. <https://github.com/Gil2015/react-native-table-component>
- [16] Status Bar Color. <https://github.com/GeekyAnts/NativeBase/issues/1619>
- [17] React Native. <https://reactnative.dev/>
- [18] Expo. <https://expo.io/>
- [19] Native base. <https://nativebase.io/>
- [20] XAMPP. <https://www.apachefriends.org/es/index.html>
- [21] Material Design. <https://material.io/design>

[22] Human Interface Guidelines. <https://developer.apple.com/design/human-interface-guidelines/ios/overview/themes/>



ANEXOS

ANEXO I

1. INICIO DE SESIÓN

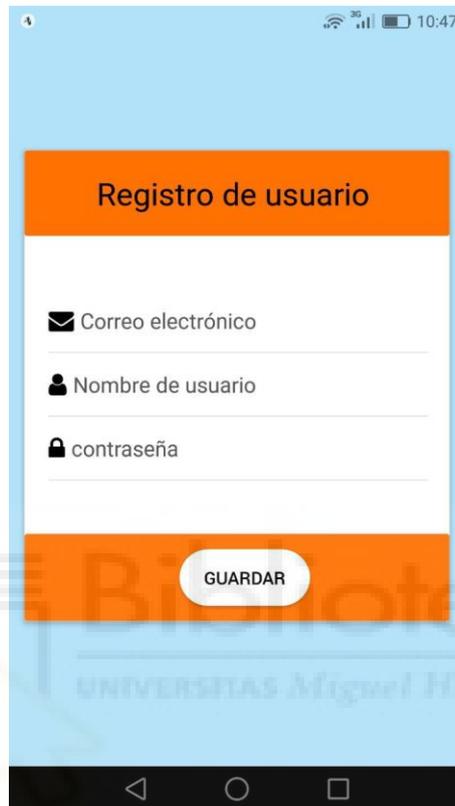
Pantalla de inicio de sesión, esta pantalla es la primera que el usuario visualiza una vez que accede a la aplicación. Esta pantalla es la encargada de verificar los datos de acceso del usuario.



Figura 55. Pantalla inicio de sesión

2. REGISTRO DE USUARIO

Pantalla de registro de usuario. Esta es la pantalla donde el usuario debe de introducir los datos que se le requieren para crear una cuenta en la aplicación.



Registro de usuario

✉ Correo electrónico

👤 Nombre de usuario

🔒 contraseña

GUARDAR

Figura 56. Pantalla registro de usuario

3. INICIO O HOME

Pantalla de inicio o home, en esta pantalla se encuentran todas las características de la aplicación.



Figura 57. Pantalla Inicio o home del administrador



Figura 58. Pantalla de inicio o home

4. MENÚ DE CAJÓN

Captura de pantalla donde se muestra el menú de cajón que se ha implementado para que el usuario en todo caso pueda ir al inicio de la aplicación o salir de ella.

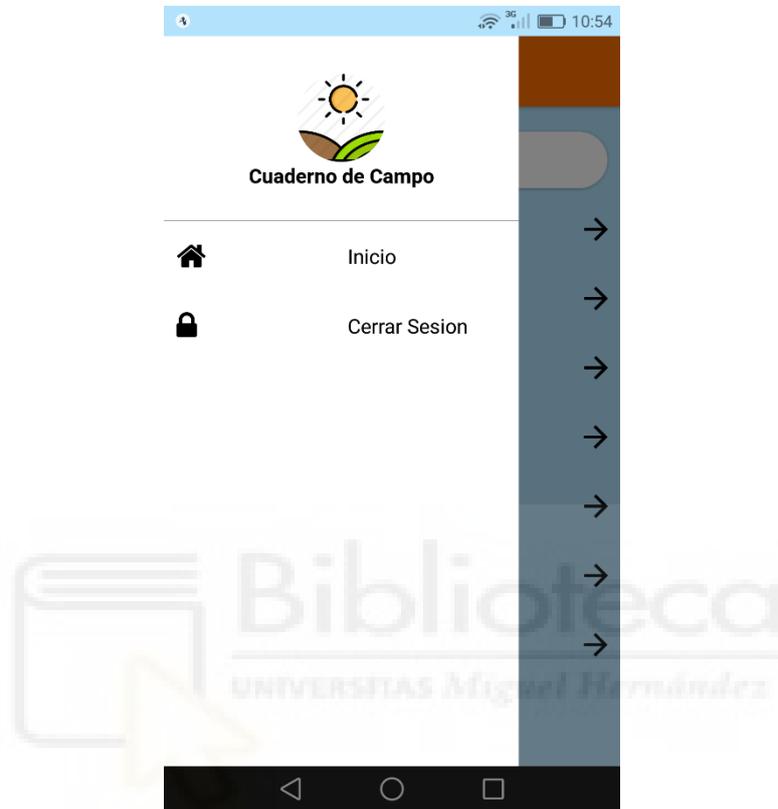


Figura 59. Menú de cajón o lateral

5. CREAR PRODUCTOR

Esta pantalla es donde es dirigido el usuario una vez registrado y se loguea por primera vez, para así, introducir todos los datos necesarios para que se almacenen en la base de datos.

CREAR PRODUCTOR

DNI/NIF

Nombre, apellidos / Nombre de la Empresa

Seleccione la fecha de inscripción en CAERM:
Selecciona la fecha

Identificador del responsable técnico

Cooperativa o S.A.T de la que forma parte

GUARDAR

Figura 60. Pantalla de crear productor

6. LISTAR PRODUCTORES

Esta pantalla es exclusiva del administrador, en ella se muestran todos los usuarios que están dados de alta en la aplicación.

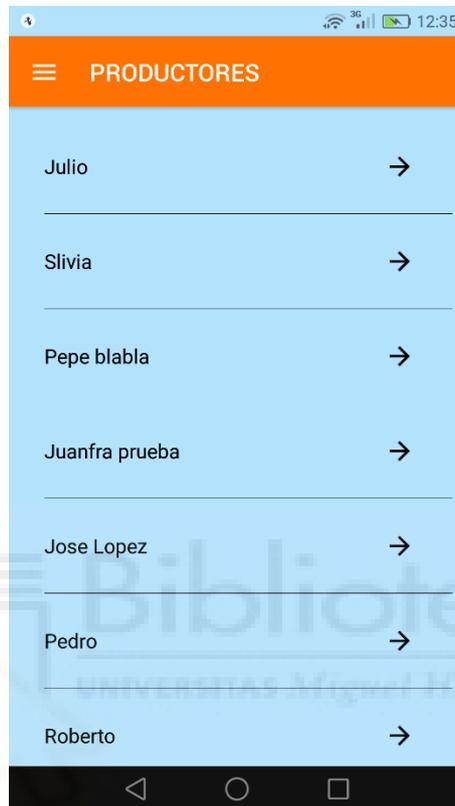


Figura 61. Pantalla listar productores.

7. EDITAR PRODUCTORES

Pantalla donde el administrado puede editar los datos de un usuario que ya esté registrado.

DNI/NIF
00888777H

Nombre, apellidos / Nombre de la Empresa

Seleccione la fecha de inscripción en CAERM:
Selecciona aquí

Identificador del responsable técnico

Cooperativa o S.A.T de la que forma parte

GUARDAR

Figura 62. Pantalla editar productor

8. CREAR TERRENO

Pantalla donde el usuario da de alta los terrenos sobre los que requieren de un seguimiento.



Superficie Cultivada (ha)

Calificación

Fecha de inicio de prácticas
Selecciona la fecha

Cultivo

Secano/Regadío

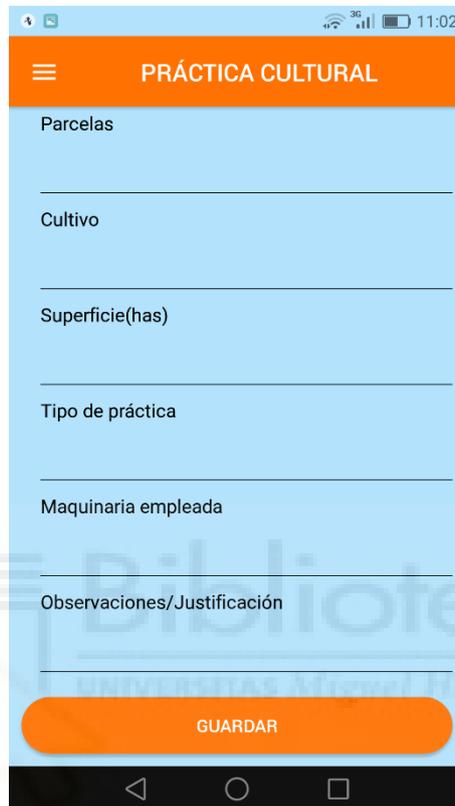
Distribución del cultivo

GUARDAR

Figura 63. Pantalla crear terreno

9. CREAR PRÁCTICAS CULTURALES

Pantalla donde se crean las prácticas culturales que se van realizando en los diferentes terrenos que estén dados de alta.

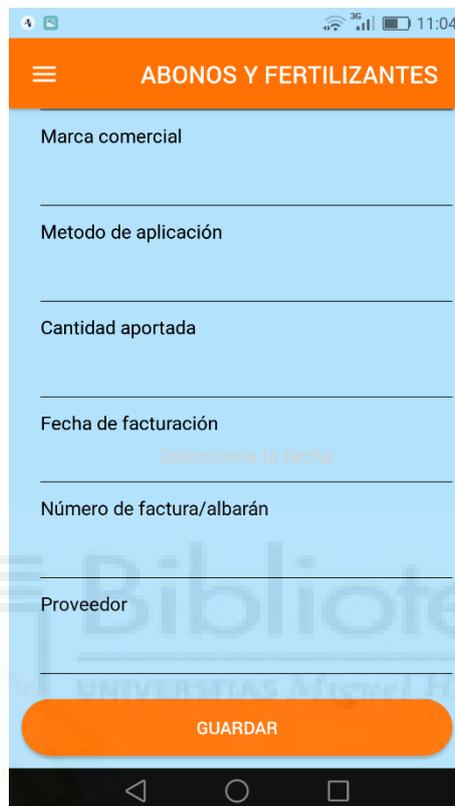


The image shows a mobile application screen for creating a cultural practice. The screen has a light blue background and an orange header bar with the text 'PRÁCTICA CULTURAL' and a menu icon. Below the header, there are several input fields with labels: 'Parcelas', 'Cultivo', 'Superficie(has)', 'Tipo de práctica', 'Maquinaria empleada', and 'Observaciones/Justificación'. Each field is separated by a horizontal line. At the bottom of the form is a large orange button labeled 'GUARDAR'. The top of the screen shows a status bar with a signal strength indicator, a 3G network icon, a battery icon, and the time 11:02. A watermark for 'Biblioteca UNIVERSIDAD Miguel Hernández' is visible in the background.

Figura 64. Pantalla crear práctica cultural

10. ABONO Y FERTILIZANTES

Pantalla donde se crea las acciones de abonado y fertilizado que se van realizando en los diferentes terrenos que estén dados de alta.



ABONOS Y FERTILIZANTES

Marca comercial

Metodo de aplicación

Cantidad aportada

Fecha de facturación
Selecciona la fecha

Número de factura/albarán

Proveedor

GUARDAR

Figura 65. Pantalla crear abonados

11. CONTROL DE PLAGAS

Pantalla donde se crean los controles de plagas que se van realizando en los diferentes terrenos que estén dados de alta.

Fecha de control de plaga
Selecciona la fecha

Parcela/s

Cultivo

Superficie(has)

Composición y riqueza

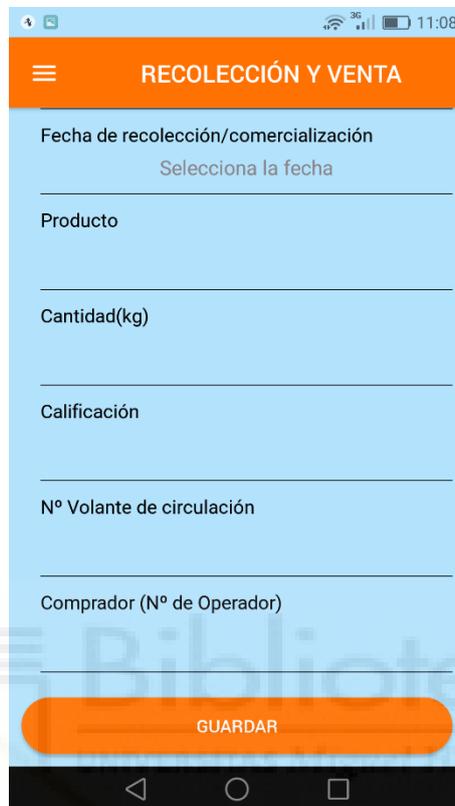
Marca comercial

Metodo de aplicación

Figura 66. Pantalla crear control de plagas

12. RECOLECCIÓN Y VENTA

Pantalla donde se registran las recolecciones y ventas de las diferentes producciones.



RECOLECCIÓN Y VENTA

Fecha de recolección/comercialización
Selecciona la fecha

Producto

Cantidad(kg)

Calificación

Nº Volante de circulación

Comprador (Nº de Operador)

GUARDAR

Figura 67. Pantalla crear recolección y venta

13. GRÁFICAS

Pantalla donde los datos son mostrados gráficamente para que el usuario con un simple vistazo vea la información que lleva introducida.

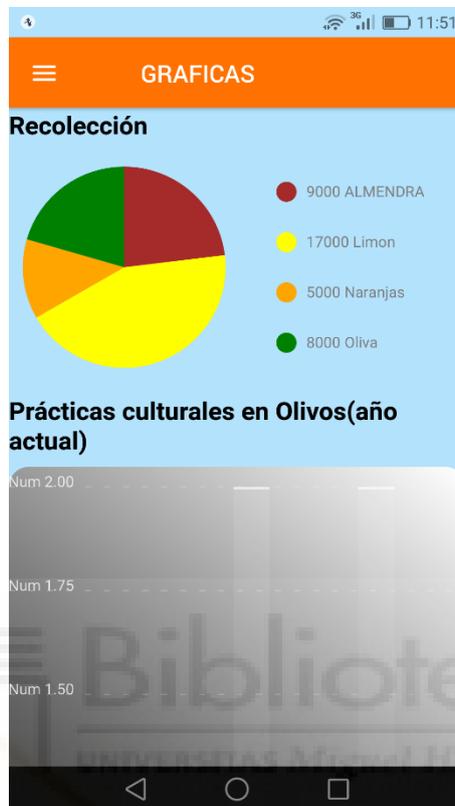
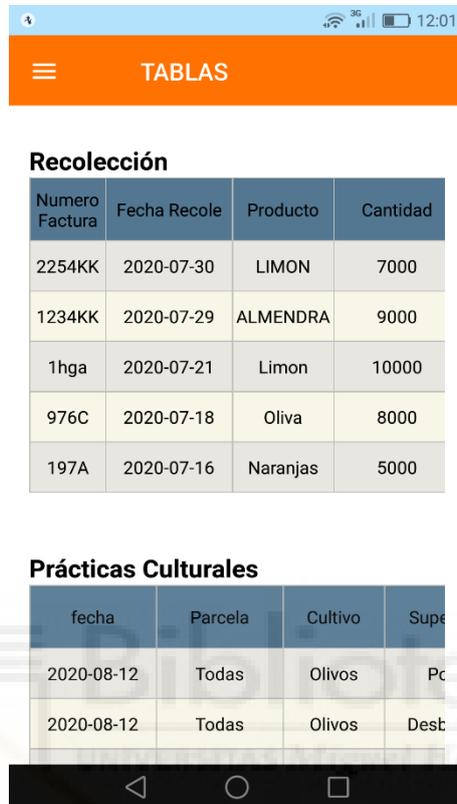


Figura 68. Pantalla donde se muestran las gráficas

14. TABLAS

Pantalla donde se muestran algunas tablas con todos los datos que el usuario lleve introducidos.



The screenshot shows a mobile application interface with an orange header bar containing a menu icon and the word 'TABLAS'. Below the header, there are two tables. The first table, titled 'Recolección', has four columns: 'Numero Factura', 'Fecha Recole', 'Producto', and 'Cantidad'. The second table, titled 'Prácticas Culturales', has four columns: 'fecha', 'Parcela', 'Cultivo', and 'Supe'. A watermark 'Agencia de Extensión y Transferencia de Tecnología' is visible in the background.

Numero Factura	Fecha Recole	Producto	Cantidad
2254KK	2020-07-30	LIMON	7000
1234KK	2020-07-29	ALMENDRA	9000
1hga	2020-07-21	Limon	10000
976C	2020-07-18	Oliva	8000
197A	2020-07-16	Naranjas	5000

fecha	Parcela	Cultivo	Supe
2020-08-12	Todas	Olivos	Pc
2020-08-12	Todas	Olivos	Dest

Figura 69. Pantalla donde muestran las tablas de los datos que tiene el productor