Universidad de Valladolid

Escuela de Ingeniería Informática

Trabajo de fin de grado

Grado en Ingeniería Informática

Mención en Ingeniería de Software

OrtiDis

Herramienta de ayuda de diseño de huertos

Autor:

D. Adrián Vega Acero

Tutora:

Dña. Margarita Gonzalo Tasis

Resumen del proyecto:

El objetivo de este proyecto es el de crear una aplicación gráfica para el diseño de huertos y zonas de cultivo que sea cómoda y simple, de forma que un usuario con o sin experiencia pueda utilizar la aplicación sin dificultades y de una manera ágil, permitiéndole así planificar mediante un dispositivo móvil el diseño de su huerto de forma virtual para poder llevar a cabo el diseño físico posteriormente, descartando así los problemas que pudieran aparecer en el diseño virtual, ahorrando de esta forma costes y sobre todo tiempo al usuario.

Abstract:

The objective of this Project is to create a graphical application for the design of orchards and cultivation areas that is conmfortable and simple, so that a user with or without experience can use the application without difficulties and in an agile way, thus allowing plan the design of your garden virtually using a mobile device to be able to carry out the physical design later, thus ruling out the problems that could appear in the virtual design, thus saving cost and above all time for the user.

Agradecimientos:

Quisiera aprovechar unas pocas líneas de este trabajo para dar las gracias a todas aquellas personas que me han ayudado a lo largo de toda mi etapa universitaria. A mis padres que me han apoyado durante estos años en los malos momentos, a mis amigos de toda la vida Miguel, Rodrigo, Álvaro y mi primo Diego que además ha sido mi compañero de carrera y que siempre han confiado en mí, y a todos los compañeros que he conocido en la universidad y con los que he tenido oportunidad de trabajar.

Índice de Contenidos:

Índice de Figuras:

Índice de Tablas:

Capítulo 1: Introducción y contexto

1.1 Contexto

Nos encontramos ante una sociedad en la que la informática y las nuevas tecnologías avanzan a un ritmo muy acelerado, las personas cada vez estamos más acostumbradas a que esta tecnología influya en nuestro día a día de forma natural y nos aprovechamos cada vez más de la cantidad casi infinita de recursos que estas nos pueden ofrecer para hacer nuestra vida más cómoda y nos permiten realizar muchas tareas mucho más rápido lo que a su vez nos deja más tiempo para poder dedicarnos a nosotros mismos. Además la sociedad está cada día más concienciada con uno de los grandes problemas sociales que se nos presenta a la humanidad como es el cambio climático, provocando así, que muchas personas busquen la forma de poner su granito de arena para solucionar este problema lo que ha llevado a que muchas personas se interesen por producir ellos mismos productos biosostenibles para su consumo propio. Todos estos factores unidos a la creciente demanda de casas con jardín o balcones debido a la pandemia de COVID que sufrimos en todo el mundo desde 2020 ha hecho que muchas personas se aficiones a la jardinería para aprovechar todos estos nuevos espacios de los que disponen.

OrtiDis pretende ayudar a que las personas se acerquen cada vez más a este mundo ayudando mediante una interfaz simple a que los usuarios diseñen y construyan su huerto ideal antes de llevarlo a cabo en el mundo real.

1.2 Motivación

La principal motivación de este proyecto consiste en facilitar a cualquier usuario la elaboración de una simulación virtual para su futuro huerto, permitiéndole planificar el diseño físico final sin coste alguno y pudiendo de esta forma ahorrar tiempo, espacio y dinero a la hora de llevarlo a cabo en el mundo real ayudando también al usuario con una interfaz simple que no requiera grandes conocimientos ni informáticos para su manejo y ofreciendo también al usuario información acerca de las características y necesidades de las plantas y flores que deseen cultivar.

1.3 Objetivos

Construir una aplicación para tablets que permita, a cualquiera de sus usuarios, simular un huerto o zona de cultivo de una forma simple y clara, dando información a los usuarios a cerca de las diferentes características y necesidades de las plantas, permitiéndoles de esta forma que su huerto sea lo más eficiente posible una vez llevado a la vida real.

Principalmente se desea construir una aplicación en una perspectiva 2,5D para que los usuarios puedan visualizar el huerto de una forma más realista, pudiendo mover la cámara en diferentes ángulos para facilitar la vista de todo el huerto. Además se deberá poder visualizar la información de los diferentes tipos de plantas de forma simple y clara para que cualquier persona pueda acceder a ella sin dificultad.

Las plantas una vez situadas en el plano deberán mostrar al usuario también de forma clara y visual, la relación que existe entre plantas cercanas, para que se pueda observar si la distribución del huerto es la correcta. El usuario también deberá poder ser capaz de visualizar la forma de riego más óptima para el huerto simulado.

1.4 Análisis de aplicaciones similares

Se ha realizado un estudio de aplicaciones similares, tanto aplicaciones de sistemas móviles como aplicaciones en línea para extraer sus puntos fuertes y débiles y compararla con la aplicación final que deseamos desarrollar.

1.4.1 Gardena

Figura 1: Captura de pantalla de la aplicación "Gardena".

Es una herramienta online para diseñar nuestro jardín sobre un plano cenital en cuadrícula, no dispone de aplicación móvil actualmente pero es una herramienta muy versátil con la que planificar el diseño de nuestro jardín en proporciones realistas, ya

que podremos medir de forma detallada los diferentes elementos que deseemos colocar, incluyendo tanto diferentes tipos de plantas como arbustos, árboles, o vegetales, como distintos elementos característicos de un jardín, como pueden ser estanques o mobiliario. Permite también agregar al plano infraestructuras, pudiendo reflejar nuestra propia casa o invernaderos y garajes para que el diseño se asemeje lo máximo posible a la realidad. Además dispone de unas pocas plantillas que si bien en poco probable que se asemeje al terreno que deseamos planificar, nos puede servir para conocer los varios de los elementos de los que disponemos y ayudarnos a conocer el manejo de la herramienta.

Uno de sus puntos más fuertes, es el de permitir añadir el diseño de los aspersores que servirán para mantener el jardín, permitiendo observar en el plano la distancia que puede alcanzar cada uno de los aspersores, pudiendo de esta forma conocer el número de ellos y la distribución adecuada para que todas las plantas puedan mantenerse correctamente.

Su principal desventaja sería no disponer de una app móvil como ya se explicó anteriormente además de no poder observar el plano más que con la vista cenital en 2D.

1.4.2 Kitchen garden aid

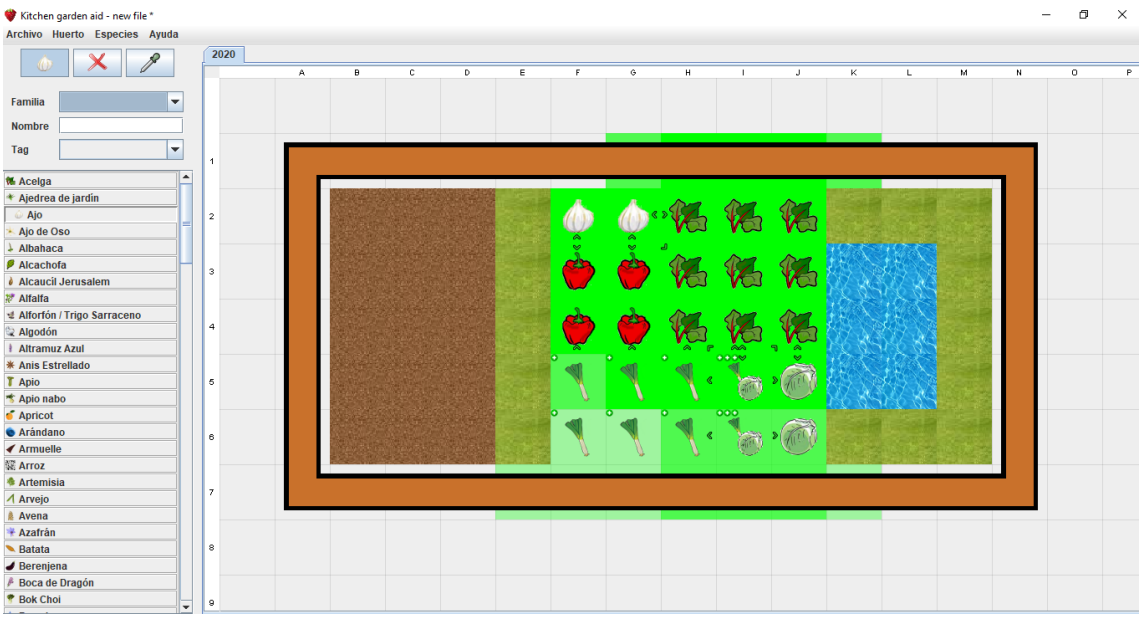

Figura 2: Captura de pantalla de la aplicación "Kitchen garden aid".

Esta herramienta permite diseñar nuestro huerto mediante una cuadrícula en la que podremos escoger el tipo de planta que deseamos cultivar, está disponible en varios idiomas entre los que se incluye el castellano y dispone de una enorme cantidad de

plantas entre las que escoger para nuestro huerto, además muestra información de cada una de ellas, siendo este uno de sus puntos más fuertes, ya que nos permite observar de una forma visual que plantas se pueden cultivar juntas y cuales se perjudican más entre ellas, pudiendo hacernos una idea de las mejores opciones a la hora de escoger y distribuir las plantas de nuestro huerto. También permite guardar y exportar los diseños que creemos y mostrar estadísticas del huerto por años, número de especies, etc. además de mostrar en la información de las plantas enlaces a diferentes artículos sobre las mismas.

Dentro de sus desventajas es importante señalar que aunque dispone de una cuadrícula, los espacios no se encuentran a escala, por lo que puede que a la hora de plantar físicamente nuestro huerto el espacio final diste mucho del que teníamos en mente. Otro de sus puntos más desfavorables es que por el momento sólo se encuentra disponible para pc, y requiere de tener instalado java para funcionar.

1.4.3 Planificador de jardines

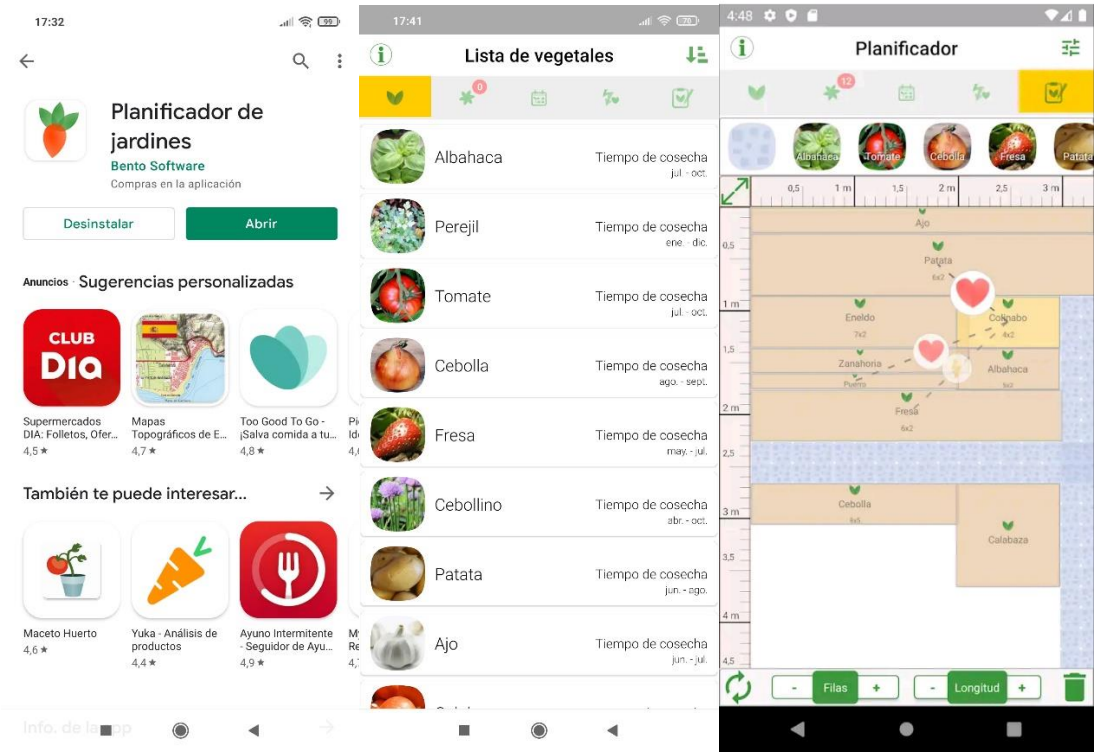

Figura 3: Capturas de pantalla de la aplicación "Planificador de jardines".

Esta aplicación es similar a kitchen garden aid, con la importante diferencia de que esta sí que es una aplicación para dispositivos móviles. Sus funciones son muy similares al de la aplicación anteriormente estudiada y cuenta con más de 60 especies de plantas y flores para nuestro diseño, además de mostrar información muy útil para cada una de las especies como el tiempo de cosecha y períodos de siembra, el espacio que necesita un conjunto de plantas, la temperatura ideal a la que se debe encontrar la planta, el tipo de raíz que tiene, y al igual que kitchen garden aid, muestra la compatibilidad de las diferentes especies de plantas para que podamos escoger las que más se beneficien.

El método de diseño de esta aplicación cuenta con un plano medido a escala en el que podremos ir colocando las diferentes plantas en un espacio rectangular, donde se mostrará la cantidad de plantas que se deben poner, entiendo que como máximo, en la parcela que señalemos, mostrando que para el mismo espacio, como es normal, dependiendo del tipo de planta se podrán plantar más o menos ejemplares. También hay que destacar que la aplicación está en castellano, y también dispone de un calendario en el que se muestran los diferentes tipos de plantas que hemos añadido en el plano con sus fechas de siembra y cosecha para poder acceder a estas fácilmente en lugar de comprobar las fechas de cada especie por separado.

Si tengo que destacar alguna desventaja sería el apartado estético ya que es bastante pobre, sobre todo la ventana del plano, que no muestra ni siquiera un dibujo de la planta que se ha colocado, limitándose a marcar la zona en cuestión con un pequeño icono de una planta genérica.

1.4.4 Planter – Graden Planner

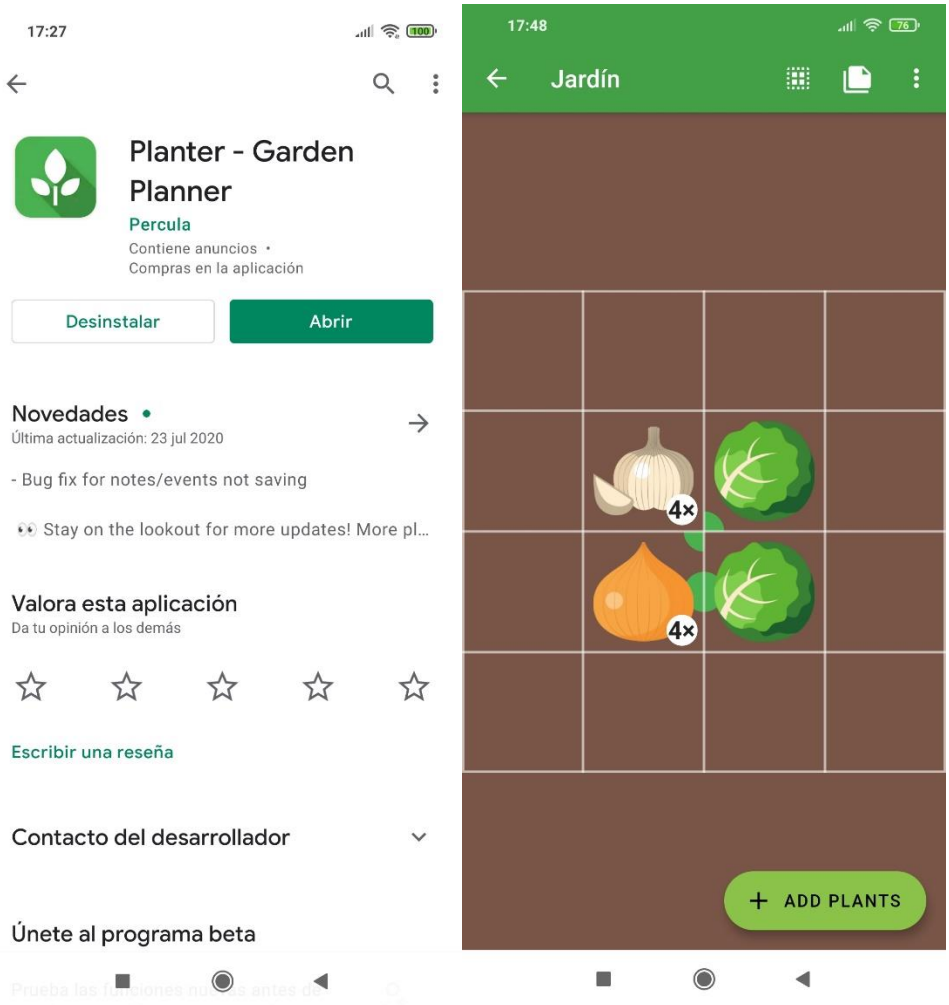

Figura 4: Capturas de pantalla de la aplicación "Planter – Garden Planner".

Aplicación de dispositivos móviles para el diseño de huertos bastante similar a Planificador de jardines, dispondremos nuevamente de un plano cuadriculado esta vez medido en pies ya que esta aplicación a diferencia de la anterior está completamente en inglés, lo cual la hace algo más difícil de manejar especialmente por el cambio en el sistema métrico ya comentado.

Esta aplicación también muestra gran variedad de datos de las plantas que escojamos para nuestro huerto como compatibilidad entre plantas, ambiente que necesita para crecer, cómo plantar las diferentes especies, etc. y aunque dispone de

menos tipos de plantas que la aplicación anterior, ofrece un servicio que la diferencia de las anteriores aplicaciones analizadas, y es que esta aplicación muestra un enlace en cada planta seleccionada, que nos redirecciona a Amazon para la compra de las semillas de la planta seleccionada.

1.5 Estudio de Trabajos de Fin de Grado similares

Se ha llevado a cabo una lectura de diferentes trabajos de fin de grado de ingeniería informática para tener una referencia a la hora de seguir una serie de pasos en la realización de este proyecto. Los trabajos estudiados son, "Route66app"[1], "Desarrollo de aplicación para la generación de juegos geográficos a medida"[2], "GuiaMeUVa"[3], centrándose en aquellos trabajos que trabajaban con aplicaciones similares o que utilizaban herramientas parecidas a las que se utilizarán en el trascurso de este trabajo, para tener una mejor idea no sólo a la hora de implementar la aplicación software final, sino también a la hora de diseñar, estructurar y escribir la memoria de este trabajo de fin de grado. Se han observado las diferentes metodologías de desarrollo de proyectos software y se ha adaptado este proyecto en consecuencia basándose en estos otros trabajos.

Destacar además el TFG de Carlos López Garcinuño "Estudio aplicación de un juego en red multijugador para dispositivos móviles"[4], que ha servido para aprender mejor el funcionamiento de la herramienta Unity[11][12] con la cual se llevará a cabo la implementación de la funcionalidad de este proyecto, y que comenta de forma simple y detallada las principales características de este programa de diseño y desarrollo principalmente de videojuegos.

1.6 Datos generales para el desarrollo del proyecto

Antes de comenzar es importante destacar uno de los aspectos más característicos que se debe tener en cuenta a la hora de desarrollar este proyecto, y es que no sólo se deberá llevar a cabo un estudio sobre las herramientas que se utilizarán en el desarrollo de la aplicación y la memoria de la misma, sino que se deberán estudiar todas aquellas hortalizas, flores y en general toda aquella planta que se desea que aparezca en la aplicación para conocer sus características, y de esta manera hacer que la aplicación final sea lo más útil posible para los usuarios y se acerque lo máximo posible a la realidad, dando información valiosa de cara a la plantación real del huerto que se haya diseñado.

Por ello de todas aquellas plantas que se seleccionarán y que aparecerán en la aplicación se obtendrán una serie de datos tales como la relación de una planta con otra, la fecha de siembra preferente, la fecha de recogida, el clima ideal para la planta y el agua que necesita para su cultivo. Datos que serán de mucha utilidad para los usuarios que deseen utilizar la aplicación, y que aparecerá dentro de la misma de forma que el usuario pueda consultarlos fácilmente.

Una de las características más relevantes de las diferentes plantas que se han seleccionado es su interacción con otras plantas y los beneficios y perjuicios que pueden tener si son plantadas en espacios cercanos. Por ellos se ha consultado en varias páginas que contienen artículos en los que se habla de estos temas, tales como “Tuinen”[6] o “Los peñotes”[7] de las cuales se ha extraído la información que aparecerá reflejada dentro de la aplicación.

1.7 Estructura de la memoria

Ahora se pasará a hablar de la estructura que seguirá esta memoria en los siguientes capítulos.

Primero se explicará la fase planificación que se ha estudiado para afrontar este proyecto, tratando de explicar la metodología que se utilizará de aquí en adelante, para después mostrar una planificación ideal y un estudio de los riesgos que se creen pueden llegar a presentarse en el transcurso del proyecto, terminando por un estudio de los agentes interesados en el proyecto y realizando una estimación de los costes que supondría en desarrollo del proyecto.

Después de la fase de planificación se mostrará la fase de análisis previo tanto al diseño como a la implementación del prototipo software. En esta fase de análisis se explicarán los diferentes roles que aparecerán en el proyecto y se llevarán a cabo tanto el estudio y clasificación de los requisitos de la aplicación como el análisis de casos de uso y el análisis de clases software iniciales, terminando con una serie de estimaciones de mediante diagramas de la realización de los casos de uso respecto a la interacción de los usuarios con el sistema.

El siguiente capítulo que se mostrará será el capítulo de diseño en el cual se expondrá la arquitectura del sistema, los diferentes patrones arquitectónicos que se utilizarán de cara a la implementación del prototipo entregable. Además también se hablará en este apartado del modelo de datos que se espera utilizar y se mostrará también el diseño de la interfaz de usuario junto con las clases software finales que se requerirán.

Tras comentar la arquitectura y el diseño se pasará a explicar la implementación y algunas de las decisiones tomadas a la hora de programar el código de la aplicación, empezando primero por explicar las herramientas utilizadas y los motivos que llevaron a usarlas, para continuar exponiendo los motivos y forma de la implementación de la base de datos de la aplicación. Después se hablará de los distintos tipos de algoritmos más característicos e importantes utilizados para llevar a cabo la implementación de la funcionalidad en la aplicación.

Los siguientes capítulos corresponderán a las pruebas realizadas con la aplicación corroborando que todo funciona de acuerdo a los deseos del programador y se terminará con una exposición de los objetivos cumplidos y el posible trabajo futuro que se llevará a cabo para mejorar la aplicación final.

Para terminar se mostrará toda la bibliografía consultada, y se añadirá un anexo donde se mostrarán los pasos para la instalación de la aplicación.

Capítulo 2: Planificación del proyecto

2.1 Introducción

En el siguiente capítulo se expondrá la metodología utilizada para el desarrollo del proyecto así como la planificación ideal del mismo y se realizará un pequeño estudio de los diferentes riesgos que pueden aparecer durante el desarrollo del proyecto así como las posibles soluciones para mitigar los efectos de los riesgos o para prevenirlos parcial o totalmente.

2.2 Metodología de desarrollo del proyecto

El proyecto software se llevará a cabo siguiendo el método de desarrollo software denominado proceso unificado[5], cuyas principales características son estar dirigido por los casos de uso, tanto en la especificación como en el mantenimiento, estar centrado en la arquitectura en todo el proceso de desarrollo hasta la obtención del producto final y ser iterativo e incremental, dividiendo el trabajo en una serie de pequeñas tareas a menudo coincidiendo con los diferentes casos de uso, y dividiéndose en función de su importancia y riesgo en el proyecto.

Es importante tener en cuenta que cada tarea en la que se divide el desarrollo de la aplicación final, produce un entregable, por lo que utilizando un enfoque de Proceso unificado se conseguirá tener un mayor control y seguimiento del avance del proyecto, lo que minimizará los riesgos que se puedan ir produciendo y conseguirá un producto de buena calidad al mantener un control por parte del tutor académico de cada entregable intermedio producido.

Teniendo claro el método de desarrollo pasaré a explicar las diferentes fases en las que se divide, de forma general, un proyecto que sigue el proceso unificado, las cuales son:

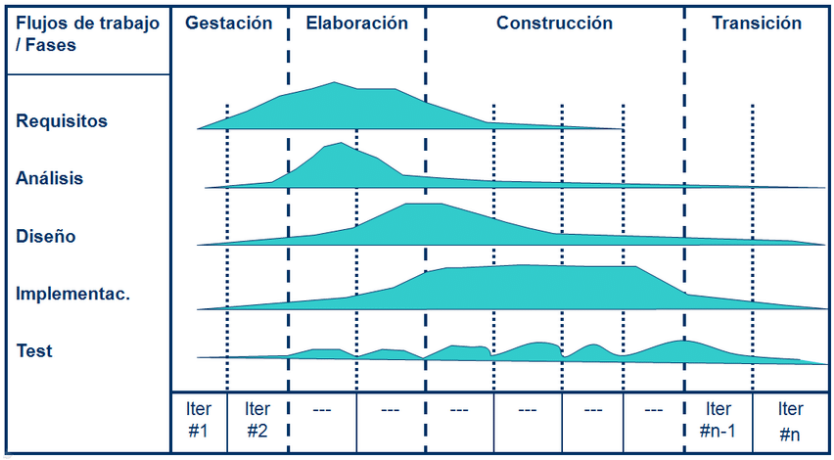

Figura 5: Ciclo y fases del proceso unificado.

- **Fase de inicio o concepción:** en esta fase el trabajo que debemos desarrollar no produce un entregable ejecutable de tipo software, sino que se trata de una fase de planificación de todo nuestro proyecto, incluyendo la planificación de los recursos, entregables, iteraciones y requisitos así como la planificación de los riesgos y de su gestión. Se trata en definitiva de definir los objetivos de nuestro proyecto y su alcance.

- **Fase de elaboración:** la finalidad de esta fase es la de terminar de definir los casos de uso de nuestro sistema y de definir una estructura inicial del mismo. Esta fase termina con un plan concreto de desarrollo que se seguirá a lo largo de la realización del proyecto con el objetivo de minimizar los riesgos.

- **Fase de construcción:** la fase más larga del proyecto, compuesta de una serie de iteraciones que a menudo coinciden con los diferentes casos de uso definidos, lo que provoca partes completas de nuestro sistema que podrán además ser probadas para reducir los riesgos del proyecto.

- **Fase de transición:** la fase final del proyecto, en la cual se obtendrá una versión totalmente funcional pasando por una primera fase beta también funcional.

2.3 Planificación ideal del proyecto

Una vez teniendo claras las fases que sigue el proceso unificado, se procederá a una planificación ideal del proyecto, siendo dicha planificación una estimación de lo que puede ser la duración de las diferentes fases del proyecto, y que puede no corresponderse con el tiempo final utilizado siendo también fechas aproximadas a las que se espera se realicen las diferentes fases.

Se ha tenido en cuenta a la hora de realizar esta estimación un tiempo total para la realización del proyecto completo, tanto la documentación como la aplicación software, de 300 horas, que es el tiempo que se estima dura un trabajo de fin de curso en relación a los créditos que aporta a la hora de obtener el título universitario.

Teniendo en cuenta también que el inicio de este trabajo ha tenido lugar en fecha 15 de febrero de 2021, aproximadamente un semana más tarde del comienzo del segundo cuatrimestre universitario, y calculando una estimación media de trabajo semanal en dicho proyecto de 20 horas, la finalización del proyecto se dará 15 semanas después de su inicio, siendo esta fecha estimada el día 28 de Mayo de 2021.

A continuación se muestra una tabla con los tiempos aproximados de cada una de las fases junto con el número de iteraciones que se darán en cada una de ellas, las horas y las fechas en las que previsiblemente tendrán lugar.

Fase	Iteraciones	Duración estimada	Fecha Inicio	Fecha Fin
Inicio	1	40 horas	15/02/2021	26/02/2021
Elaboración	2	60 horas	01/03/2021	19/03/2021
Construcción	3	180 horas	22/03/2021	21/05/2021
Transición	1	20 horas	24/05/2021	28/05/2021

Tabla 1: Estimación del tiempo necesario para la realización de cada fase del proceso unificado.

Planificación de la fase de Inicio:

Tarea	Duración	Fecha Inicio	Fecha Fin
1.1.1 Estudio del sistema	4 horas	15/02/2021	15/02/2021
1.1.2 Identificación del material necesario	4 horas	16/02/2021	16/02/2021
1.1.3 Descarga y obtención del material	4 horas	17/02/2021	17/02/2021
1.1.4 Realización de la planificación ideal	12 horas	18/02/2021	22/02/2021
1.1.5 Estudio de aplicaciones similares	8 horas	23/02/2021	24/02/2021
1.1.6 Especificación de requisitos iniciales	4 horas	25/02/2021	25/02/2021
1.1.7 Planificación de riesgos	4 horas	26/02/2021	26/02/2021

Tabla 2: Tareas y tiempo estimado para su realización en la fase de inicio del proceso unificado.

Planificación de la fase de Elaboración:

Se prevén dos iteraciones para esta fase, una primera iteración en la que se realizarán la mayoría de diagramas que necesitará nuestro sistema implementar en el futuro así como el diseño de la base de datos y la arquitectura de nuestro sistema, siendo esta iteración de una duración estimada de 40 horas, y una segunda iteración en la que se revisarán dichos diagramas modificándolos incluso añadiendo nuevos casos de uso o eliminando alguno de los iniciales, la cual durará significativamente menos que la anterior con un total de 20 horas.

Fase de elaboración iteración 1:

Tarea	Duración	Fecha Inicio	Fecha Fin
2.1.1 Realización del diagrama de casos de uso	4 horas	01/03/2021	01/03/2021
2.1.2 Realización de los diagramas de secuencia	12 horas	02/03/2021	04/03/2021
2.1.3 Realización del Modelo de dominio del sistema	12 horas	05/03/2021	09/03/2021
2.1.4 Diseño de la base de datos	8 horas	10/03/2021	11/03/2021
2.1.5 Realización del diagrama de despliegue y arquitectura del sistema	4 horas	12/03/2021	12/03/2021

Tabla 3: Tareas y tiempo estimado para su realización en la primera iteración de la fase de elaboración del proceso unificado.

Fase de elaboración iteración 2:

Tarea	Duración	Fecha Inicio	Fecha Fin
2.2.1 Revisión diagrama de casos de uso	4 horas	15/03/2021	15/03/2021
2.2.2 Revisión diagramas de secuencia	4 horas	16/03/2021	16/03/2021
2.2.3 Revisión modelo de dominio	4 horas	17/03/2021	17/03/2021
2.2.4 Revisión diseño de la base de datos	4 horas	18/03/2021	18/03/2021
2.2.5 Revisión diagrama de despliegue y arquitectura del sistema	4 horas	19/03/2021	19/03/2021

Tabla 4: Tareas y tiempo estimado para su realización en la segunda iteración de la fase de elaboración del proceso unificado.

Planificación de la fase de Construcción:

La planificación se ha realizado teniendo en cuenta 3 iteraciones, en las cuales se implementará la aplicación en su totalidad en la primera iteración, se realizarán casos de prueba así como la depuración de los posibles errores que puedan darse en la aplicación en la segunda iteración, y la fase se terminará escribiendo el manual de usuario en la tercera iteración.

Fase de construcción iteración 1:

Tarea	Duración	Fecha Inicio	Fecha Fin
3.1.1 Implementación de la aplicación	120 horas	22/03/2021	30/04/2021

Tabla 5: Tareas y tiempo estimado para su realización en la primera iteración de la fase de construcción del proceso unificado.

Fase de construcción iteración 2:

Tarea	Duración	Fecha Inicio	Fecha Fin
3.2.1 Implementación de casos de prueba	40 horas	03/05/2021	14/05/2021

Tabla 6: Tareas y tiempo estimado para su realización en la segunda iteración de la fase de construcción del proceso unificado.

Fase de construcción iteración 3:

Tarea	Duración	Fecha Inicio	Fecha Fin
3.3.1 Creación del manual de usuario de la aplicación	20 horas	17/05/2021	21/05/2021

Tabla 7: Tareas y tiempo estimado para su realización en la tercera iteración de la fase de construcción del proceso unificado.

Planificación de la fase de Transición:

Tarea	Duración	Fecha Inicio	Fecha Fin
4.1.1 Revisión de la aplicación software	8 horas	24/05/2021	25/05/2021
4.1.2 Revisión de la documentación del proyecto	8 horas	26/05/2021	27/05/2021
4.1.3 Revisión de la memoria del proyecto	4 horas	28/05/2021	28/05/2021

Tabla 8: Tareas y tiempo estimado para su realización en la fase de transición del proceso unificado.

2.4 Análisis y gestión de riegos

Según el PM-BOK[8], un riesgo: "Es un evento o condición incierta que, si ocurre, tiene un efecto positivo o negativo en los objetos del proyecto".

Hay que tener en cuenta que los riesgos son problemas futuros que pueden o no darse, por lo que es importante hacer una planificación de riesgos que pueden aparecer en nuestro proyecto para poder prevenirlos y evitarlos en la medida de lo posible teniendo en cuenta el impacto que puede acarrear en el proyecto, y la probabilidad de que dicho evento suceda.

Las siguientes dos tablas han sido extraídas de la quinta edición del libro Software Project Management de Bob Hughes y Mike Cotterell [9], donde podemos observar una estimación del rango de probabilidades de riesgo y el aumento del impacto en el costo del proyecto.

La probabilidad de que un riesgo suceda sigue la siguiente tabla:

Probabilidad	Rango
Alta	Más de 50% de probabilidades de que el evento suceda.
Significativa	30% - 50% de probabilidades de que el evento suceda.
Moderada	10% - 29% de probabilidades de que el evento suceda.
Baja	Menos de 10% de probabilidades de que el evento suceda

Tabla 9: Probabilidad de aparición de riesgos.

La clasificación del nivel de impacto en el proyecto seguirá la siguiente tabla, teniendo en cuenta que el costo del proyecto viene dado por la planificación inicial y que incluye el tiempo invertido en el proyecto:

Nivel de Impacto	**Rango**
Alta	Más de 30% de aumento de costo
Significativa	20% - 29% de aumento de costo
Moderada	10% - 19% de aumento de costo
Baja	Menos de 10% de aumento de costo

Tabla 10: Nivel de impacto en el costo de un proyecto de un riesgo.

A continuación se han identificado los siguientes riesgos que pueden aparecer en el transcurso de la creación del proyecto. Las tablas muestran los riesgos junto con su probabilidad estimada y efecto sobre el mismo así como las medidas tomadas para paliar dicho riesgo que aparecen en el plan de contingencia de cada riesgo.

Las estrategias que seguiremos para tratar los diferentes riesgos que aparecen en este proyecto son:

- **Aceptación del riesgo:** Cuando el coste de evitar o eliminar el riesgo es superior al coste de que se produzca el riesgo en sí, por lo que se opta por convivir con el riesgo.
- **Eliminación del riesgo:** Eliminar por completo la parte que puede generar los problemas.
- **Reducción del riesgo:** Prevenir la aparición del riesgo.
- **Transferencia del riesgo:** Trasladar a otra persona o empresa el tratamiento del riesgo, como podría ser una empresa de seguros que se haga cargo del coste producido por un incidente.

Identificación	R01 – El proyecto no se ajusta a la planificación inicial.
Descripción	El desarrollo del proyecto no se ajusta a la planificación planteada inicialmente.
Impacto	Significativo.
Probabilidad	Alta.
Consecuencia	El tiempo de desarrollo del proyecto no se ajustará con los tiempos especificados inicialmente.
Estrategia	Aceptación del riesgo.
Plan de contingencia	Revisar y modificar la planificación del proyecto con los nuevos datos de fechas y tiempos que se han obtenido hasta el momento.

Tabla 11: Riesgo número 1 identificado en el proyecto.

Identificación	R02 – Problemas técnicos de los recursos utilizados para el desarrollo de la práctica.
Descripción	Problemas tanto software como hardware en los recursos utilizados dejándolos inservibles total o parcialmente.
Impacto	Alto.
Probabilidad	Baja.
Consecuencia	Se paralizará o ralentizará el avance del proyecto.
Estrategia	Reducción del riesgo.
Plan de contingencia	Reparar o reemplazar los recursos dañados por otros que funcionen correctamente

Tabla 12: Riesgo número 2 identificado en el proyecto.

Identificación	R03 – Problemas Técnicos en recursos ajenos al desarrollador.
Descripción	Problemas en recursos externos como fallos de red, de suministro eléctrico, etc.
Impacto	Alto
Probabilidad	Baja
Consecuencia	Se paralizará o ralentizará el avance del proyecto.
Estrategia	Aceptación del riesgo.
Plan de contingencia	Sustituir los recursos dañados por otros que funcionen correctamente.

Tabla 13: Riesgo número 3 identificado en el proyecto.

Identificación	R04 – Indisposición del desarrollador del proyecto por problemas personales.
Descripción	Problemas que acarreen una indisposición para avanzar en el proyecto por parte del desarrollador.
Impacto	Moderado.
Probabilidad	Moderado.
Consecuencia	Retraso en las entregas del proyecto.
Estrategia	Reducción del riesgo.
Plan de contingencia	Tratar el problema de la forma más rápida posible.

Tabla 14: Riesgo número 4 identificado en el proyecto.

Identificación	R05 – Problemas derivados de recursos software
Descripción	Problemas que tengan relación con las herramientas utilizadas con el desarrollo del proyecto, como puede ser Unity
Impacto	Moderado.
Probabilidad	Alta.
Consecuencia	Retraso en las entregas del proyecto.
Estrategia	Aceptación del riesgo.
Plan de contingencia	Replantear el diseño o la implementación para aplacar el riesgo.

Tabla 15: Riesgo número 5 identificado en el proyecto.

Identificación	R06 – Pérdida del trabajo realizado.
Descripción	Pérdida total o parcial del proyecto.
Impacto	Alto.
Probabilidad	Baja.
Consecuencia	Rehacer todo el trabajo que se haya perdido.
Estrategia	Reducción del riesgo.
Plan de contingencia	Restaurar la última versión que haya sido guardada.

Tabla 16: Riesgo número 6 identificado en el proyecto.

Identificación	R07 – Puesta en marcha del estado de alarma en el país.
Descripción	Puesta en marcha del estado de alarma en el país debido a pandemias, guerras o desastres naturales.
Impacto	Alto.
Probabilidad	Baja.
Consecuencia	Retraso en las entregas parciales del proyecto así como en la entrega final del mismo.
Estrategia	Aceptación del riesgo.
Plan de contingencia	Intentar trabajar con normalidad en la medida de lo posible.

Tabla 17: Riesgo número 7 identificado en el proyecto.

Identificación	R08 – Baja experiencia con Unity
Descripción	Problemas derivados de la baja experiencia con Unity, la herramienta utilizada para llevar a cabo la implementación de la aplicación.
Impacto	Moderado
Probabilidad	Alta
Consecuencia	La velocidad de desarrollo del proyecto se puede ver reducida.
Estrategia	Aceptación del riesgo.
Plan de contingencia	Intentar recuperar el tiempo perdido o retrasar en la medida de lo posible las entregas.

Tabla 18: Riegos número 8 identificado en el proyecto.

2.5 Agentes principales interesados en el producto

Los clientes objetivo de la aplicación son todas aquellas personas interesadas en el mundo de la agricultura y el cultivo de plantas de todo tipo incluyendo no sólo plantas comestibles o que den algún tipo de fruto o producto ecológico, sino también las meramente decorativas como plantas de interior, flores y un largo etcétera.

Además la aplicación busca un manejo simple para que cualquier persona independientemente de sus conocimientos y destrezas en el uso de dispositivos electrónicos pueda ser capaz de aprender y utilizar la aplicación con soltura. De esta forma se busca conseguir abarcar a un grupo de población mucho mayor, considerando los rangos de edad de los posibles usuarios de la aplicación entre 18 y 70 años sin hacer distinción entre sexos.

2.6 Presupuesto total requerido para la realización del proyecto

Para la estimación del presupuesto final que requerirá el desarrollo del proyecto se ha calculado de forma que se supondrá que el proyecto lo realiza un autónomo, por lo que el coste tiene que ser calculado mediante muchos factores que se relatarán a continuación. Para ello nos fijaremos en proyectos reales y en los conocimientos obtenidos de la asignatura de 4º de carrera de ingeniería informática, Planificación y Gestión de proyectos cuyas teorías se basan en el libro “Software Project Management” anteriormente mencionado.

2.6.1 Costes hardware

Los costes hardware para el desarrollo del proyecto son los siguientes, destacar que los días de uso corresponden con los días que se ha estimado llevará la realización total del proyecto:

Hardware	Coste aproximado de compra	Tiempo de uso	Días de uso	Coste de uso total
Portátil Asus	600€	4 horas al día según las estimaciones iniciales.	75 días aproximadamente	0.09€ x 75 = **6.75€**
Móvil Xiami Pocophone	250€	24 horas, de las cuales activo 2 horas al día aproximadamente	75 días aproximadamente	0.16€ aproximadamente

Tabla 19: Costes hardware del proyecto.

Debido a que los dispositivos mostrados anteriormente pueden ser usados para posteriores proyectos y que de hecho ya han sido usado para anteriores proyectos, no se tendrá en cuenta el total del coste de compra sino solamente el coste de mantenimiento y el coste de la factura de la electricidad que consumen, es decir el total queda en 6.91€.

2.6.2 Costes software

Respecto a los costes software y en el caso de este primer prototipo en un primer momento es 0€ debido a que todo el software utilizado para la realización de este proyecto ha sido gratuito, sin embargo se ha tenido la consideración de hacer una estimación de los costes realistas de un proyecto pagado.

Software	Precio	Total en el proyecto
Unity pro	1800 € al año	369.86 € por 75 días
Recursos de la asset store de Unity	100 €	100 €

Tabla 20: Costes software del proyecto.

Pequeña aclaración de que los costes en la asset store de unity son muy variables y la cantidad de recursos que sería necesario obtener por este medio también es variable y difícil de estimar, por lo que teniendo en cuenta que los precios suelen rondar los 20 €, aunque hay mucho mayores, y estimando que se necesite utilizar 5 de estos recursos diferentes el coste estimado de este medio ha sido de 100 € para el total del proyecto.

Por esto el coste de software total para el proyecto podría oscilar entre los 0 € utilizando solamente herramientas y recursos gratuitos, o hasta 469.86 €.

2.6.3 Costes de esfuerzo

Para calcular el coste del esfuerzo lo primero que se ha tenido en cuenta es el sueldo medio en 2021 de un ingeniero informático en España, el cual es de 36500€ brutos al año, unos 1980 € netos por mes según la página Jobted[13], por lo que para el desarrollo del proyecto teniendo en cuenta la estimación de la duración de este de unos 75 días, el coste sería de unos 7500 € brutos.

También deberemos tener en cuenta la cuota de autónomo que actualmente es de 286 € al mes si se cotiza la base mínima, es decir 944,4 €, o de 1.233,2 € si se cotiza la base máxima, o lo que es lo mismo 4.070,1 €. Por lo que teniendo en cuenta esto y realizando una regla de tres la cuota que nos tocaría pagar con un sueldo de ingeniero informático es de aproximadamente 605 €.

2.6.4 Costes indirectos:

Dentro de los costes indirectos hay que tener en cuenta el alquiler del espacio de trabajo, así como la calefacción y todos los gastos externos asociados al contrato con el proveedor de internet y telefonía móvil. Los gastos calculados en costes indirectos por el alquiler del espacio de trabajo han sido obtenidos del servicio de vivero de empresas que ofrece el parque científico de la uva[25] en el cual ya viene incluida la calefacción y el servicio de internet, por lo que el coste indirecto total calculando un espacio de trabajo de 15m2 son:

Motivo	**Coste**
Alquiler del espacio de trabajo	12€/m2 al mes – 180€ al mes en total

Tabla 21: Costes indirectos.

Por lo tanto el coste indirecto previsto para la realización de este proyecto asciende a unos 613,63€ teniendo en cuenta que se estima que el proyecto dure 75 días laborables y estimando que un mes tiene 22 días laborales.

2.7 Seguimiento del proyecto

En este apartado se procederá a explicar los pasos que ha seguido la elaboración real del proyecto teniendo en cuenta la planificación inicial que se estimó con anterioridad en esta memoria.

La primera fase de inicio transcurrió sin dificultades puesto que el tiempo que se estimó para su realización se cumplió fácilmente puesto que aunque el paso 1.1.4 **Realización de la planificación ideal** duró algo más de lo previsto, los pasos anteriores y posteriores se realizaron a mayor velocidad por lo que el tiempo estimado se llegó a cumplir sobradamente.

Llegando a la fase de elaboración los problemas comenzaron una vez llegados al último paso de la primera iteración, es decir, el paso 2.1.5 **Realización del diagrama de**

despliegue y arquitectura del sistema, ya que al ser una aplicación desarrollada con la plataforma Unity, la arquitectura del sistema era diferente a muchas de las estudiadas anteriormente por lo que se tuvo que realizar una investigación a mayores en ese ámbito ya que no se había previsto este problema. Teniendo en cuenta que la fecha de finalización del proyecto distaba mucho de la hora final de entrega y había varias semanas de diferencia, se optó por no realizar grandes modificaciones en el plan inicial, simplemente se retrasaron el resto de tareas posteriores a este problema.

La segunda iteración de la fase de elaboración transcurrió de la forma planificada sin dificultades.

Dentro de la fase de construcción en la primera iteración, el paso 3.1.1 **Implementación de la aplicación** fue donde aparecieron los primeros problemas graves no previstos, ya que el tiempo de implementación para la aplicación estimado en unas 120 horas se amplió enormemente debido a la inexperiencia por parte del desarrollador con la plataforma de desarrollo de video juegos Unity, provocando que el tiempo final empleado en este apartado aumentara enormemente. Por ello se optó por retrasar nuevamente el resto de tareas posteriores debido a que entraba dentro del rango de entrega del proyecto, aunque ante la situación de ver como el tiempo iba a terminar siendo justo se optó por aumentar de las 4 horas diarias los días que fuera posible para intentar paliar en la medida de lo posible este problema.

Tras esto el único inconveniente que se produjo fue la localización de bugs durante el revisionado de la aplicación ya finalizada, puesto que se encontraron bugs que tenían que ver con el dispositivo y la versión de Android que tuviera el usuario, lo que entraba dentro del riesgo 2**: "Problemas técnicos de los recursos utilizados para el desarrollo de la práctica."** Y que provocaba que la aplicación, aunque funcionaba en toda su funcionalidad, esta fuera poco eficiente, haciendo entre otras cosas, casi incontrolable el movimiento de la cámara por parte del usuario por lo que se modificaron ciertas partes de la implementación y se tuvo que hacer también cambios en algunas de las partes de esta memoria.

Debido a esto el coste indirecto del proyecto ascendió a 793,63€ ya que el proyecto se demoró 42 días respecto de la fecha de entrega original.

Capítulo 3: Análisis del proyecto

3.1 Introducción

En este capítulo se mostrará todo el análisis previo a la realización del proyecto software, desde los diferentes tipos de actores y roles de las personas que se espera interactúen con la aplicación hasta los diferentes casos de uso que se podrán realizar con nuestro programa, explicando también los requisitos tanto funcionales como no funcionales que se deberán cumplir en el proyecto final ideal y mostrando los diagramas iniciales tanto del modelo de dominio como de la secuencia.

3.2 Actores y roles

El único actor que se contempla en el proyecto es el actor usuario o cliente que es el que se beneficiará de las funcionalidades de la aplicación y que será capaz de realizar todos los casos de uso que se pretenden ver cubiertos. Esto es debido principalmente a que se espera que el trabajo final sea una aplicación sencilla y no se contempla la idea de tener ningún tipo de administrador o superusuario que permita realizar más tareas que las del usuario cliente normal de la aplicación. No se contemplan casos de uso como cambios de perfil ya que el proyecto está pensado para que cada usuario cree sus huertos virtuales en un dispositivo propio.

3.3 Análisis de requisitos

Según el estándar IEEE[10]: “un requisito es una condición o capacidad que necesita el usuario para resolver un problema o conseguir un objetivo determinado, también se aplica a las condiciones que debe cumplir o poseer un sistema o una de sus componentes para satisfacer un contrato o una especificación”. En esencia los requisitos definen las características que tendrá nuestra aplicación así como su funcionalidad y señalará también los puntos que nuestra aplicación no será capaz de abarcar por múltiples factores, siendo de esta forma un contrato entre desarrollador y cliente a cerca de las propiedades de la aplicación.

Los tipos de requisitos que aparecen en los sistemas software se pueden abarcar en varios tipos que no son a su vez excluyentes:

- **Requisitos de usuario:** Servicios que ofrece nuestro proyecto y restricciones del mismo sobre las que opera.

- **Requisitos del sistema:** Describe como bien indica su nombre los requisitos del sistema y sirve como contrato entre el desarrollador y el cliente y que es especialmente útil para los desarrolladores a la hora de diseñar el sistema final.
- **Requisitos funcionales:** Describen las acciones que debe desencadenar el sistema como respuesta a las órdenes ejecutadas por los usuarios así como las capacidades del sistema.
- **Requisitos funcionales de información:** Son todos aquellos requisitos del sistema relacionados con la información que se almacenará o será capaz de almacenar el programa, dentro de estos requisitos encontramos por ejemplo la información de los usuarios como su nickname, contraseña, y diferentes configuraciones de la cuenta, por lo que son muy importantes a la hora de desarrollar productos software.
- **Reglas del negocio:** Son un tipo especial de requisito funcional que describe restricciones sobre otros requisitos funcionales, dentro de este tipo de requisitos entrarían las políticas de empresa e imposiciones legales por ejemplo.
- **Requisitos no funcionales:** Requisitos que afectan en gran medida a la calidad del servicio que ofrece nuestro sistema así como las características que debe tener el hardware para mover el software correctamente, dentro de este tipo de requisitos encontramos todos los relacionados con el tiempo de respuesta y la seguridad del software así como la utilización de determinadas herramientas y hardware como ya se comentó anteriormente.

Una vez identificados los diferentes requisitos que nos podemos encontrar a la hora de desarrollar un producto software, se procederá a realizar un estudio de los requisitos tanto funcionales como no funcionales que se han identificado de manera inicial para en este proyecto.

3.3.1 Requisitos funcionales

Los requisitos funcionales que se han identificado para este proyecto de cara a la realización de la lista de casos de uso del sistema son los siguientes:

Identificador	Descripción del requisito:
RF01	El usuario deberá ser capaz de iniciar un proyecto nuevo.
RF02	El usuario deberá ser capaz de elegir la dimensión de terreno que desea simular.
RF03	El usuario deberá ser capaz de borrar un proyecto existente.
RF04	El usuario deberá ser capaz de abrir un proyecto ya existente.

RF05	El usuario deberá ser capaz de modificar un proyecto ya existente.
RF06	El usuario deberá ser capaz de guardar las modificaciones en un proyecto.
RF07	El usuario deberá ser capaz de colocar diferentes tipos de plantas en el plano.
RF08	El usuario deberá ser capaz de borrar una planta que aparezca en el plano.
RF09	El usuario deberá ser capaz de generar un sistema óptimo de riego en el plano.
RF10	El usuario deberá ser capaz de visualizar la información relativa a las plantas que escoja.
RF11	El usuario deberá ser capaz de volver al menú inicial desde el juego.
RF12	El usuario deberá ser capaz de visualizar las diferentes relaciones entre las plantas que hay actualmente en el pleno.
RF13	El usuario deberá ser capaz de cerrar la aplicación en cualquier momento.

Tabla 22: Requisitos funcionales del sistema

3.3.2 Requisitos funcionales de baja prioridad

Se han identificado además una serie de requisitos funcionales de baja prioridad que no se implementarán en el entregable final de este proyecto, pero que se estudia podrían ser beneficiosos para la mejora de la aplicación en un futuro:

Identificador	Descripción del requisito
RFBP01	El usuario deberá ser capaz de editar las dimensiones del huerto en tiempo de ejecución
RFBP02	El usuario deberá ser capaz de girar la cámara para observar el plano desde diferentes ángulos
RFBP03	El usuario deberá ser capaz de copiar el plano en diferentes slots de guardado.
RFBP04	El usuario deberá ser capaz de cargar planos de forma externa a la aplicación.
RFBP05	El usuario deberá ser capaz de elegir el tipo de riego del huerto.

RFBP06	El usuario deberá ser capaz de mover la posición del riego.
RFBP07	El usuario deberá ser capaz de visualizar si el riego está colocado correctamente en el plano para que llegue a todas las plantas colocadas en el mismo.

Tabla 23: Requisitos funcionales de baja prioridad

3.3.3 Requisitos funcionales de información

Nuestro sistema no requerirá de almacenar apenas datos, ya que se prevé que se ejecute en un sistema móvil como una Tablet o similares con un sistema Android por lo que se ha decidido que no requiera de identificación por parte del usuario. Los requisitos funcionales de información que se han identificado para el sistema son:

Identificador	Descripción del requisito
RFI01	El sistema deberá almacenar los nombres que el usuario haya decidido poner a sus huertos.
RFI02	El sistema deberá almacenar las dimensiones de los huertos creados por el usuario.
RFI03	El sistema deberá almacenar los datos relativos a las plantas colocadas, tanto el tipo de planta como su posición en el plano.

Tabla 24: Requisitos funcionales de información

3.3.4 Requisitos no funcionales

A continuación se expondrá los requisitos no funcionales del sistema que se han identificado inicialmente para este proyecto junto con su impacto en el sistema final, ya sea crítica, alta, media o baja.

Identificador	Relevancia	Descripción del requisito:
RNF01	Crítica	El proyecto deberá ser realizado en Unity.
RNF02	Media	El programa deberá ser capaz de ejecutarse en un sistema Android.
RNF03	Alta	El sistema deberá ser capaz de generar un plano en menos de 1 segundo.
RNF04	Crítica	El sistema deberá ser capaz de realizar cálculos acerca de las relaciones entre las plantas mostrándolas por pantalla en tiempo de ejecución y de forma casi instantánea.
RNF05	Alta	El sistema deberá ser capaz de guardar la información del proyecto en cualquier momento y de forma instantánea permitiendo también su sobreescritura.

RNF06	Media	El sistema deberá ser capaz de reescalar los menús, el plano y las plantas en función del dispositivo que ejecute el programa.
RNF07	Media	El sistema deberá ser capaz de ejecutarse en la versión MIUI 12.0.3 de Android.
RNF08	Alta	El sistema deberá ejecutarse en Tablets de forma prioritaria para aprovechar mejor la resolución de estas.

Tabla 25: Requisitos no funcionales del sistema

3.4 Casos de Uso

Un caso de uso tal y como se explica en los apuntes de la asignatura de la carrera "Modelado de software" [10] es una secuencia de acciones realizadas por el sistema, que producen un resultado valioso para un actor en particular. Esta definición se puede simplificar, para que se entienda más fácil, en una serie de acciones que desencadenan eventos en el sistema y que producen una serie de salidas que el usuario es capaz de captar.

Para obtener estos casos de uso se han estudiado los requisitos funcionales de la aplicación para obtener de esta forma las diferentes acciones que debe ser capaz de realizar la aplicación final. Además los actores encargados de llevar a cabo los diferentes casos de uso en la aplicación siempre serán los mismos, y sólo uno, el usuario particular que maneje la aplicación.

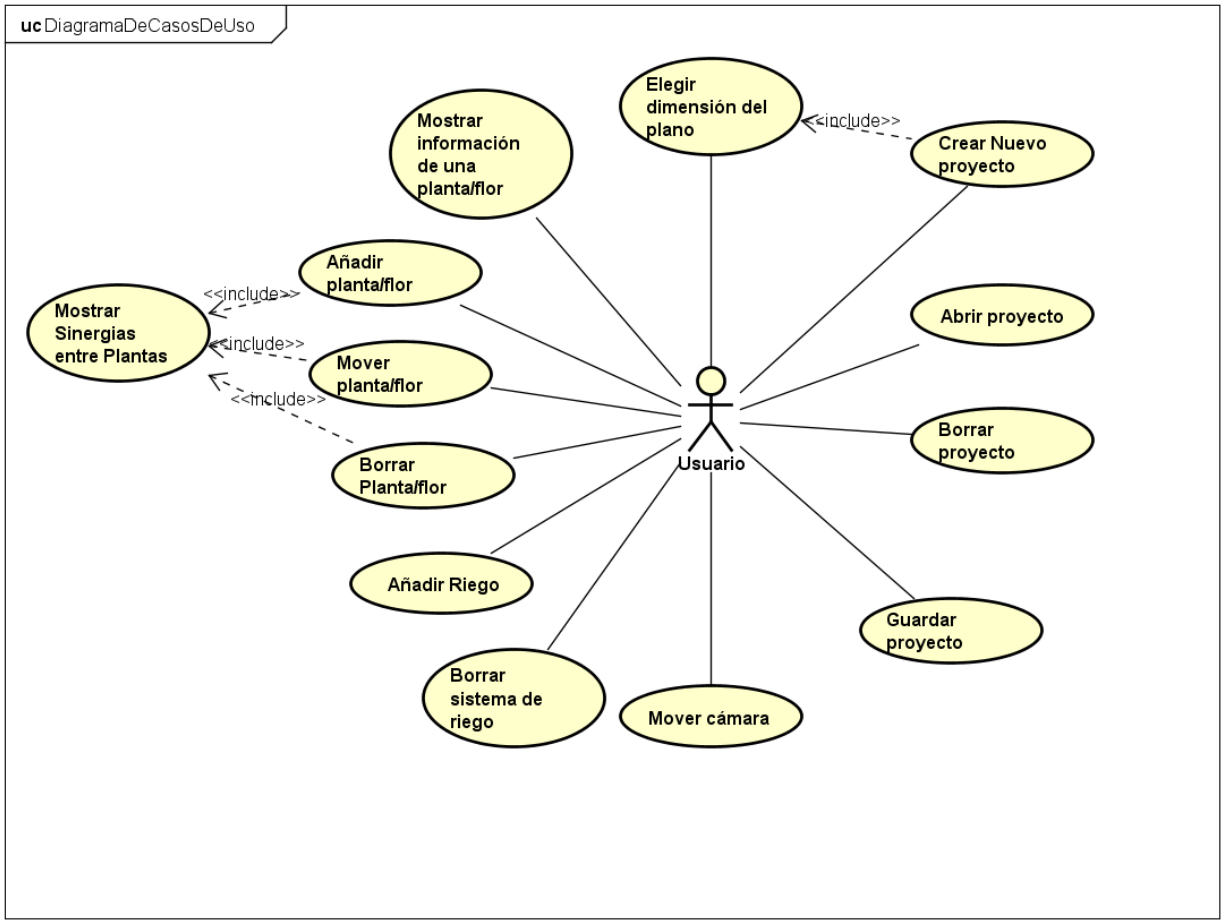

Figura 6: Diagrama inicial de Casos de uso.

3.5 Especificación de los casos de uso

En esta sección se explicarán detalladamente los diferentes casos de uso que se han mostrado en el diagrama anterior, así como el actor que realizará el caso de uso, la condición que se debe cumplir para que se realice el caso de uso, su secuencia de acciones normal, y su secuencia de acciones alternativa.

CU1	**Crear nuevo proyecto**
Actor	Usuario.
Precondición	Ninguna.
Secuencia normal	1- El usuario pulsará el botón “Nuevo proyecto”. 2- El sistema abrirá la escena de trabajo. 3- El usuario escogerá un nombre para el nuevo proyecto y pasará a realizar el caso de uso 6: Elegir las dimensiones del plano. 4- El Usuario pulsará el botón “Aceptar”
Secuencia Alternativa y excepciones	1- El usuario cancela la operación pulsando el botón “Cancelar” y el caso de uso queda sin efecto. 2- Alguno de los campos introducidos por el usuario es incorrecto, por lo que se mostrará un mensaje de error y el caso de uso quedará sin efecto.
Postcondición	Ninguna.

Tabla 26: Caso de uso “Crear nuevo proyecto/jardín”.

CU2	**Abrir proyecto**
Actor	Usuario.
Precondición	• Deberá existir un proyecto guardado previamente.
Secuencia normal	1- El usuario pulsará uno de los proyectos existentes y dicho proyecto se abrirá.
Secuencia Alternativa y excepciones	No existe secuencia alternativa para este caso de uso.
Postcondición	• Se pasará a la escena de edición que mostrará el proyecto seleccionado.

Tabla 27: Caso de uso “Abrir Proyecto”.

CU3	**Borrar proyecto**
Actor	Usuario.
Precondición	• Deberá existir un proyecto guardado previamente.
Secuencia normal	1- El usuario pulsará el botón de borrar al lado del proyecto. 2- El sistema borrará el fichero con los datos del proyecto.
Secuencia Alternativa y excepciones	No existe secuencia alternativa para este caso de uso.
Postcondición	• El proyecto seleccionado se borrará de la aplicación.

Tabla 28: Caso de uso “Borrar proyecto/jardín”.

CU4	**Guardar proyecto**
Actor	Usuario.
Precondición	• El usuario deberá haber creado un proyecto nuevo o abierto uno ya existente.
Secuencia normal	1- El usuario pulsará el botón de guardar proyecto. 2- El sistema creará un fichero con los datos del proyecto actual.
Secuencia Alternativa y excepciones	No existe secuencia alternativa para este caso de uso.
Postcondición	• Las modificaciones del proyecto quedarán guardadas en la aplicación.

Tabla 29: Caso de uso "Guardar Proyecto".

CU5	**Elegir dimensiones del plano.**
Actor	Usuario.
Precondición	• El usuario deberá haber abierto un proyecto o haber creado uno nuevo. • Las dimensiones no podrán ser menores a 10 casillas en el eje X del plano. • Las dimensiones no podrán ser mayores a 10 casillas en el eje Y del plano.
Secuencia normal	1- El usuario indicará las dimensiones en ancho y largo del plano que desea generar. 2- El usuario pulsará el botón de Aceptar.
Secuencia Alternativa y excepciones	1- El usuario cancela la operación y el caso de uso queda sin efecto.
Postcondición	• Se creará un plano con las dimensiones seleccionadas previamente por el usuario.

Tabla 30: Caso de uso "Elegir dimensiones del plano".

CU6	**Mover la cámara**
Actor	Usuario
Precondición	• El usuario deberá haber abierto un proyecto o haber creado uno nuevo.
Secuencia normal	1- El usuario pulsará en cualquier punto del plano y deslizará el dedo. 2- El sistema moverá la cámara en la dirección contraria a la del dedo.
Secuencia Alternativa y excepciones	No existe secuencia alternativa para este caso de uso.
Postcondición	• La cámara cambiará de posición.

Tabla 31: Caso de uso "Mover cámara".

CU7	**Añadir planta o flor**
Actor	Usuario
Precondición	• El usuario deberá haber abierto un proyecto o haber creado uno nuevo.
Secuencia normal	1- El usuario pulsará el botón de añadir planta. 2- El sistema abrirá el menú de plantas. 3- El usuario pulsará la planta que desee de la lista de plantas que le aparecerán en el menú. 4- El sistema instanciará la planta en el primer hueco del plano que quede libre.
Secuencia Alternativa y excepciones	1- El usuario cancela la operación y el caso de uso queda sin efecto. 2- El plano carece de espacios libres donde colocar nuevas plantas, por lo que el caso de uso queda sin efecto
Postcondición	• Se añadirá la planta seleccionada en la primera casilla del plano libre empezando en la esquina superior izquierda y avanzando de izquierda a derecha y de arriba hacia abajo.

Tabla 32: Caso de uso "Añadir planta o flor".

CU8	**Mover planta o flor**
Actor	Usuario
Precondición	• El usuario deberá haber abierto un proyecto o haber creado uno nuevo.
Secuencia normal	1- El usuario mantendrá pulsada una planta del plano. 2- El sistema cambiará la planta cambiará de color lo que indicará que puede ser trasladada. 3- El usuario arrastrará la planta hasta una casilla libre y dejará de pulsar la planta. 4- El sistema colocará la planta en la casilla a la que ha sido arrastrada.
Secuencia Alternativa y excepciones	1- El usuario cancela la operación y el caso de uso queda sin efecto.
Postcondición	• La planta se moverá de la casilla inicial a la casilla donde la haya arrastrado el usuario.

Tabla 33: Caso de uso "Mover planta o flor".

CU9	**Borrar planta o flor**
Actor	Usuario
Precondición	• El usuario deberá haber abierto un proyecto o haber creado uno nuevo. • Deberá existir una planta en el plano.
Secuencia normal	1- El usuario pulsará una planta. 2- El usuario moverá la planta fuera del plano. 3- El sistema borrará la planta de la escena.
Secuencia Alternativa y excepciones	1- El usuario cancela la operación y el caso de uso queda sin efecto.
Postcondición	• La planta será eliminada del plano.

Tabla 34: Caso de uso "Borrar planta o flor".

CU10	**Añadir sistema de riego**
Actor	Usuario
Precondición	• El usuario deberá haber creado un proyecto nuevo o abierto un proyecto ya existente.
Secuencia normal	1. El usuario pulsará el botón de añadir sistema de riego. 2. El sistema generará un sistema de riego óptimo para el plano que hemos diseñado.
Secuencia Alternativa y excepciones	No existe secuencia alternativa para este caso de uso.
Postcondición	• Se creará un sistema de riego óptimo para el huerto.

Tabla 35: Caso de uso "Añadir sistema de riego".

CU11	**Borrar sistema de riego**
Actor	Usuario
Precondición	• Deberá haber un sistema de riego activo en el plano.
Secuencia normal	1. El usuario pulsará el botón de añadir/borrar sistema de riego. 2. El sistema eliminará le riego se eliminará del plano.
Secuencia Alternativa y excepciones	No existe secuencia alternativa para este caso de uso.
Postcondición	• El sistema de riego se eliminará del plano.

Tabla 36: Caso de uso "Borrar sistema de riego".

CU12	**Mostrar sinergias entre plantas**
Actor	Usuario
Precondición	• Deberá haber mínimo 2 plantas colindantes situadas en el plano.
Secuencia normal	1. El sistema actualizará cada frame el estado del plano para mostrar las relaciones entre las plantas colindantes, por lo que este estado sólo se verá perturbado en el caso en el que el usuario realice cambios sobre el plano.
Secuencia Alternativa y excepciones	No existe secuencia alternativa para este caso de uso.
Postcondición	• Se mostrarán si las hay las relaciones de beneficio o perjuicio entre las plantas colindantes.

Tabla 37: Caso de Uso "Mostrar sinergias entre plantas".

CU13	**Mostrar información de la planta/flor**
Actor	Usuario
Precondición	• El usuario deberá encontrarse ante el menú de plantas.
Secuencia normal	1. El usuario pulsará el botón de información de una planta. 2. El sistema mostrará la información de la planta que aparecerá por pantalla.
Secuencia Alternativa y excepciones	No existe secuencia alternativa para este caso de uso.
Postcondición	• Se mostrará la información de la planta/flor seleccionada por el usuario.

Tabla 38: Caso de Uso "Mostrar información de la planta/flor".

3.6 Modelo de Dominio inicial

Se mostrará a continuación el modelo de dominio obtenido inicialmente para la aplicación, teniendo en cuenta que es probable que se realicen cambios en el transcurso del desarrollo del proyecto ya que seguimos un modelo de desarrollo del proceso unificado, y sobre todo debido a que Unity es una herramienta nueva para mí y es posible que las consideraciones y propuestas que se hagan en un primer momento tengan que ser descartadas bien porque no se puedan realizar con c#[14], el lenguaje que utiliza principalmente Unity para sus scripts, o bien porque Unity proporcione algún tipo de herramienta que permita realizar el mismo cometido con menos esfuerzo o con mayor calidad.

El modelo de dominio inicial pensado para OrtiDis, la herramienta de ayuda de diseño de huertos, es el mostrado a continuación, cabe destacar que está sujeto a cambios con respecto al diseño final de la aplicación:

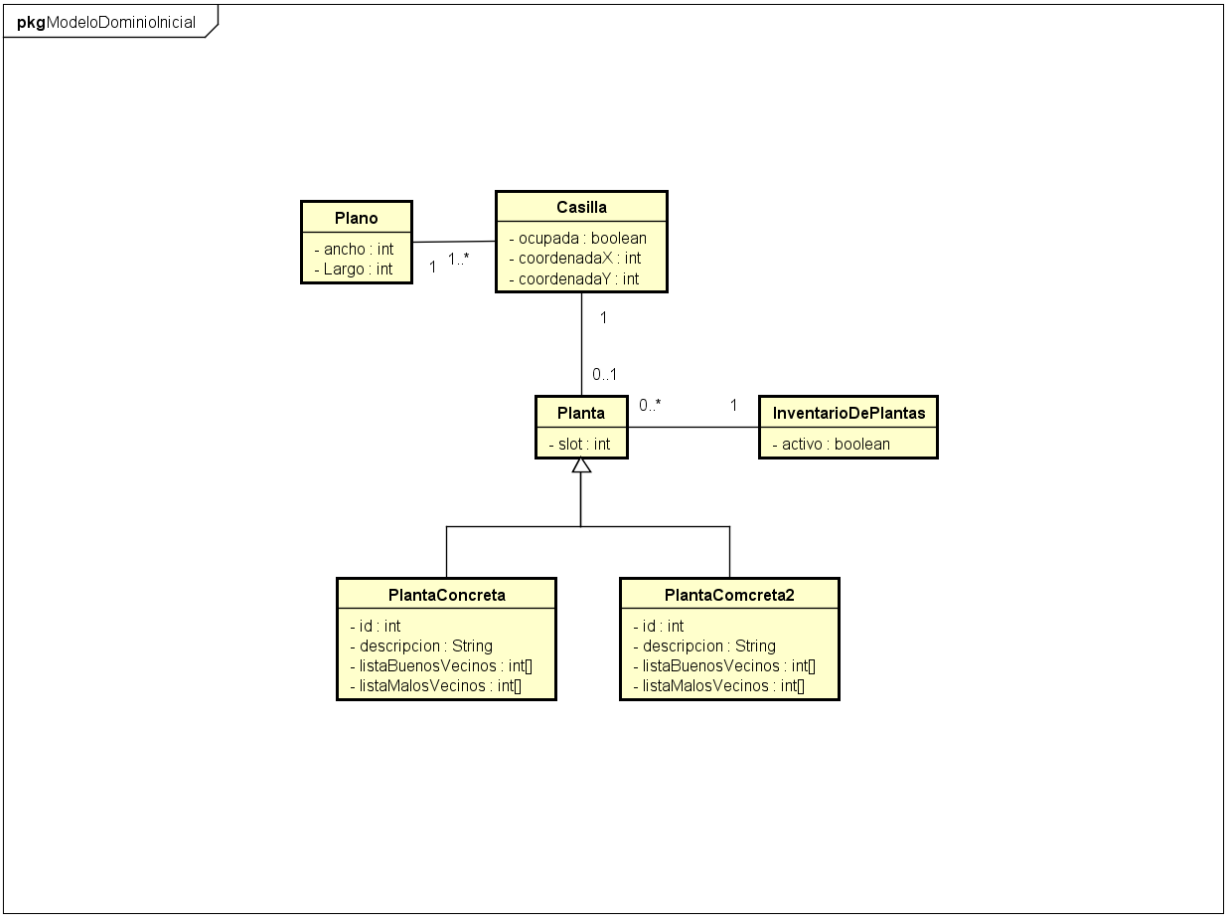

Figura 7: Modelo de dominio conceptual inicial de la aplicación

Como se puede apreciar nuestra aplicación en un primer momento no necesita de demasiadas clases. Dispondrá de 2 escenas, que corresponderían con las "Vistas" de un modelo MVC, un menú inicial donde se mostrarán los slots de guardado de los diferentes huertos creados, y una escena de juego donde se creará el plano y se podrá

interactuar con él y que contará con la mayor parte de la funcionalidad del juego. A través de estas escenas se navegará mediante una clase controlador, además de que también se requerirá de una clase que modelo siguiendo el patrón MVC que almacenará ciertos datos de interés como las dimensiones del plano que serán necesarias para realizar ciertas operaciones, o incluso el nombre que el usuario haya dado al plano.

3.7 Modelo de clases inicial

El diagrama de clases inicial de la aplicación en esta fase de análisis tendrá el siguiente aspecto:

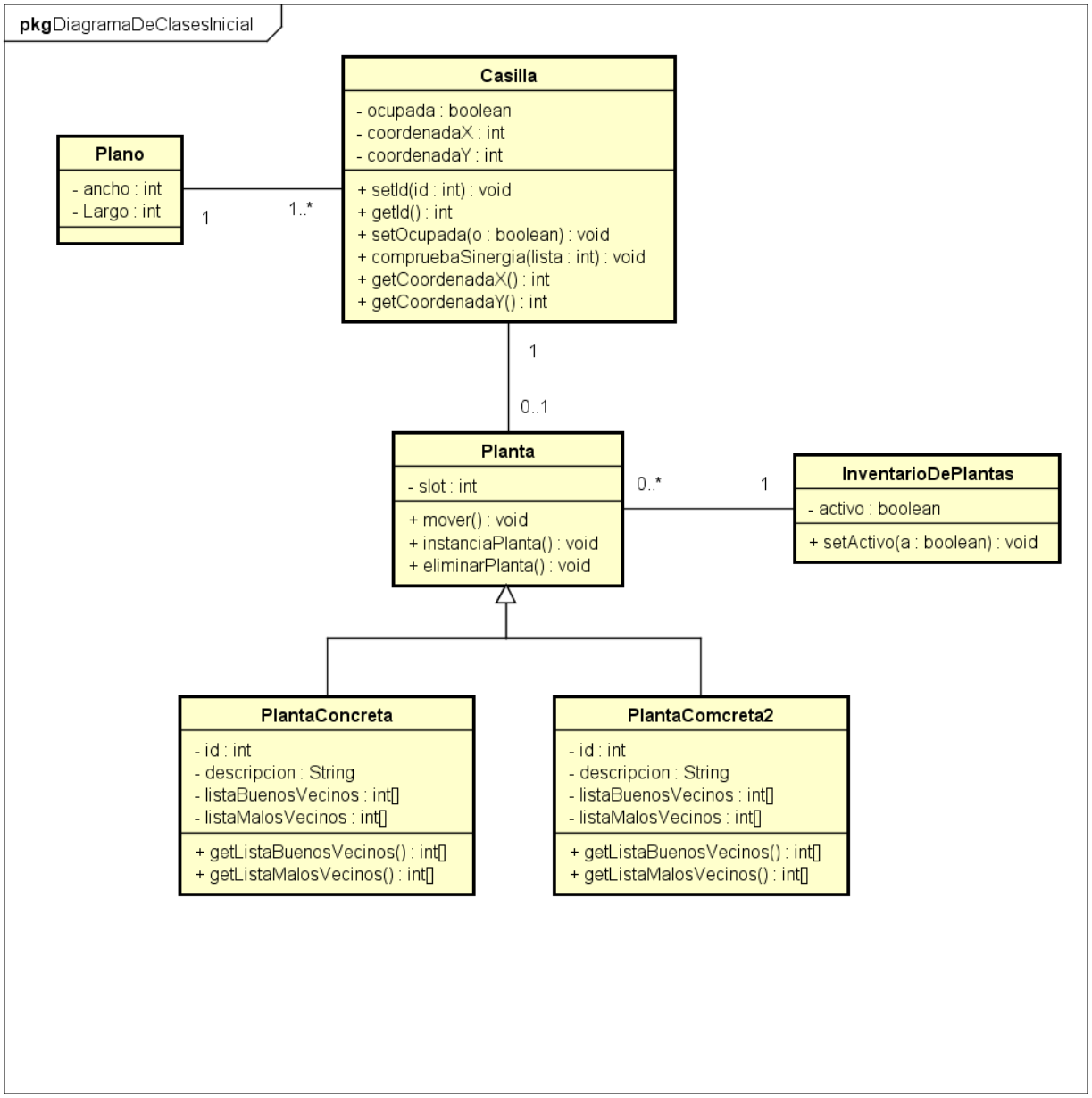

Figura 8: Diagrama de clases inicial de la aplicación

Se pueden observar que el diagrama de clases coincide con el modelo de dominio en esta fase de análisis de la aplicación. Las operaciones de las diferentes clases en un primer momento no son demasiadas pero varias de ellas serán complejas a la hora de implementarse en el programa, teniendo especial importancia los métodos de la clase "Planta" ya que será la encargada de gestionar el movimiento de las mismas así como su creación y eliminación y que probablemente requiera de utilizar algún tipo de librería de las que disponga ya Unity. También es muy importante destacar la labor de la clase "Casilla" ya que será la encargada de evaluar las sinergias entre plantas, es decir, será la encargada de comprobar mediante el id de la planta que se aloje en ese momento en esa casilla y mediante la lista que cada planta concreta tiene de sus buenos y malos vecinos, si la sinergia con las plantas que se encuentren en casillas de alrededor, es buena o mala, y mostrará en pantalla esta sinergia para que el usuario sea capaz de visualizar las relaciones entre las plantas. Esta operación debe aparecer en casilla en vez de en planta, ya que la planta no será consciente de su posición en el plano, mientas que las casillas tendrán mediante un identificador similar a unas coordenadas, pero en números enteros, su posición en el plano y de esta forma poder relacionarse con las casillas colindantes.

3.8 Diagrama de secuencia o realización de casos de uso en análisis

A continuación se expondrán los diagramas de secuencia correspondientes a la realización de los casos de uso incidiendo en la interacción del usuario con el sistema:

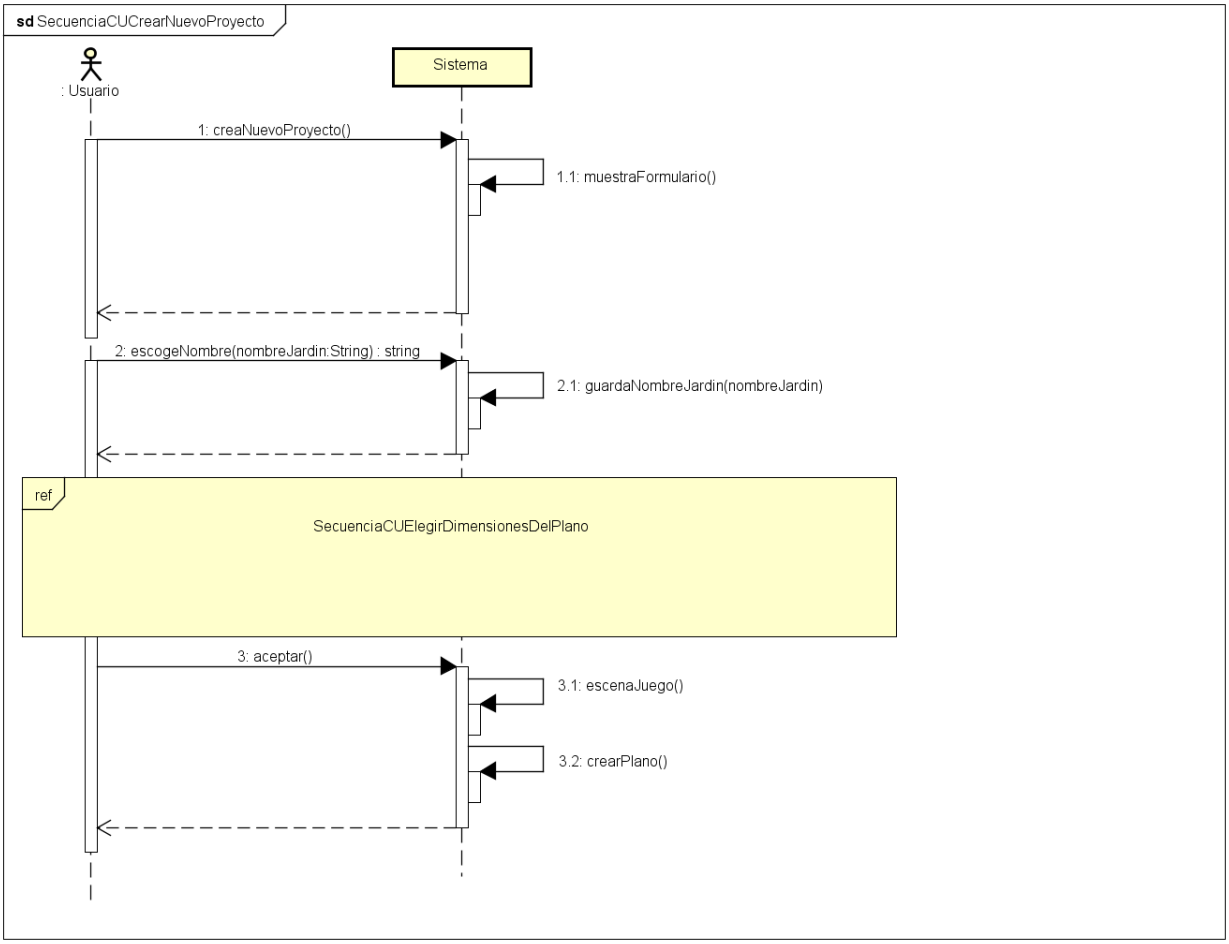

Figura 9: Diagrama de secuencia del CU "Crear nuevo proyecto" incluye al CU "Elegir dimensiones del plano".

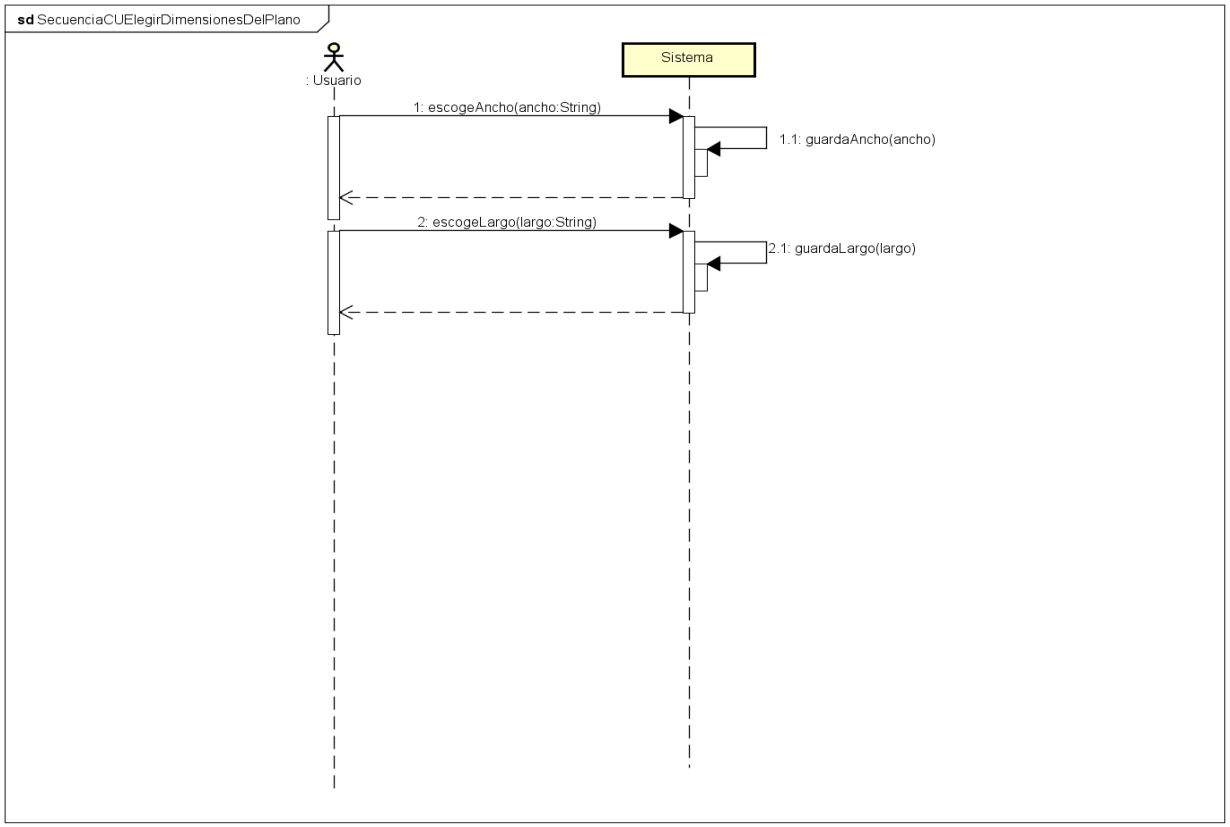

Figura 10: Diagrama de Secuencia del CU "Elegir dimensiones del plano".

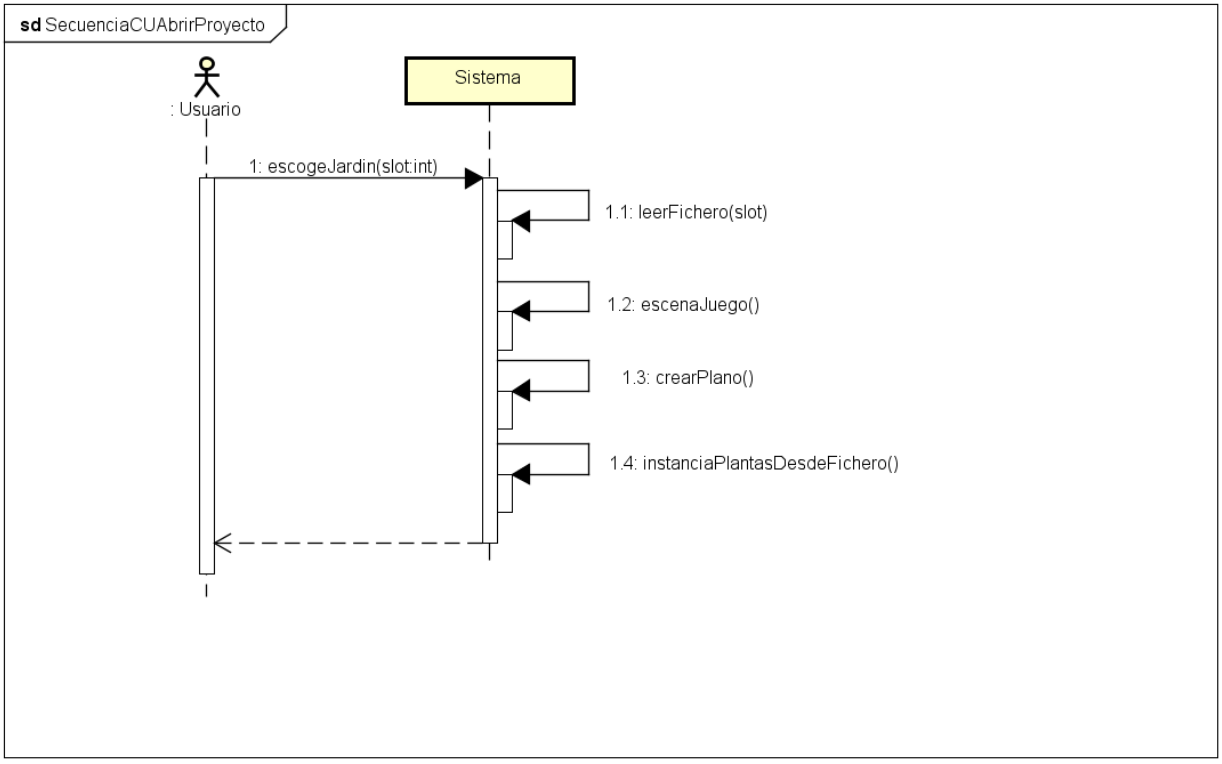

Figura 11: Diagrama de Secuencia del CU "Abrir proyecto".

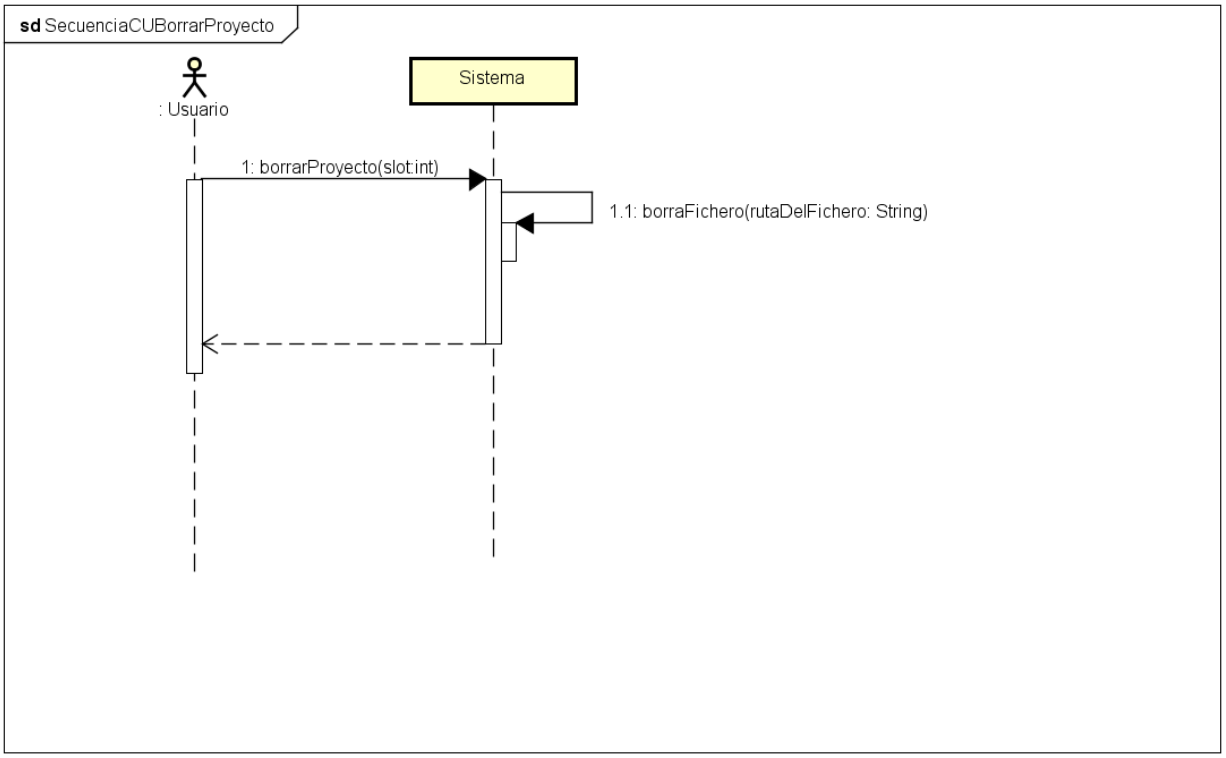

Figura 12: Diagrama de Secuencia del CU "Borrar proyecto".

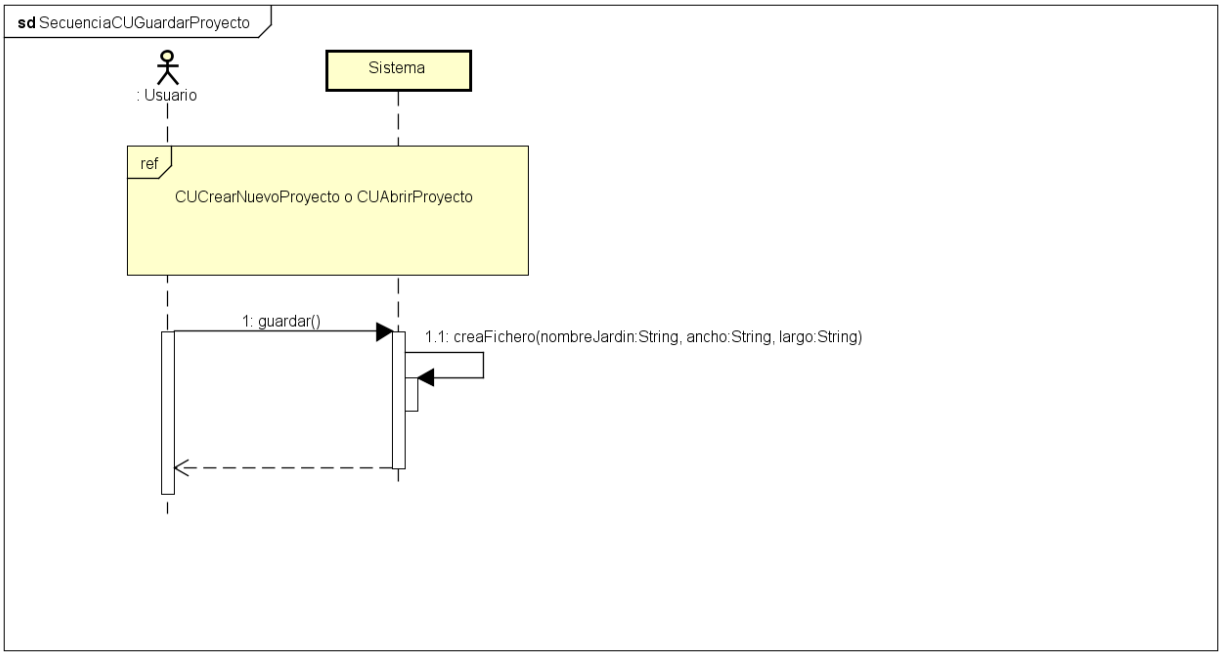

Figura 13: Diagrama de Secuencia del CU "Guardar proyecto".

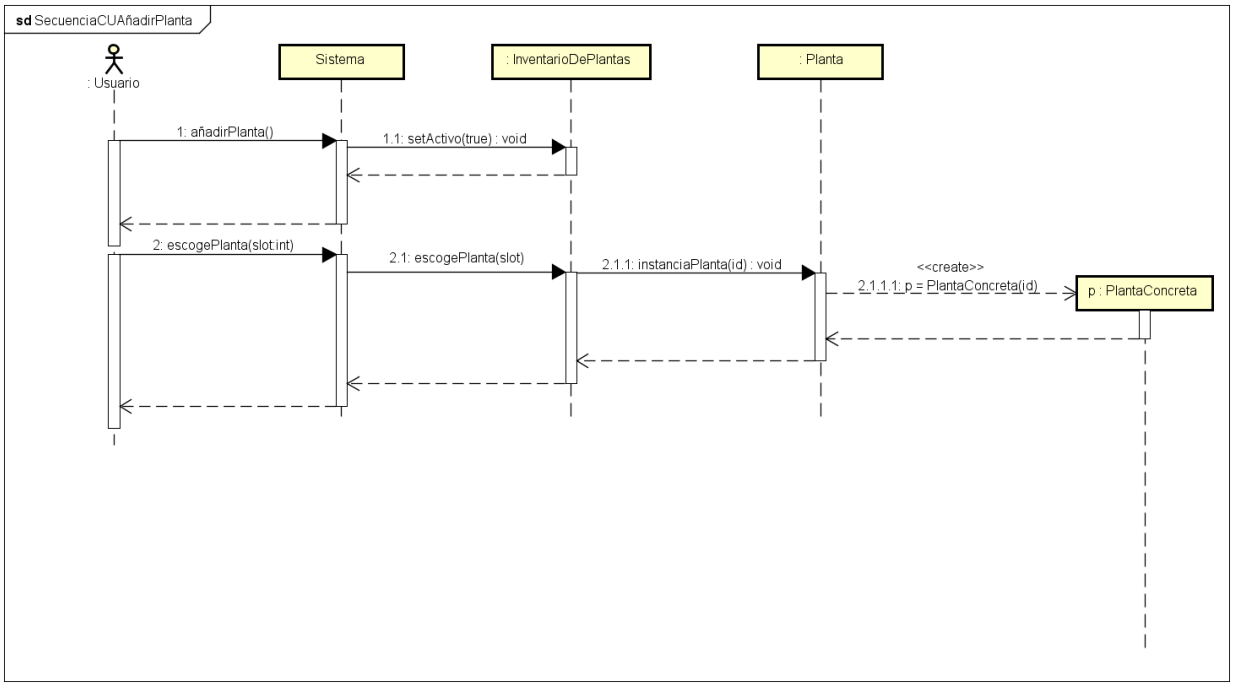

Figura 14: Diagrama de Secuencia del CU "añadir planta".

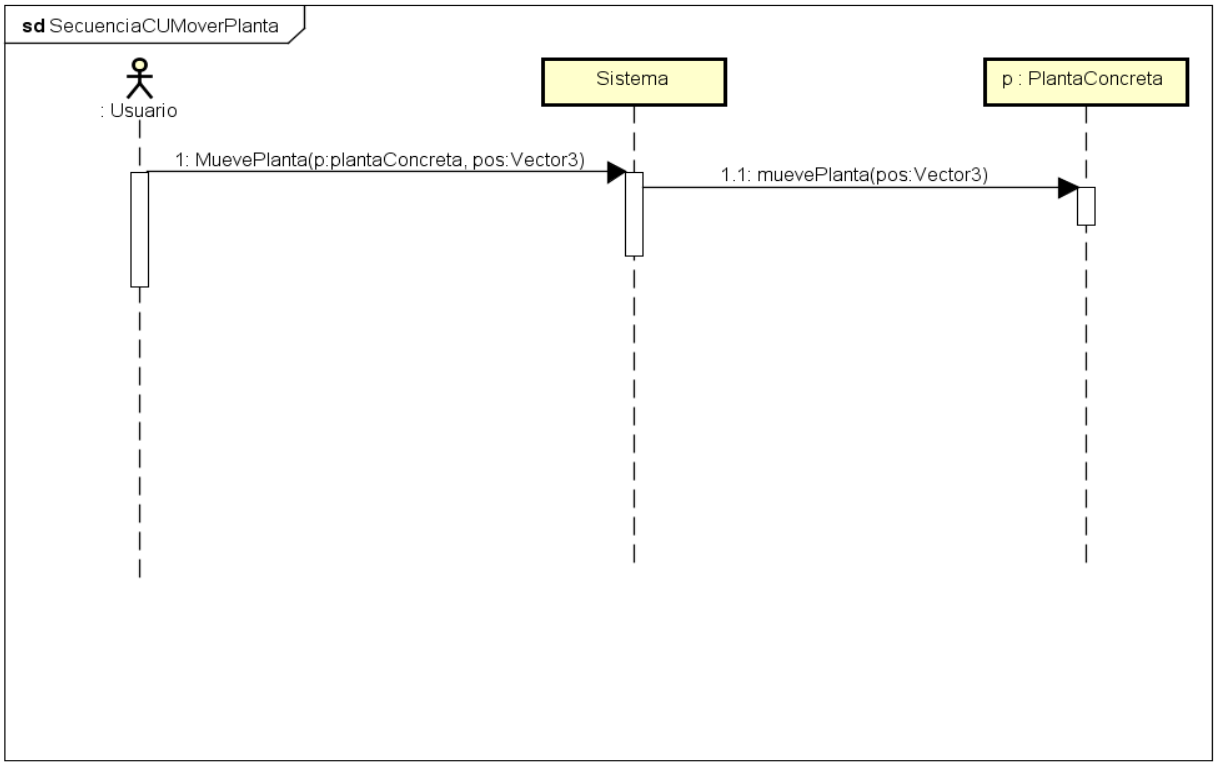

Figura 15: Diagrama de Secuencia del CU "Mover Planta".

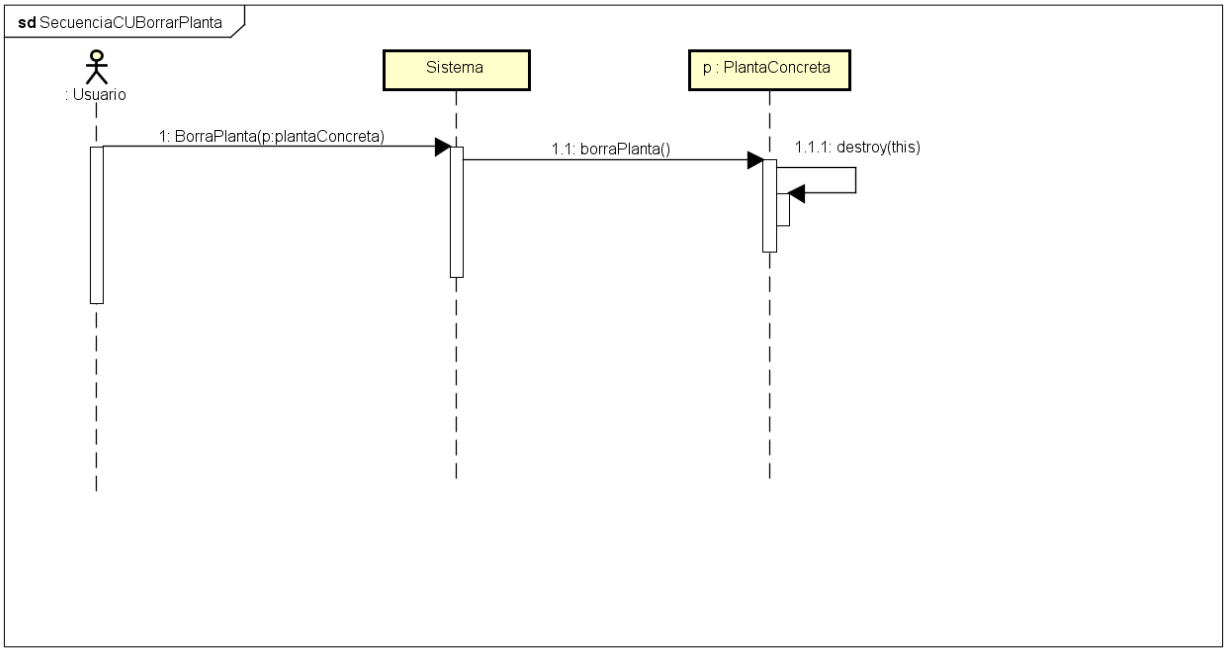

Figura 16: Diagrama de Secuencia del CU "Borrar planta".

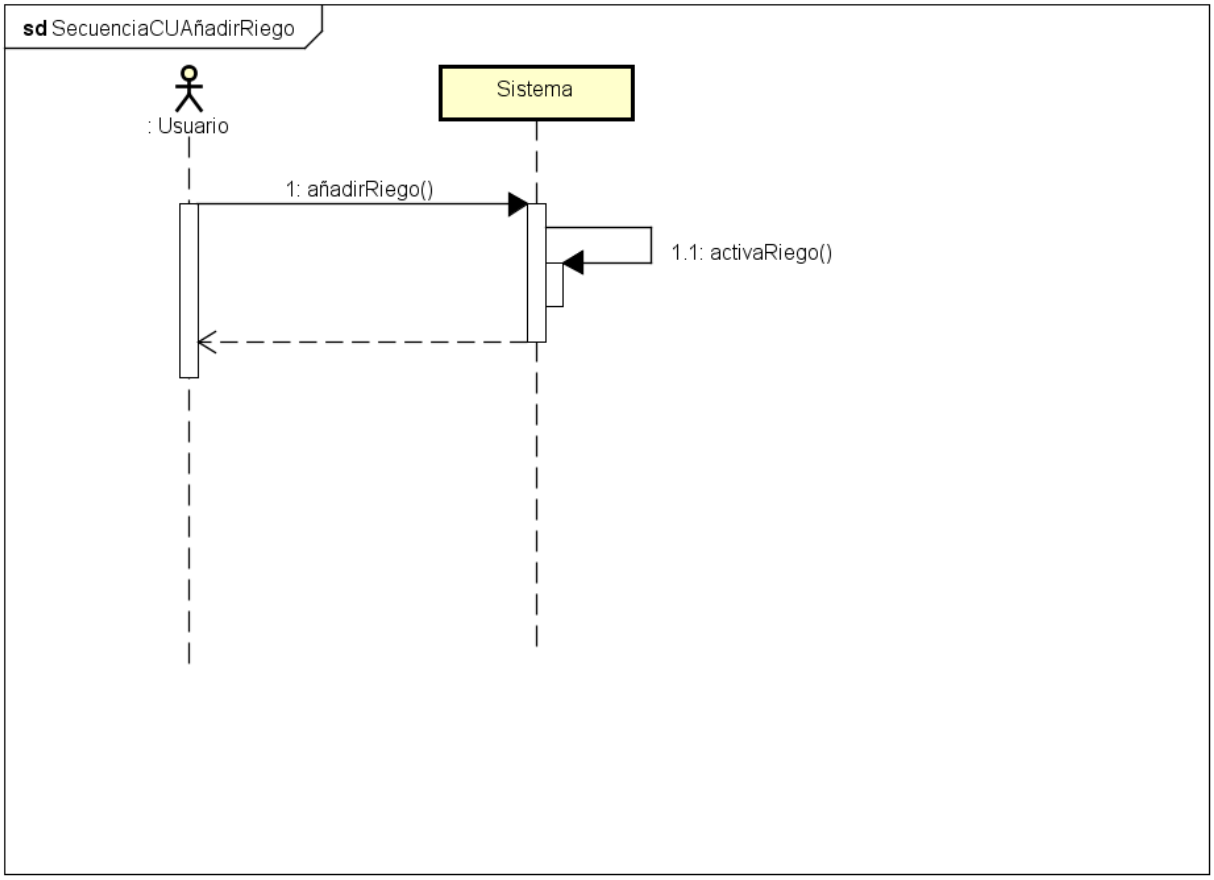

Figura 17: Diagrama de Secuencia del CU "Añadir riego".

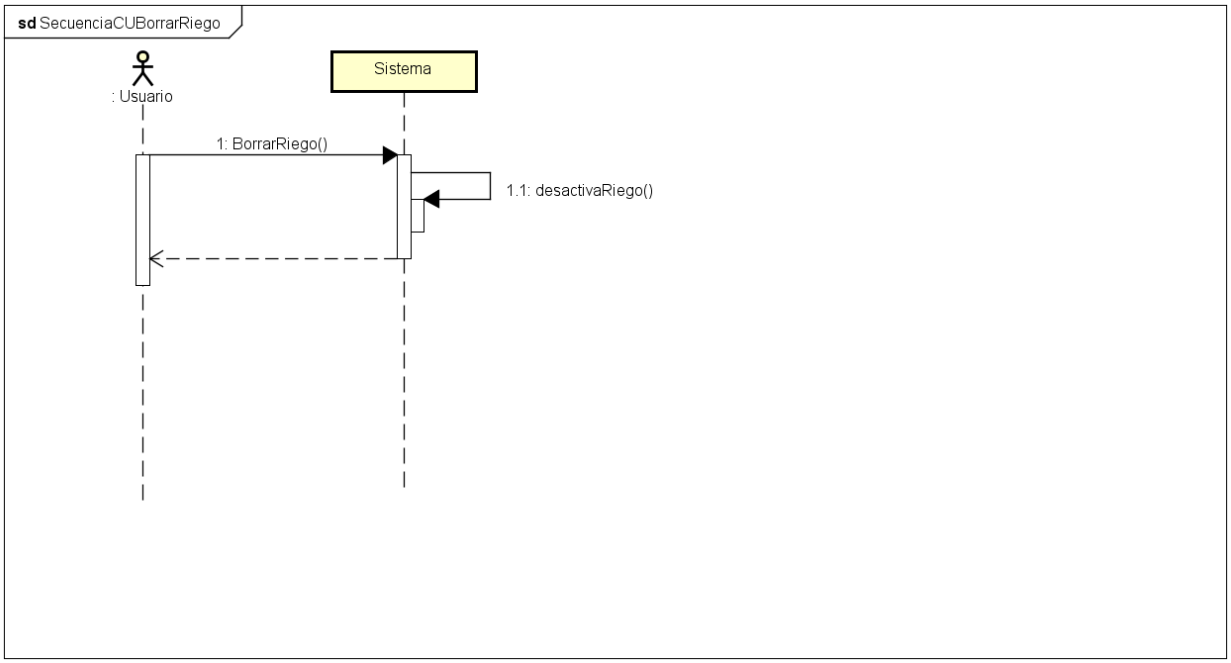

Figura 18: Diagrama de Secuencia del CU "Borrar riego".

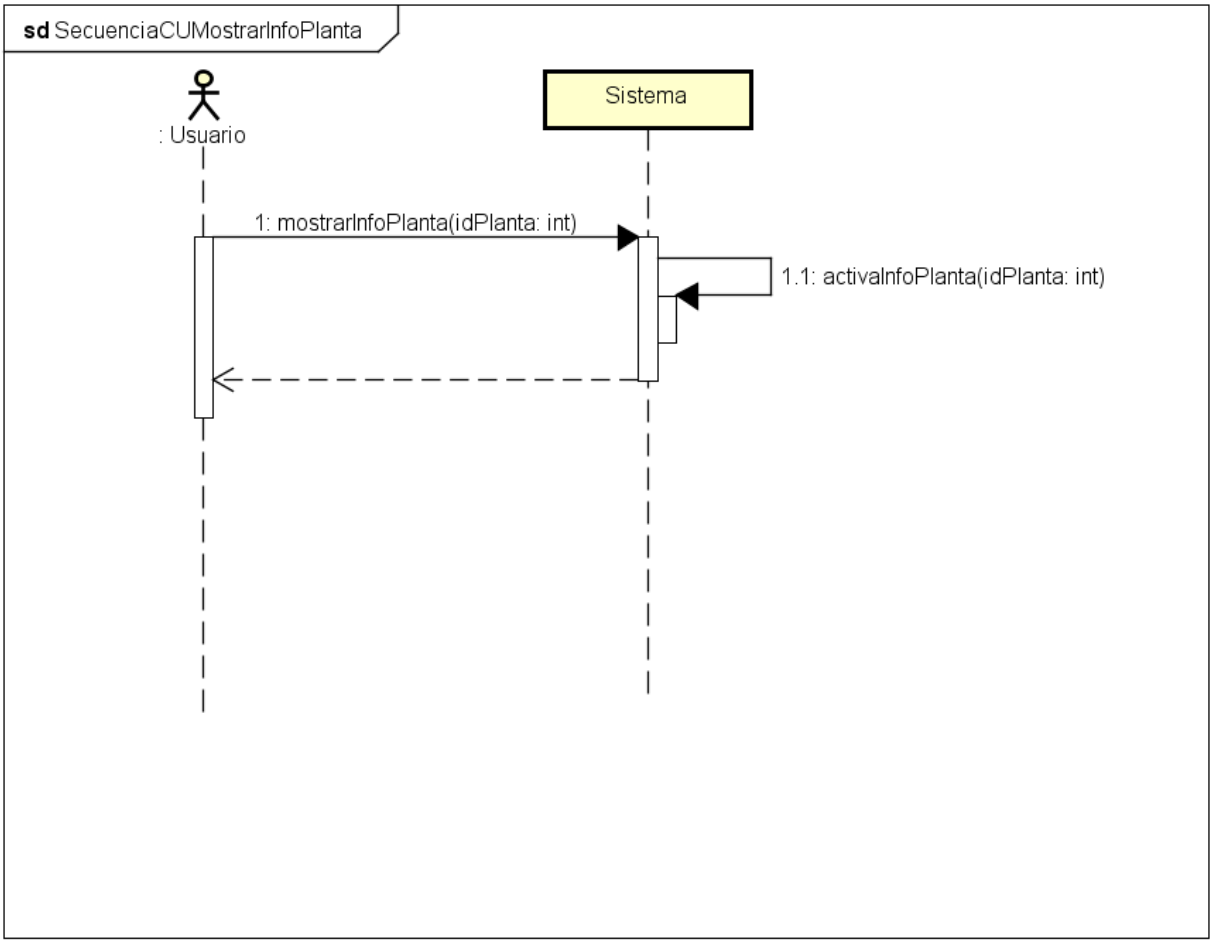

Figura 19: Diagrama de Secuencia del CU "Mostrar información de la planta/flor".

Capítulo 4: Arquitectura y Diseño del proyecto

4.1 Introducción

A continuación se tratarán de explicar lo más claramente posible las decisiones que se han tomado a la hora de diseñar el sistema final. En este proyecto finalmente el diseño final ha cambiado sustancialmente con respecto al sistema analizado en el capítulo anterior, debido principalmente al desconocimiento previo de algunas de las características que presenta Unity. Así pues, además de mostrar el diseño final al que se ha llegado, se pondrá especial atención a los cambios realizados y se tratará de explicar de forma clara el diseño final de la aplicación y la razón por la que ha habido que realizar los cambios mencionados con respecto a la fase de análisis del proyecto.

4.2 Arquitectura del sistema

Lo primero que podemos observar en la aplicación que se desea implementar es que no requiere de un avanzado sistema de gestión de datos ni de almacenamiento de los mismos, por lo que no se ha pensado una arquitectura ni se han utilizado patrones de acceso a datos salvo la lectura y escritura de fichero. Además en este primer prototipo de la aplicación, tampoco requerimos de un servidor ya que todo el peso de la aplicación estará en el cliente que la alojará, por lo que seguir una arquitectura cliente-servidor queda descartada para este primer prototipo entregable, aunque puede quedar abierta a una posible mejora del producto en un futuro.

Antes de exponer la arquitectura utilizada para el prototipo final se expondrán una serie de clases y herramientas internas que proporciona Unity y que son necesarias para entender la aplicación y que también servirán para aclarar algunas de las decisiones tomadas.

4.2.1 Modelo de Dominio de Unity

Para poder explicar mejor el funcionamiento de Unity es importante mostrar su Modelo de Dominio obtenido del TFG de Carlos López Garcinuño[4], y en el cual podemos apreciar varias de las clases más importantes que proporciona Unity y en las cuales se basa principalmente su funcionamiento.

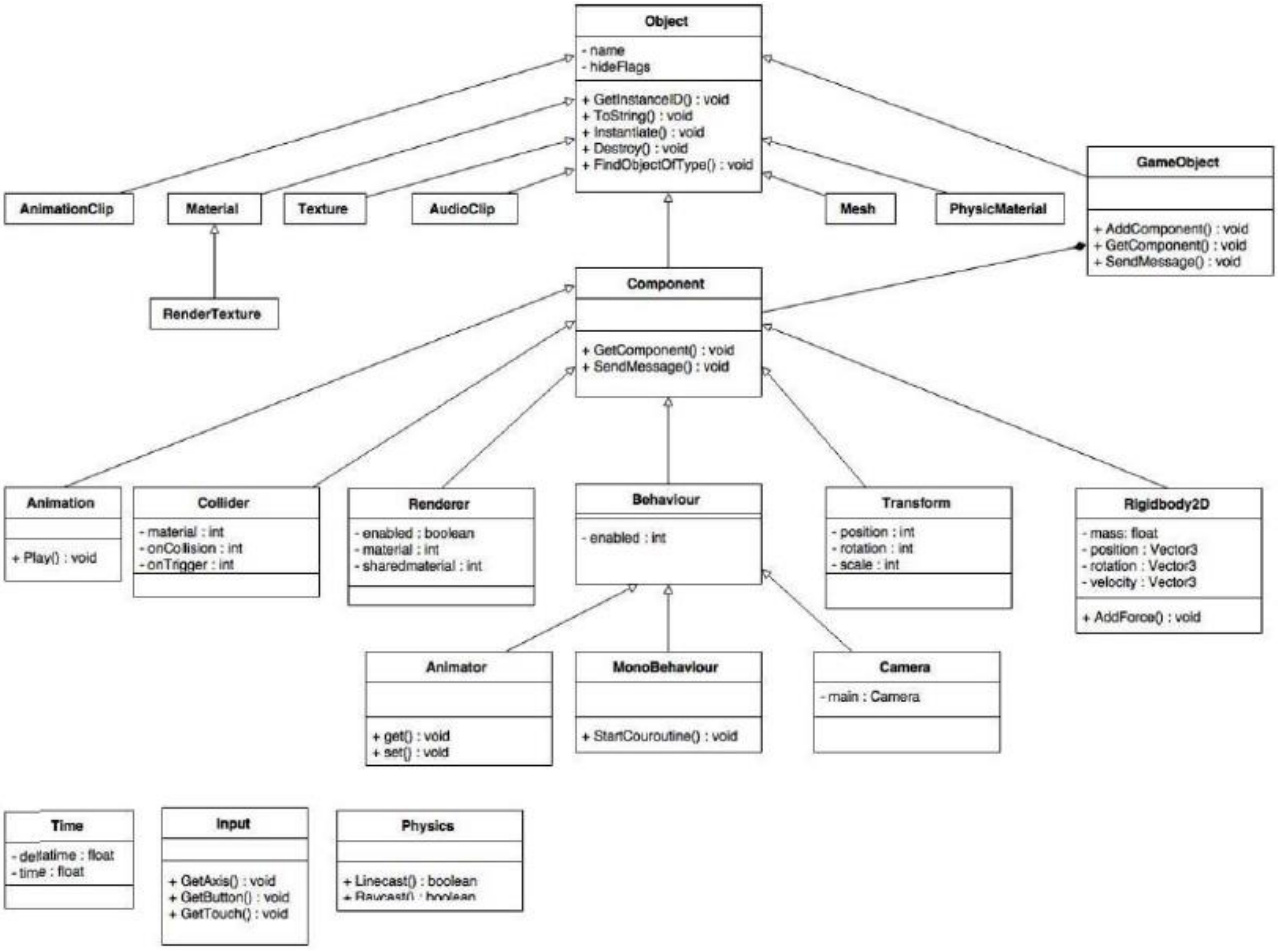

Figura 20: Modelo de Dominio de Unity.

Unity utiliza una clase llamada GameObject que sirve para la creación de diferentes objetos que se utilizarán en la escena, desde un plano, un cubo o incluso la cámara heredan de esta clase GameObject, a la cual se deben añadir componentes que serán los que le dan su funcionalidad, muchos de los cuales son proporcionados por Unity, y que van desde componentes que añaden físicas al GameObject hasta componentes básicos que dan textura o aportan un sprite, etc. Dentro de todas las posibilidades que tienen estos componentes se encuentra la capacidad de añadir Scrips a estos GameObjects, que serán las clases que se han desarrollado externamente ya que no son proporcionados por Unity como es lógico, y que serán los que aportarán la gran mayoría de la funcionalidad que tendrá el GameObject y que puede modificarlo enormemente incluso modificar o desactivar y activar los propios componentes del mismo. Los GameObjects pueden además guardarse creando lo que en Unity denominan prefab, que no son más que GameObjects prefabricados por los desarrolladores, y que se utilizan principalmente en este y en cualquier otro proyecto Unity para instanciar GameObjects concretos que cuenten ya con ciertos componentes y scripts que les den diferentes funcionalidades según el uso que se les desee dar.

Todo lo explicado anteriormente es muy importante tenerlo en cuenta ya que como se mostrará a continuación el uso de prefabs y GameObjects es constante además de que gran parte de la funcionalidad del proyecto se encontrará encapsulada en clases c# dentro de GameObjects que no aparecen en la escena del juego.

Por último una de las características del desarrollo de aplicaciones en Unity es que los GameObjects desaparecen al cambiar de escena. Para tener claro lo que es una escena, estas se utilizan en diferentes videojuegos sobre todo para cargar niveles, cada nivel sería una escena y todos los objetos que se encuentran en ella, el terreno, arboles, cajas, enemigos, etc. Son GameObjects que Unity por defecto elimina al cambiar de una escena a otra y los carga una vez cargamos una escena, ya que forman parte de ella, y toda la información de los GameObjects se borra y carga con ellos a no ser que se especifique que el GameObject no debe ser eliminado de forma explícita en un script asociado a él por ejemplo.

4.2.2 Estructura de paquetes de alto nivel de la aplicación

Ahora se pasará a mostrar la estructura de paquetes de alto nivel en los que se compondrá la aplicación:

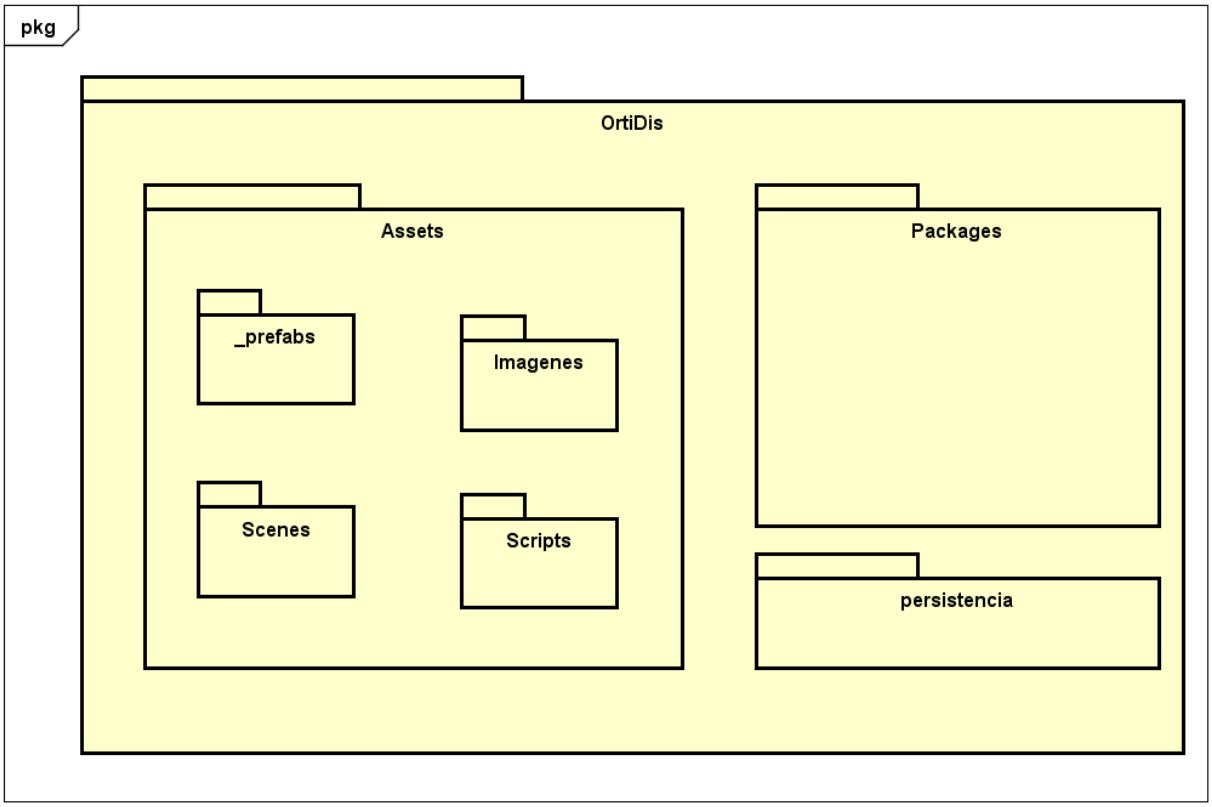

Figura 21: Diagrama de paquetes de alto nivel de la aplicación.

Del diagrama es muy importante destacar la carpeta “Packages” es una carpeta que crea Unity por defecto al crear un nuevo proyecto y la cual contiene los diferentes

ficheros que se importan y se utilizan para el buen funcionamiento de las diferentes herramientas y componentes de los que dispone Unity.

4.3 Patrones arquitectónicos

Se procederá en este apartado a explicar los patrones de diseño que se han aplicado en la implementación del proyecto:

- Patrón experto: este patrón de diseño está dentro del grupo de patrones denominado patrones GRASP[15] *(General Responsibility Assignment Software Patterns)* que pueden considerarse más bien una serie de buenas prácticas a la hora de desarrollar software, y que describen principios básicos de asignación de responsabilidades. Dentro de este grupo de patrones como se ha explicado se encuentra el patrón experto, o experto en información, que se encarga de definir las responsabilidades de una clase, en este caso el patrón experto nos dice que se debe asignar las responsabilidades de una acción a la clase que tiene la información necesaria para poder llevar a cabo esa acción.
  Un ejemplo de la estructura de este patrón es el siguiente:

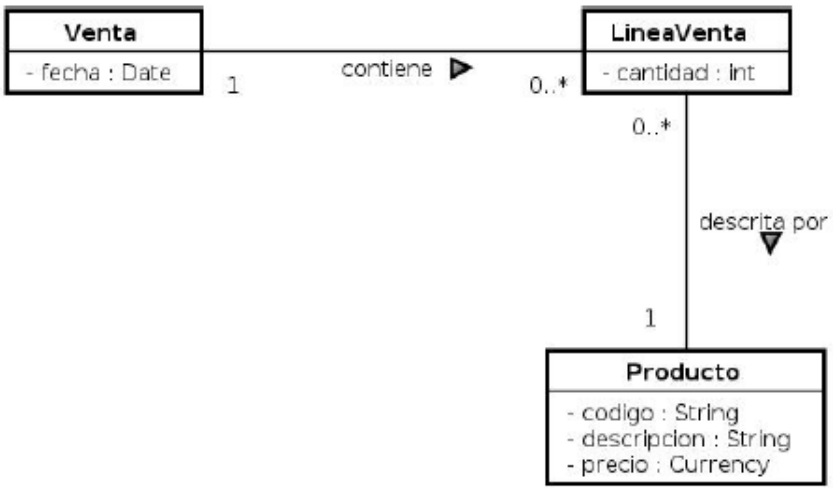

Figura 22: Ejemplo de la estructura del patrón experto.

En este ejemplo con estas tres clases particulares se puede apreciar que en el caso de que deseáramos calcular el precio total de una venta, esta operación debería realizarla la clase venta, ya que es la que tiene acceso a todas las líneas de venta que a su vez conocen los productos y su cantidad. En este proyecto en particular podemos encontrar el patrón experto en las casillas a la hora de calcular las sinergias entre las plantas ya que son las que conocen su posición dentro del plano y pueden acceder a la información de la planta que se deposite en ellas para calcular si hay una buena o una mala relación entre dos plantas colindantes. Además también se aplica este patrón a la hora de mover las

plantas, ya que debe ser cada planta concreta la que realice y calcule su propio movimiento.

- Patrón factoría[15]: este patrón es un tipo de patrón de creación, define una interfaz común que se encarga de crear objetos dejando a las subclases que elijan el tipo concreto de objeto que se va a crear. La estructura del patrón factoría es la siguiente:

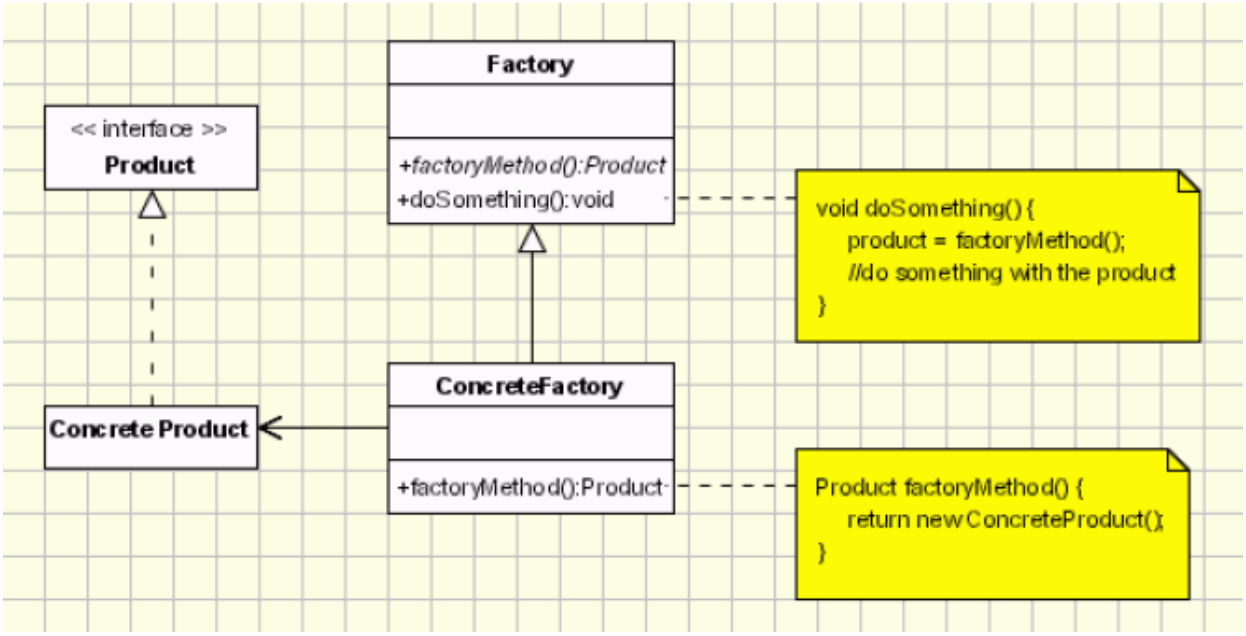

Figura 23: Imagen de ejemplo de la estructura del patrón factoría[24].

Este patrón se pone de manifiesto en este proyecto a la hora de escoger el tipo de planta que se creará, siendo la responsable de la instanciación de las plantas concretas, la clase planta de la que heredan todas las diferentes clases de plantas concretas.

- Patrón singleton[15]: se encarga de asegurarse de que una clase tenga una sola instancia y además proporciona un acceso global a ella. El patrón singleton se pone de manifiesto en Unity por defecto a la hora de generar los diferentes GameObjects ya que estos disponen de funciones para acceder a ellos mediante diferentes opciones como el propio nombre que se les haya puesto, o etiquetas que los diferencien unos de otros. La estructura de este patrón es muy simple ya que simplemente consta de una clase con un método de tipo get instance, que nos permita devolverla y acceder a ella.

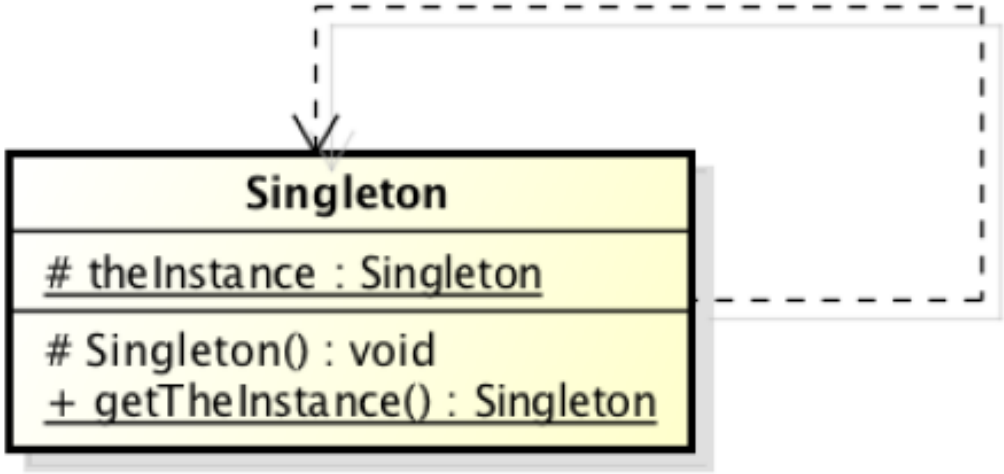

Figura 24: Ejemplo de la estructura del patrón singleton.

4.4 Arquitectura del sistema detallada

Ahora se expondrán más detalladamente los diagramas de paquetes, comenzando por el UsesStyle. Este diagrama se utiliza para mostrar las dependencias de unos módulos y otros, y nos da una idea de cómo planificar el mantenimiento y desarrollo de la aplicación:

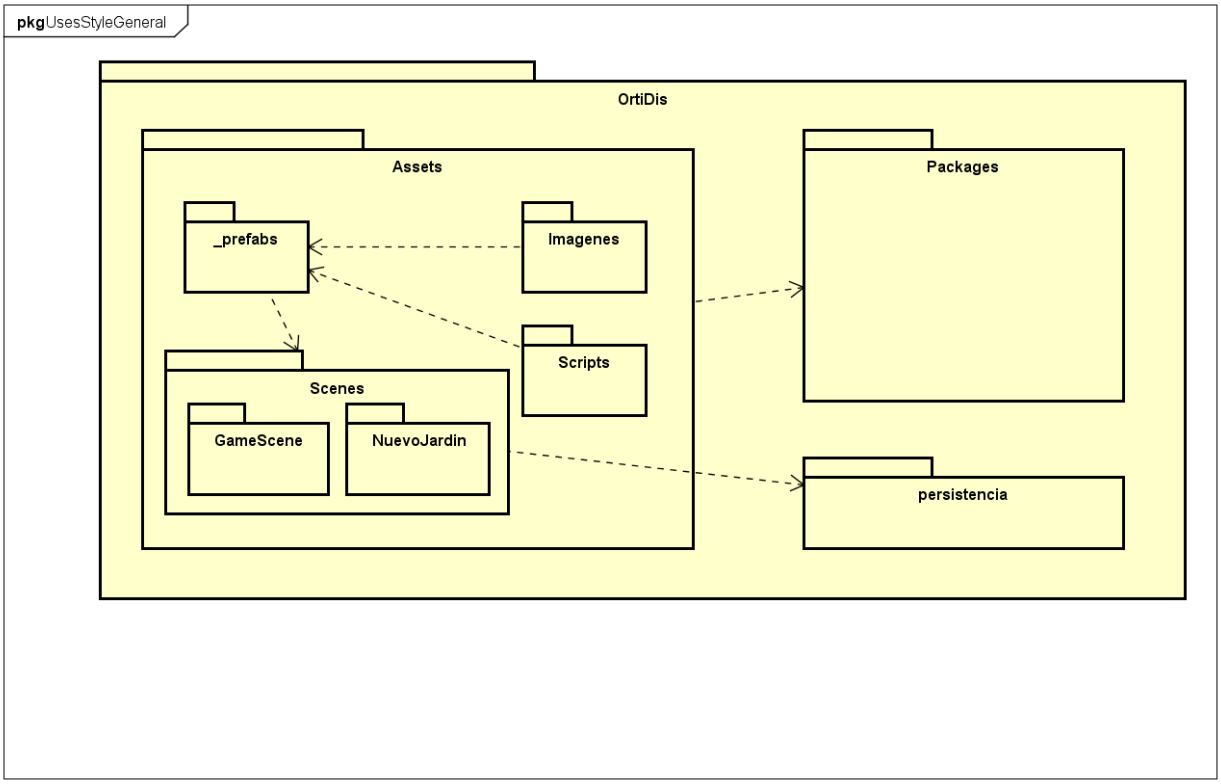

Figura 25: Uses style detallado de la aplicación.

Ahora se mostrará el diagrama de despliegue de la aplicación. Estos diagramas se utilizan para visualizar, mediante nodos, el hardware y software físico del sistema.

El diagrama de despliegue de la aplicación correspondería con el siguiente diagrama:

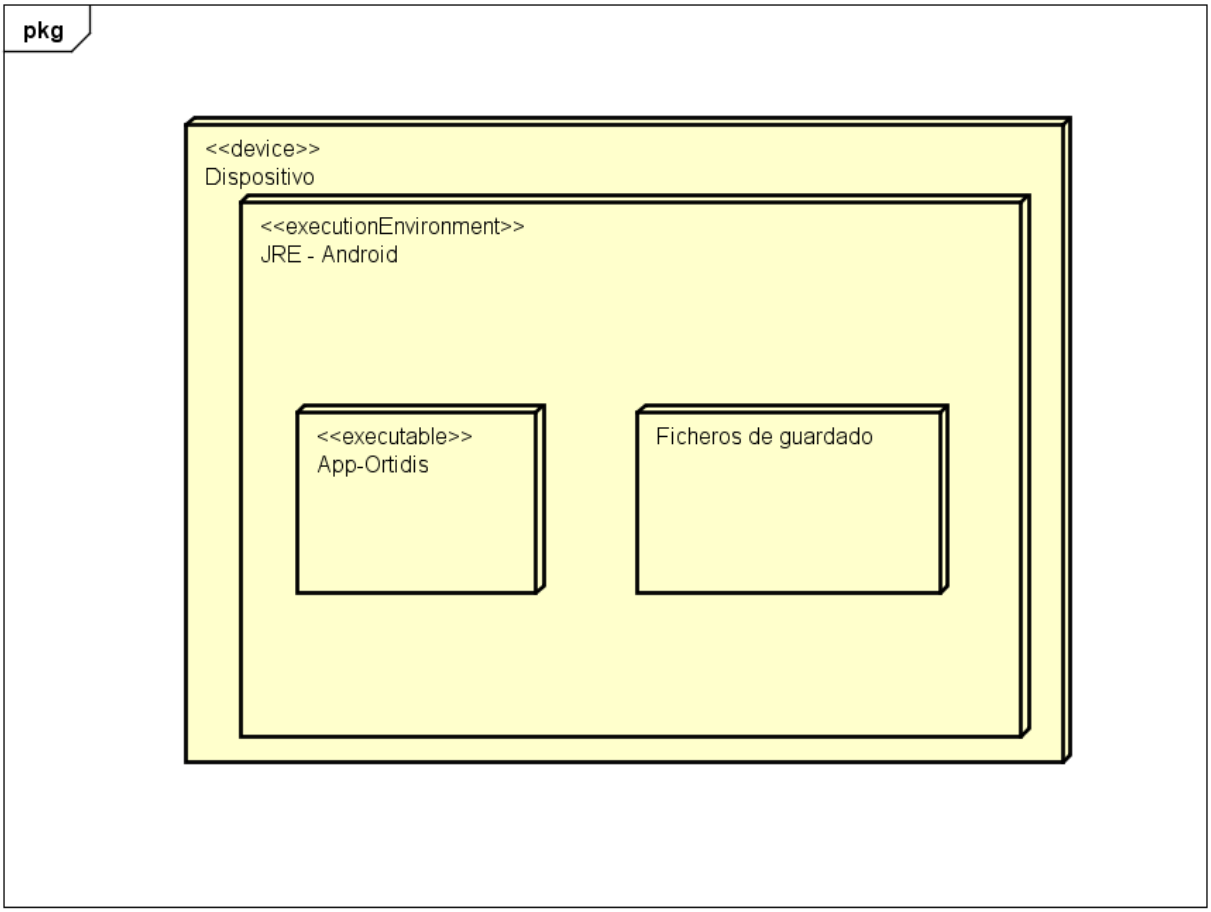

Figura 26: Diagrama de despliegue de la aplicación.

Como se puede comprobar la aplicación en este primer prototipo carece de servidor puesto que no es necesaria la comunicación con ningún elemento externo al dispositivo en el que se encontrará alojada. Además tal y como se aclaró en la fase de análisis la aplicación carece en este primer prototipo de base de datos ya que los datos que deseamos almacenar son muy simples y la acción que desempeñan puede llevarse a cabo mediante ficheros que se alojarán en el propio dispositivo también, facilitando de esta forma la implementación de la aplicación y simplificando como se puede observar el diagrama de despliegue.

4.5 Modelo de Datos

Respecto al modelo de datos utilizado, se ha llegado a la conclusión de que en vista del proyecto que se espera entregar atendiendo a los requisitos funcionales y dejando de lado los requisitos funcionales de baja prioridad, no es necesario disponer de un sistema de bases de datos complejo, ya que la única información que necesita guardar nuestro sistema es la del tamaño y nombre de los jardines, así como la información de las posiciones que ocupan nuestras plantas en el plano.

Por lo tanto para el almacenamiento de estos datos se ha optado por utilizar simples ficheros de texto que se almacenarán de forma local en el dispositivo que contenga nuestra aplicación pudiendo también de esta forma acceder a ellos más cómodamente desde la propia aplicación.

La escritura del fichero se realizará básicamente mediante un bucle que recorra las casillas del plano y obtenga el id de la planta en cada casilla, de esta forma a la hora de cargar el plano simplemente conociendo el ancho y largo del mismo podremos ir leyendo línea a línea el fichero e instanciando las plantas en las casillas correspondientes ya que el orden de lectura y escritura del fichero será el mismo.

4.6 Diseño de la interfaz de usuario

Se ha realizado un diseño de interfaz de usuario simple para permitir que cualquier persona independientemente de sus conocimientos informáticos o de horticultura, sea capaz de manejar la aplicación con soltura y de visualizar los datos de las plantas así como las marcas que indican las relaciones entre las plantas colindantes.

La aplicación consta de dos escenas, el menú inicial en el que se podrán observar hasta cinco slots que mostrarán tanto el nombre como las dimensiones de los planos guardados, que nos permitirá cargarlos pulsando sobre dichos slots, junto con un botón de borrado a la derecha de cada slot que borrará el jardín de dicho slot. Este menú también dispondrá de un botón “Salir” para finalizar la tarea de la aplicación y cerrarla, y otro para añadir nuevos jardines el cual activará un cuestionario que el usuario rellenará para posteriormente pulsar el botón de aceptar y pasar a la segunda escena correspondiente al juego.

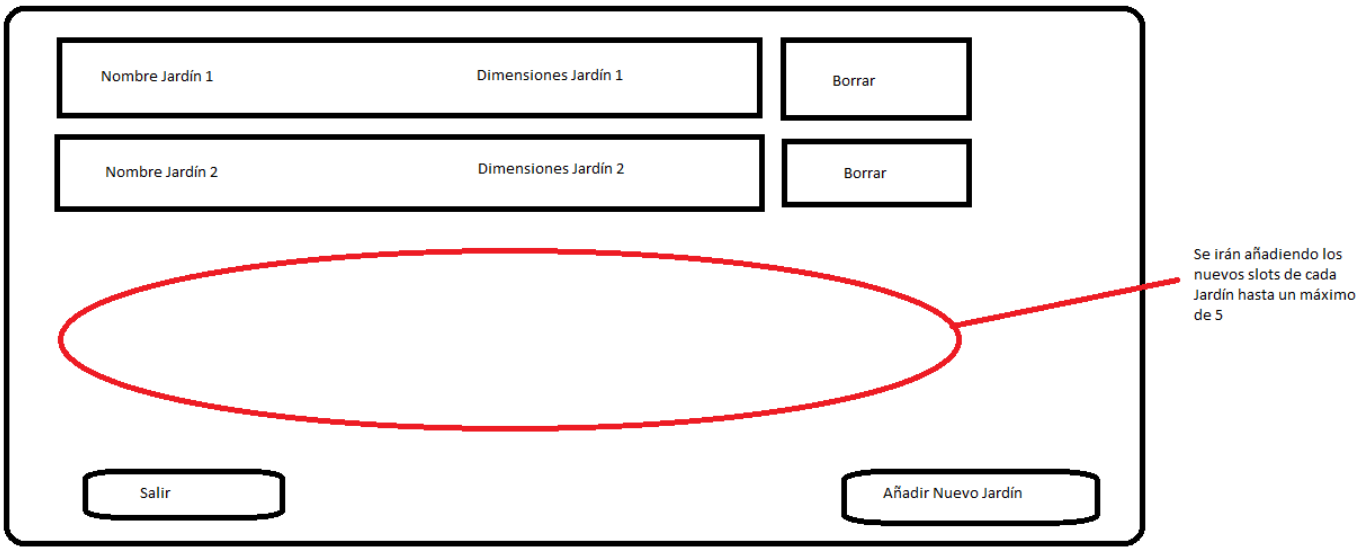

Figura 27: Vista de la primera escena de la aplicación.

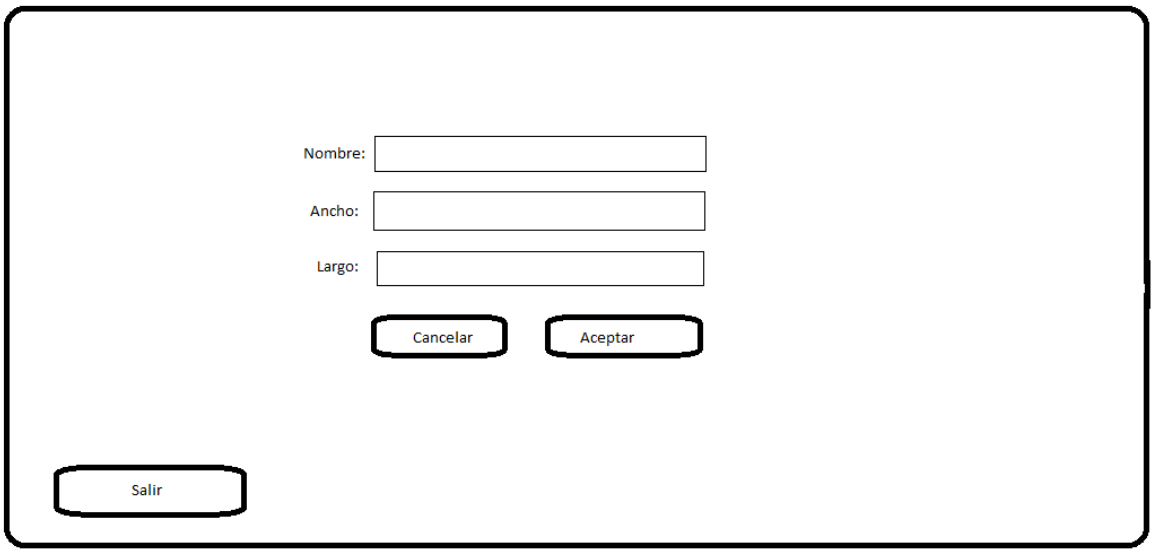

Figura 28: Vista del formulario que se activa tras pulsar el botón "Añadir nuevo Jardín".

Esta escena es más simple que la anterior puesto que sólo dispondrá del plano creado con las dimensiones que especificó el usuario, y varios botones, uno para volver al menú principal, es decir, a la escena inicial, otro botón para guardar el proyecto, un botón que activa el riego, y el botón de las plantas que activará el inventario con todas las plantas disponibles, y de cada una de ellas dispondremos a su vez de 2 botones más uno para plantarla en el plano en la primera casilla que se encuentre libre, y otro botón de información que mostrará la información de la planta correspondiente.

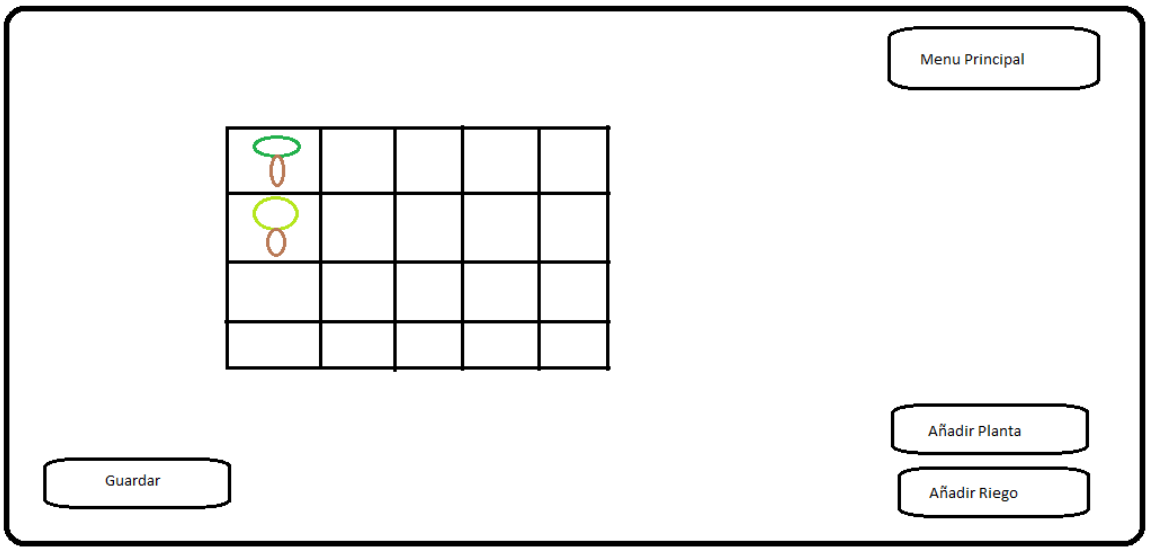

Figura 29: Vista de la segunda escena de la aplicación, correspondiente a la escena del juego.

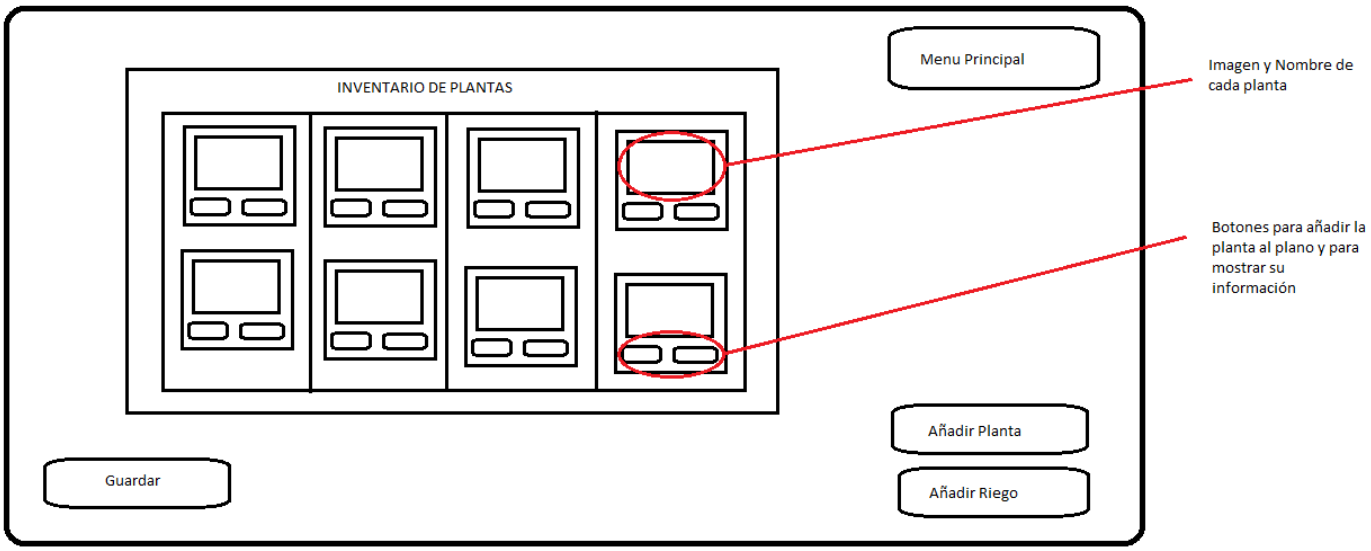

Figura 30: Vista del menú que se activa al pulsar el botón "Añadir Planta".

4.7 Clases de diseño detalladas

Ahora se procederá a mostrar las clases de diseño detalladas y se comentarán los cambios sufridos con respecto al diagrama de clases que se planteó inicialmente para la aplicación y se argumentarán los motivos por los que se han realizado dichos cambios además de comentar las características que se considere que no quedan suficientemente claras con el diagrama.

En el diagrama se muestra el total de clases programadas, es decir los scripts C# que se insertarán como componentes en los GameObjects que a menudo y por simplicidad tendrán el mismo nombre que su script.

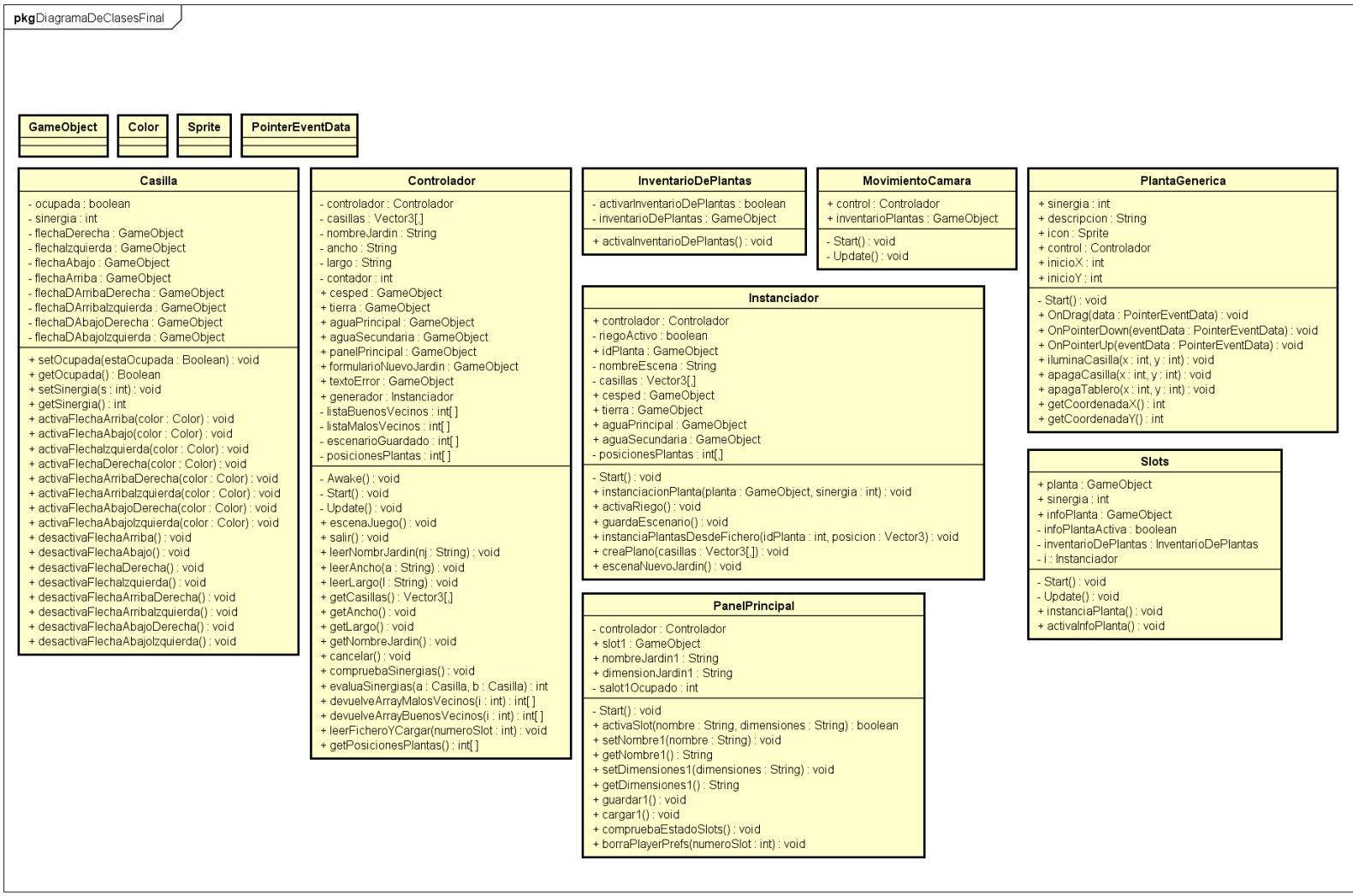

Figura 31: Clases en diseño detalladas de la aplicación.

Primero se comentarán los cambios realizados con respecto al diagrama de clases inicial. Las clase más característica es el instanciador, este se encargará de instanciar todas aquellas plantas que el usuario desee colocar en el plano y de controlar la vista del riego.

El motivo de la creación de esta clase, que irá dentro de un GameObject con su mismo nombre en la escena del juego, es la de separar todo ese trabajo del controlador y además solventar algunos problemas que aparecían en el mismo si le dábamos esta funcionalidad. Hay que recordar que el controlador es el único GameObject que no será destruido al cambiar de la escena inicial a la escena del juego, puesto que necesita almacenar información que se utilizará en el juego.

Además el instanciador deberá almacenar la referencia a todos aquellos prefabs que se desee que pueda instanciar, es decir, que aunque no se muestre en el diagrama, deberá tener un atributo idPlanta1, idPlanta2… idPlanta15… que no se añadieron en el

diagrama pues resultaba más simple explicarlo posteriormente, y cuya referencia se pasa mediante el propio Unity en su editor, no mediante código.

Pasa algo similar con la clase PanelPrincipal, que se encargará de gestionar los slots de guardado de la escena principal de la aplicación. Esta clase deberá tener un atributo para cada uno de los distintos stlots que tiene, es decir cinco, y tal y como pasaba con la clase anteriormente comentada deberá recibir la referencia a los slots mediante Unity. Es importante no confundir estos slots con la clase Slot que se encarga de la funcionalidad de los slots del inventario de plantas del juego.

La clase PlantaGenerica pasa a sustituir a la clase Planta inicialmente propuesta, y cada una de las plantas concretas pasa su funcionalidad a la clase padre. El motivo de esto es muy simple, a la hora de crear los GameObjects que se utilizarán como plantas, se les añadirá el script de planta genérica, y ya desde el editor podemos darle valor a sus atributos, por lo que es innecesario crear dos clases diferentes para posteriormente tener que añadir diferentes scripts a cada planta y terminar con un montón de clases que hacen lo mismo salvo por la diferencia de un par de atributos.

Nuestra clase controlador al ser el único GameObject que no desaparece de la escena inicial, se utilizará también como una especie de modelo que almacenará los datos correspondientes al ancho y largo del plano que se utilizarán en varias de las operaciones internas del programa. El controlador se destruye una vez volvemos al menú principal, y es sustituido por otro controlador “vacío”. Puesto que dispone de la información de las dimensiones del tablero, conoce el número de casillas y será el encargado de realizar las comprobaciones en las sinergias de las plantas para mostrárselas al usuario.

Capítulo 5: Implementación

5.1 Introducción

En este capítulo se hablará de todos los conceptos de implementación que han dado lugar al proyecto, desde las herramientas utilizadas así como la implementación del sistema de persistencia que tendrá el proyecto, el entorno de desarrollo utilizado para su implementación, algoritmos utilizados y demás características que se deben resaltar respecto al código para aclarar las dudas respecto al mismo y poder de esta forma entender mejor las decisiones que han llevado a usar los diferentes tipos de herramientas, código, etc.

5.2 Herramientas utilizadas

Se comentarán a continuación todas las herramientas utilizadas para la elaboración del proyecto comentando en algunos casos los motivos por los que se ha tomado la decisión de utilizar dichas herramientas.

5.2.1 Astah

Figura 32: Logotipo de Astah.

Astah[16] es una herramienta de modelado UML muy práctica especialmente para java pero que puede utilizarse para muchos otros lenguajes, como en el caso de este proyecto, c#.

Anteriormente su nombre era JUDE, y fue desarrollada por Kenji Hiranabe, CEO de Change Vision, Inc. Esta herramienta Ha sido utilizada para la realización de todos los diagramas que aparecen en esta memoria, desde los diagramas de clase, digramas de

secuencia, diagramas de despliegue, así como para mostrar la estructura de paquetes del sistema final.

5.2.2 Unity

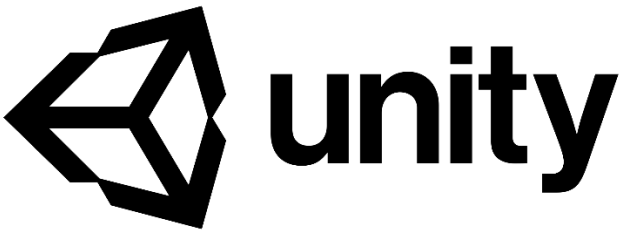

Figura 33: Logotipo de Unity.

La herramienta principal utilizada en este proyecto como entorno de desarrollo. Se hablará más extensamente de esta herramienta en el punto 5.4 de esta misma memoria cuando se pase a explicar el IDE utilizado para la realización del proyecto.

5.2.3 Visual Studio code

Figura 34: Logotipo de Visual Studio Code.

Visual Studio code[17] ha sido el IDE utilizado para la realización de este proyecto, ya que permite la programación en C# que es el lenguaje utilizado para la realización de los scripts que he han utilizado en los diferentes gameObjects creados por Unity. Al igual que Unity se hablará más extensamente sobre las funcionalidades y los motivos por los que se ha usado este IDE más adelante en el apartado 5.4 de esta memoria.

5.2.4 Microsoft Project

Figura 35: Logotipo de Microsoft Project.

Microsoft Project[18] es una herramienta de gestión de proyectos, principalmente para su uso en empresas o para personal autónomo, dispone de muchísima funcionalidad que va desde la gestión y estimación del tiempo empleado en cada una de las tereas en las que se divide un proyecto hasta el sueldo de los empleados que se encargan de la realización del proyecto, gestión de vacaciones y calendario y muchas cosas más. Si bien es cierto que esta herramienta es más útil para proyectos colaborativos por toda la funcionalidad de la que dispone, ha sido principalmente usada en este proyecto a la hora de la realización de las estimaciones de duración de las diferentes tareas que componen el proyecto.

5.2.5 One Drive

Figura 36: Logotipo de One Drive.

One drive[19] es el servicio de nube de Microsoft, no necesita mucha más explicación, permite compartir archivos entre usuarios de forma remota, y disponer de un almacenamiento remoto al que se puede acceder desde cualquier dispositivo siempre que seas el dueño de esa cuenta o tengas los permisos que el dueño pude aportar a otros usuarios. En este proyecto se ha utilizado el servicio que proporciona One Drive para realizar una copia de seguridad de la memoria del mismo así como poder compartir dicha memoria con la tutora de este TFG.

5.2.6 GitLab

Figura 37: Logotipo de GitLab.

GitLab[20] es un servicio web de control de versiones software basado en Git que es a su vez un software de control de versiones diseñado por Linus Tolvards, y que permite realizar volcados de código de programación a un repositorio remoto a medida que se va actualizando el código permitiendo de esta forma ver cómo avanza el proyecto software versión a versión, pudiendo visibilizar los cambios realizados de una versión a otra además de poder retroceder a versiones anteriores, y además, permite una participación colaborativa, por lo que es un muy buen sistema para el desarrollo de software entre varios programadores.

Su uso en este proyecto ha sido anecdótico reduciéndose a su utilización a una copia de seguridad así como una forma de poder acceder de forma remota al código de programación de la aplicación para su evaluación y revisión.

5.3 Implementación de la base de datos

Como ya explicó con anterioridad en el capítulo 3 de esta memoria, se ha decidido no implementar una base de datos debido a la simplicidad de los datos que se requiere almacenar en este proyecto, aunque se podría llegar a implementar en un futuro en el caso de que la aplicación se incrementara añadiendo más funcionalidad o más plantas y objetos para modelar el plano.

5.4 Entorno de desarrollo, IDE

El entorno de desarrollo utilizado para la implementación de este proyecto ha sido Unity junto con el IDE Visual Studio para la programación de las clases en c# ya que es el lenguaje que utiliza Unity para sus scripts.

5.4.1 Unity

Es un motor de videojuegos multiplataforma creado por Unity Technologies que actualmente está en alza y compite con otro de los motores de juego más conocidos como es Unreal Engine. Unity tiene una versión gratuita para usuarios que es la que se ha utilizado en el desarrollo del proyecto, además de tres planes diferentes pensados para empresas o autónomos, los cuales son, la versión plus, la versión pro, y el plan para empresas más grandes y que ya tienen un beneficio notable, que incluye diez puestos de Unity pro. Todos estos planes, que son naturalmente de pago, añaden funcionalidad a la versión gratuita y su licencia se debe renovada cada año.

Unity contiene como es lógico un editor que nos permite realizar una cantidad de acciones enorme, por lo que en esta memoria se tratará de explicar las características que más han ayudado y más se han utilizado en la realización del proyecto aunque primero se hablará de forma resumida de los aspectos técnicos que ofrece este motor de videojuegos.

Unity utiliza un motor gráfico OpenGL en ordenadores, tanto Windows como Linux en incluso Mac, Direct3D solamente en Windows y OpenGL ES para Android e iOS. Tiene además soporte para mapeados en relieve y sombras, aunque no ha sido necesario utilizarlas en este proyecto ya que es mucho más básico. Contiene también soporte integrado para la modelación de físicas para los diferentes objetos del entorno que deseemos crear. El scrpting se basa y realiza a través de Mono y el lenguaje utilizado como bien se ha comentado anteriormente en esta memoria es principalmente C# aunque los usuarios pueden también utilizar otros como UnityScript que es un lenguaje personalizado inspirado en la sintaxis ECMAScript, o Boo cuya sintaxis se inspira en Python. Unity provee al usuario de una versión de MonoDevelop para la edición de sus scripts, pero en el caso de este proyecto se ha decidido utilizar Visual Studio Code para este propósito ya que está más actualizado y además es posible importar las librerías que se necesitan para la programación en C# de Unity.

Unity tiene además una página a través de la cual los usuarios pueden subir y descargar assets[22], tanto modelos 2D o 3D como scripts, audio, animaciones y todo tipo de elementos y archivos compatibles con unity, aunque lamentablemente los mejores modelos y assets en general son de pago. En el caso de este proyecto solamente se han utilizado assets gratuitos limitados principalmente a sprites y texturas, aunque no se han utilizado demasiados, pero hay que destacar que en el caso de disponer y querer gastar un poco de dinero, se podría mejorar ampliamente la estética como poco de la aplicación.

Otra de las características que se ha utilizado en este proyecto de Unity es la de su editor de imágenes para usarse como sprites, aunque su uso ha sido limitado, la funcionalidad de la que dispone permite a los usuarios recortar y extraer fragmentos de imagen de una mayor así como cambiar sus colores por defecto aplicando una especie de filtros, lo que permite también extraer de una sola imagen diferentes texturas y que en el caso de este proyecto en concreto se ha utilizado para extraer los sprites del plano y el agua.

Por último la característica principal utilizada en este proyecto de Unity es su capacidad para crear lo que denominan “prefabs” y que ya se ha comentado anteriormente en la memoria, GameObjects prefabricados por el usuario que se pueden almacenar y utilizar de diferentes formas, cuya característica más representativa se podría decir que es la de permitir realizar en todos los objetos de un mismo tipo cambios a la vez, ya que al modificar el prefab guardo, todas las instancias del mismo se modificarán de igual forma. Los prefabs se han utilizado en este proyecto a la hora de crear los diferentes elementos del plano así como, los diferentes tipos de plantas y el GameObject controlador que realiza los cambios de escena y almacena los datos de tamaño y nombre del plano para los correspondientes cálculos a la hora de realizar ciertas acciones en el juego.

5.4.2 Visual Studio Code

Ha sido el editor de código utilizado para la realización de este proyecto ya que tiene soporte para Git[23], además de ayudas al programador como son autocompletados, marcas de errores, dispone también de soporta para la depuración del código y es gratuito. Ha sido desarrollado por Microsoft y puede ser utilizado tanto en Windows como en Linux y macOS. Además puede ampliarse su funcionalidad mediante complementos.

El principal motivo por el que se decidió utilizar este editor de código y no otro como puede ser MonoDevelop que es el que puede descargarse junto con Unity, es que este último está más desactualizado mientras que visual studio code presentaba más ayudas al programador como las expuestas anteriormente y además se había utilizado durante la carrera en algunas de sus asignaturas, por lo que se estaba más familiarizado con su uso.

Soporta múltiples lenguajes de programación que no merece la pena mencionar aquí ya que son demasiados y muy variados, pero hay que destacar que incluye C#, que es el lenguaje que se ha decidido utilizar en este proyecto, por lo que unido a las ventajas anteriormente mencionadas lo convertía en una de las mejores opciones para llevar a cabo la codificación de las clases de C# utilizadas en la aplicación.

5.5 Algoritmos utilizados

En este apartado se mencionarán algunos de los algoritmos utilizados en la programación de los diferentes scripts que modelan la funcionalidad del proyecto y que se han considerado de relevancia, y se comentará además el motivo por el cual se han decidido utilizar. Hay que destacar que ha habido muchas clases que se han utilizado que por el tipo de proyecto su uso no es habitual en muchos otros programas, por lo que se ha decidido tratar de explicarlas brevemente en esta memoria, pero que para usuarios acostumbrados a realizar programas en 3D o trabajar con herramientas como Unity su uso puede resultar trivial.

5.5.1 Datos básicos de los scripts en Unity

En Unity cuando añadimos un script a un GameObject este dispone de 2 métodos por defecto, el primero es el método Start. En este método se realizan la acciones que se encuentren en su interior una vez es instanciado el GameObject que contiene el script, o en el caso de que el GameObject se encuentre ya en la escena, al iniciarse esta, y este método solamente es llamado una vez.

El segundo método es el método Update, este método se ejecuta constantemente una vez por frame mientras el GameObject esté activo en la escena, por lo que en caso de que no sea necesario no debería de añadirse a los scripts ya que ralentizarían mucho la aplicación al estar ejecutando código constantemente. Este método se ha utilizado a la hora de implementar el movimiento de la cámara.

Por último otro método que se ha utilizado en el proyecto es el método Awake, este no viene por defecto a la hora de crear un uevo script en Unity. Su uso es similar al Start pero este se ejecuta justamente antes y es utilizado en este proyecto a la hora de mantener GameObjects al cambiar de escenas y que estos no se queden duplicados al realizar estos cambios.

5.5.2 Interfaz IDragHandler

Para poder utilizar dicha interfaz se debe importar la librería de UnityEngine.EventSystems. Esta interfaz al igual que el resto de interfaces que requieren el UnityEngine.EventSystems y que se comentarán a continuación en la memoria, se encargan de enviar eventos externos a los objetos de la aplicación. IDragHandler en concreto se encarga de realizar mediante su método OnDrag todos los eventos que ocurrirán en el juego en el momento en el que el usuario mantenga pulsado el objeto. Este método OnDrag recibe como parámetro un PointerEventData, que es a su vez una clase utilizada para cargar eventos en el juego asociados principalmente con respuestas de punteros, como puede ser el click del ratón en una aplicación de escritorio o un toque de una pantalla táctil en dispositivos móviles por lo que se entiende un poco mejor el cómo funciona esta función.

El método OnDrag, se ha utilizado en este proyecto para realizar el movimiento de las plantas activándose una vez la planta es pulsada y manteniendo la acción constantemente hasta que el usuario decida soltar dicha planta. Además también se realiza en este método la comprobación de la casilla sobre la que se encuentra en ese momento la planta, mostrando mediante colores si es posible colocar la planta en esa casilla o no y cambiando también el color de la planta en el caso de que se vaya a borrar si la soltamos en el plano.

5.5.3 Interfaz IPointerDownHandler

Nuevamente se debe importar la librería de UnityEngine.EventSystems para la usar esta interfaz. Mediante su método OnPointerDown se pueden realizar los eventos que el programador desea que ocurran cuando el usuario pulsa el objeto en cuestión, y su uso es similar al método OnDrag comentado anteriormente, sólo que este no se mantiene en el tiempo como el anterior. Su uso también ha estado presente en el movimiento de las plantas ya que a la hora de hacer pulsar una de ellas, esta cambia de color, y marca como vacía la casilla de la que provenía para dejar sitio a otras plantas.

5.5.4 Interfaz IPointerUpHandler

La última de las interfaces que utiliza UnityEngine.EventSystems, y que ha sido de increíble utilidad en el proyecto. Tal y como se puede deducir por el nombre de la anterior y el nombre de esta interfaz, esta se encarga de realizar los eventos que se desencadenan cuando el usuario suelta el objeto que está pulsando. En este proyecto su uso también está presente en el movimiento de las plantas, aunque esta vez su uso viene a la hora de centrar la planta en la casilla en la que se encuentra más cerca, así como marcar la casilla como ocupada una vez la planta se deposita en ella y cambiar su color nuevamente por su color original, mostrando de esta forma al usuario que ya no se encuentra moviendo la planta.

5.5.5 Class PlayerPrefs

Esta clase de Unity almacena referencias de diferentes valores del juego entre sesiones de manera que no se pierden aunque el usuario cierre el juego. PlayerPref es capaz de almacenar tanto números enteros como cadenas de texto con sus métodos SetString y SetInt respectivamente pasando como argumento una etiqueta, funcionando como una especie de id en una estructura de mapa. Para obtener la información almacenada simplemente se debe utilizar su método GetInt o GetString pasando como argumento el identificador que se ha utilizado para almacenarlo. La funcionalidad de esta clase se utiliza a la hora de almacenar el nombre y las dimensiones del plano en los slots de la escena principal, ya que necesitamos modificar dicha escena antes de que cambiemos a la escena del juego.

Las funciones de esta clase son muy recomendables de utilizar como persistencia de algunas aplicaciones siempre y cuando los datos que se deseen almacenar sean muy básicos, como podrían ser las puntuaciones más altas de un videojuego o datos similares.

5.5.6 Ray y Physics

La clase Ray y la clase Physics requieren de importar UnityEngine y sus funciones son tan amplias que en esta memoria sólo se explicará aquella función para la que se han utilizado ambas clases. La clase Ray genera un rayo en una posición hacia una

dirección. Este rayo es un vector en línea recta en el sentido en el que el usuario haya escogido, y su principal uso es el de obtener cierta información que el programador podrá utilizar, como es en el caso de este proyecto, en el cual mediante la clase Physics utilizando el método Raycast de esta, podemos obtener la información del objeto Ray para saber si en su trayectoria se ha encontrado algún tipo de objeto entre medias. Para que el objeto Ray considere que ha colisionado con un objecto, este debe tener a su vez un collider, que no es más que un componente que se puede añadir a un GameObject y que lo provee de ciertas físicas, en este caso, hace que el GameObject sea "sólido" y pueda detectar colisiones de otros objetos, o en el caso que se está explicando, pueda ser detectado por el objeto Ray.

Sabiendo todo esto, el uso que se ha dado a estas clases es el de saber si el usuario está tocando una planta del plano, las cuales disponen de collider, o está tocando el propio plano o fuera de él, ya que el plano no dispone de collider, y de esta forma activar el movimiento de la planta o de la cámara en función de esto.

5.5.7 Application.persistentDataPath

La función persistentDataPath de la clase Application devuelve, tanto en Android como IOs, la ruta al fichero de datos de la plicación, que por defecto se encuentra en la carpeta de la aplicación que se genera en el dispositivo una vez instalamos una apk. Se ha utilizado esta función para indicar la ruta de los fichero de guardado que genera la aplicación.

Además el fichero que se guarda en dicha ubicación aparecerá encriptado de manera que los datos que se almacenen en él sólo podrán ser interpretados por la aplicación.

Capítulo 6: Pruebas

6.1 Introducción

En este capítulo se mostrarán y explicarán las pruebas realizadas con la aplicación para corroborar que todo funciona correctamente y conseguir que la aplicación no tenga ningún tipo de bug o que tenga los mínimos posibles y que no dañen el sistema gravemente.

6.2 Pruebas

A la hora de probar la aplicación y al tratarse de una aplicación con interfaz gráfica diseñada para Android, se han llevado a cabo una serie de pruebas de caja negra[21] para verificar el buen funcionamiento de cada caso de uso de la aplicación final. La razón para dar prioridad a este tipo de prueba por encima de por ejemplo las pruebas de caja blanca, es que la gran parte de la funcionalidad se basa en movimientos, cambios de colores para mostrar las casillas y las plantas, activado o desactivado de menús etc. por lo que era innecesario realizar test de comprobación más allá de la mera observación de las salidas obtenidas a la realización de los diferentes casos de uso.

Las pruebas de caja negra son una técnica de pruebas para el software en las que se verifica el buen funcionamiento del programa o aplicación, sin tener en cuenta el funcionamiento interno del código y los diferentes detalles de implementación de la aplicación. Se basan en los requisitos y los casos de uso de la aplicación, por lo que a la hora de realizarse, la persona encargada de realizar las pruebas se centrará exclusivamente en la respuesta de la aplicación a las diferentes operaciones que se intenten realizar con ella.

Ahora se mostrarán cada una de las pruebas realizadas para la comprobación de los diferentes casos de uso.

6.2.1 Pruebas CU1: Crear nuevo proyecto

Test 1	**Crear un nuevo proyecto con slots de guardado libres.**
Resultado esperado.	Se cambia a la escena del juego y se crea un plano de las dimensiones escogidas por el usuario.
Resultado obtenido.	Se obtiene el resultado esperado.

Tabla 39: Test 1 crear un nuevo proyecto con slots de guardado libres.

Test 2	**Crear un nuevo proyecto sin slots de guardado libres.**
Resultado esperado.	Se muestra un mensaje de error y el caso de uso queda sin efecto.
Resultado obtenido.	Se obtiene el resultado esperado.

Tabla 40: Test 2 crear un nuevo proyecto sin slots de guardado libres.

6.2.2 Pruebas CU2: Abrir proyecto

Test 3	**Abrir proyecto existente.**
Resultado esperado.	Se cambia a la escena del juego y se carga el plano creado anteriormente con los cambios realizados por el usuario.
Resultado obtenido.	Se obtiene el resultado esperado.

Tabla 41: Test 2 abrir proyecto existente.

6.2.3 Pruebas CU3: Borrar proyecto

Test 4	**Borrar proyecto.**
Resultado esperado.	Se elimina el slot del menú principal y se elimina el fichero .txt que contenía la información del proyecto.
Resultado obtenido.	Se obtiene el resultado esperado.

Tabla 42: Test 4 borrar proyecto.

6.2.4 Pruebas CU4: Guardar proyecto

Test 5	**Guardar proyecto.**
Resultado esperado.	Se genera un fichero .txt con la información del plano que el usuario está modificando.
Resultado obtenido.	Se obtiene el resultado esperado.

Tabla 43: Test 5 guardar proyecto.

6.2.5 Pruebas CU5: Elegir dimensiones del plano

Test 6	**Elegir dimensiones del plano correctas**
Resultado esperado.	No se muestra ningún tipo de mensaje al usuario, si escoge generar el plano se genera correctamente.
Resultado obtenido.	Se obtiene el resultado esperado.

Tabla 44: Test 6 elegir dimensiones del plano correctas

Test 7	**Elegir dimensiones del plano incorrectas con valor por encima del rango establecido**
Resultado esperado.	Se muestra un mensaje de error por pantalla.
Resultado obtenido.	Se obtiene el resultado esperado.

Tabla 45: Test 7 elegir dimensiones del plano incorrectas con valor por encima del rango establecido.

Test 8	**Elegir dimensiones del plano incorrectas con valor diferente de un número entero.**
Resultado esperado.	Se muestra un mensaje de error por pantalla.
Resultado obtenido.	Se obtiene el resultado esperado.

Tabla 46: Test 8 elegir dimensiones del plano incorrectas con valor diferente de número entero.

Test 9	**Elegir dimensiones del plano incorrectas sin ningún valor.**
Resultado esperado.	Se muestra un mensaje de error por pantalla.
Resultado obtenido.	Se obtiene el resultado esperado.

Tabla 47: Test 9 elegir dimensiones del plano incorrectas sin ningún valor.

6.2.6 Pruebas CU6: Mover cámara

Test 10	**Mover la cámara en el eje X**
Resultado esperado.	La cámara se moverá en el sentido inverso al que el usuario arrastre el dedo.
Resultado obtenido.	Se obtiene el resultado esperado.

Tabla 48: Test 10 mover la cámara en el eje x.

Test 11	**Mover la cámara en el eje Y**
Resultado esperado.	La cámara se moverá en el sentido inverso al que el usuario arrastre el dedo.
Resultado obtenido.	Se obtiene el resultado esperado.

Tabla 49: Test 11 mover la cámara en el eje y.

6.2.7 Pruebas CU7: Añadir planta/flor

Test 12	**Añadir planta seleccionada con casillas libres.**
Resultado esperado.	Se instancia la planta seleccionada en la primera casilla del plano que se encuentre libre.
Resultado obtenido.	Se obtiene el resultado esperado.

Tabla 50: Test 12 añadir planta seleccionada con casillas libres.

Test 13	**Añadir planta seleccionada sin casillas libres.**
Resultado esperado.	No instancia la planta y el caso de uso queda sin efecto.
Resultado obtenido.	Se obtiene el resultado esperado.

Tabla 51: Test 13 añadir planta seleccionada sin casillas libres.

En el caso particular de este caso de uso se realizó una comprobación con cada planta de las disponibles en el inventario de plantas.

6.2.8 Pruebas CU8: Mover planta/flor

Test 14	**Mover planta/flor a una casilla libre.**
Resultado esperado.	La planta se centra en la casilla a la que haya sido movida dejando la anterior casilla libre.
Resultado obtenido.	Se obtiene el resultado esperado.

Tabla 52: Test 14 mover planta/flor a una casilla libre.

Test 15	**Mover planta/flor a una casilla ocupada.**
Resultado esperado.	La planta vuelve a su casilla original.
Resultado obtenido.	Se obtiene el resultado esperado.

Tabla 53: Test 15 mover planta/flor a una casilla ocupada.

6.2.9 Pruebas CU9: Borrar planta/flor

Test 16	**Borrar planta/flor.**
Resultado esperado.	La planta/flor es eliminada de la escena del juego.
Resultado obtenido.	Se obtiene el resultado esperado.

Tabla 54: Test 16 borrar planta/flor.

6.2.10 Pruebas CU10: Añadir sistema de riego

Test 17	**Añadir sistema de riego.**
Resultado esperado.	Se activa el sistema de riego óptimo para el plano haciéndose visible para el usuario.
Resultado obtenido.	Se obtiene el resultado esperado.

Tabla 55: Test 17 añadir sistema de riego.

El sistema de riego óptimo según el proyecto de fin de carrera de Diego Rivero Hernández y David Sanz Villalón “Simulación y optimización vía web de topologías de

riego en entornos de cultivo de Castilla y León"[26], siempre será el de crear un bancal principal, por donde se llevará a cabo el riego, en el lado más largo de nuestro plano, dejando los surcos secundarios perpendiculares a este principal.

6.2.11 Pruebas CU11: Borrar sistema de riego

Test 18	**Borrar sistema de riego.**
Resultado esperado.	Se desactiva el sistema de riego haciéndose invisible para el usuario.
Resultado obtenido.	Se obtiene el resultado esperado.

Tabla 56: Test 18 borrar sistema de riego.

6.2.12 Pruebas CU12: Mostrar sinergias entre plantas

Test 19	**Mostrar sinergias entre plantas beneficiosas.**
Resultado esperado.	Se muestra un enlace de color azul entre las 2 plantas cuya sinergia es buena.
Resultado obtenido.	Se obtiene el resultado esperado.

Tabla 57: Test 19 mostrar sinergias entre plantas beneficiosas.

Test 20	**Mostrar sinergias entre plantas perjudiciales.**
Resultado esperado.	Se muestra un enlace de color rojo entre las 2 plantas cuya sinergia es buena.
Resultado obtenido.	Se obtiene el resultado esperado.

Tabla 58: Test 20 mostrar sinergias entre plantas perjudiciales.

Test 21	**Mostrar sinergias entre plantas neutras.**
Resultado esperado.	No se muestra ningún tipo de enlace entre dos plantas colindantes que no tiene relación ni beneficiosa ni perjudicial entre ellas.
Resultado obtenido.	Se obtiene el resultado esperado.

Tabla 59: Test 21 mostrar sinergias entre plantas neutras.

6.2.13 Pruebas CU13: Mostrar información de la planta/flor

Test 22	**Mostrar la información de la planta/flor seleccionada.**
Resultado esperado.	Se muestra la información de la planta seleccionada.
Resultado obtenido.	Se obtiene el resultado esperado.

Tabla 60: Test 22 mostrar la información de la planta/flor seleccionada.

Las pruebas realizadas para este caso de uso se han realizado para cada una de las plantas disponibles en la aplicación.

6.3 Pruebas con usuarios

Se han realizado una serie de pruebas de la aplicación con usuarios de diversas edades y sexo, que han utilizado la aplicación para realizar diferentes casos de uso y comprobar principalmente si los usuarios son capaces de utilizar la aplicación correctamente sin grandes dificultades.

- Los casos de uso que se ha pedido realizar a los usuarios son:
- Crear un nuevo proyecto.
- Añadir diferentes tipos de plantas.
- Mover las plantas por el plano.
- Borrar una planta.
- Guardar el proyecto, volver al menú y cargar el proyecto guardado.

6.3.1 Opiniones de los usuarios

Los usuarios consultados opinan que la aplicación es simple de utilizar y han aprendido rápido su manejo.

Ningún usuario tuvo problemas a la hora de realizar la secuencia de actividades propuesta en un tiempo corto, aunque varios encontraron algunas dificultades a la hora de borrar una planta existente en el plano.

6.3.2 Conclusiones

La usabilidad de la aplicación es simple y ninguno de los usuarios tuvo problemas con la interfaz. El caso de uso “Borrar planta” puede llegar a ser ambiguo ya que no dispone de ningún botón y podrían estudiarse formas diferentes para realizar este caso de uso.

En conclusión final, se deduce con que el trabajo cumple con los objetivos propuestos inicialmente.

Capítulo 7: Concusiones y futuras aplicaciones

7.1 Introducción

Capítulo final de esta memoria en el que se explicarán los objetivos que se han cumplido con respecto a la idea original y el enunciado propuesto. Se argumentarán también los cambios más grandes realizados con respecto a la propuesta inicial y se presentarán ideas de cara a la ampliación del proyecto así como la mejora de algunas de las áreas que ya se han cubierto.

7.2 Objetivos Cumplidos

En el desarrollo de la aplicación se ha intentado cumplir con todos los objetivos planteados inicialmente en el enunciado propuesto para el proyecto aunque naturalmente algunos de ellos no han sido posibles debido a varios motivos que se comentarán en el siguiente apartado de la memoria cuando se hable del trabajo futuro de la aplicación y los posibles cambios y añadidos que mejorarían la aplicación.

Ahora se comentarán todos los objetivos iniciales que sí que se han visto satisfechos en este primer prototipo de la aplicación con respecto a los que se plantearon inicialmente.

El usuario puede con este primer prototipo de la aplicación generar un plano rectangular con las dimensiones que él desee hasta un máximo de 10 casillas de ancho y 10 de largo, pudiendo además guardar hasta 5 de los planos que haya creado, pudiendo visualizarlos y modificarlos nuevamente siempre que lo desee y guardar los cambios realizados siempre que lo desee.

El usuario también es capaz de visualizar la información de todas las plantas añadidas en el prototipo inicial desde el menú de las pantas pulsando el botón de información de las mismas de forma simple y viendo la información más relevante que necesitará a la hora de distribuir sus plantas por el huerto.

También el usuario será capaz de visualizar en tiempo real mediante marcas de colores los diferentes tipos de relaciones tanto beneficiosas como perjudiciales de las plantas que se encuentran en casillas colindantes.

Por último el usuario también podrá visualizar el diseño de bancales de riego más óptimo para el plano que ha generado así como desactivar dicho riego pulsando un solo botón del plano.

Además la aplicación es capaz de reescalarse para distintos tipos de resolución de pantalla aunque es cierto que algunos textos en ciertas pantallas pueden verse muy pequeños y mal.

7.3 Objetivos no cumplidos

El único de los grandes objetivos que no se ha llegado a cumplir ha sido el de realizar la aplicación en una perspectiva 2,5D. Esto ha sido principalmente debido a que los modelos en 3D de los diferentes tipos de plantas han sido imposibles de obtener, bien por falta de existencia de los mismos en las diferentes assets stores de Unity, o bien porque estos modelos en muchas ocasiones eran de pago. Debido a esto la única forma de cumplir con este objetivo hubiera sido la realización de los modelos en 3D por parte del programador, y esto hubiera supuesto un coste enorme tanto de tiempo para llevarlo a cabo como en el aprendizaje para poder realizar dichos modelos con una calidad decente. Debido a esto se decidió cambiar a un modelo en 2D con cámara orbital que se ajustaba mejor con los modelos en 2D que podían obtenerse mediante imágenes las cuales a su vez podían ser obtenidas fácilmente utilizando cualquier buscador en internet.

7.4 **Trabajo futuro**

Teniendo en cuenta el tipo de aplicación que se ha mostrado en este TFG se puede apreciar que la funcionalidad de la misma es capaz de aumentar ampliamente junto con la calidad en el diseño, puesto que este último aspecto no se ha tenido muy en cuenta a la hora de programar este primer prototipo de la aplicación. Por ello a continuación se expondrán algunas de las características que bien no se han podido cumplir o bien se pueden mejorar o añadir para hacer que la aplicación sea más útil, y se expondrán los diferentes motivos por los que estos cambios pueden ser beneficiosos de cara a una aplicación final más completa que pueda llegar incluso a comercializarse.

7.4.1 Resolver los objetivos no cumplidos

Las primeras mejoras que se podrían realizar al proyecto de cara a una versión futura mejorada, serían las expuestas en el apartado de los objetivos no cumplidos expuesta con anterioridad.

Afortunadamente el único de los objetivos que no se cumplió fue el de realizar la aplicación con una vista del plano en 2,5D, por lo que lo único que habría que mejorar en este aspecto sería el de conseguir los modelos en 3D de las plantas con la calidad suficiente, y añadir a la cámara orbital de la escena del plano la capacidad para moverse en tres dimensiones en vez de solamente las dos actuales, y sería interesante también añadir la rotación a la misma ya que en un entorno 2,5D podría llegar a ser de utilidad.

7.4.2 Añadidos y modificaciones

La principal modificación que podría llevarse a cabo con los conocimientos y herramientas necesarias, sería la comentada en el apartado anterior, es decir, pasar a un plano 2,5D, para ello se necesitarías como se comentó antes los modelos en 3D de las diferentes plantas, y una vez tuviéramos estos modelos habría que permitir a la cámara rotar no solamente moverse en las dos dimensiones en las que es capaz de moverse en este primer prototipo, además de añadir el movimiento de profundidad de la cámara que permitiera al usuario acercarse y alejarse del plano.

Otra modificación interesante relacionada con la anteriormente comentada, sería la de cambiar estéticamente el diseño del juego, mediante mejores fondos, sprites más detallados, botones más llamativos e incluso que las marcas que muestran las relaciones entre las plantas fuera más llamativa y que pegara más con el aspecto del huerto.

Con respecto a la persistencia de los datos de la aplicación, también sería interesante realizar un cambio en el caso de que la complejidad de la aplicación creciera mucho y se utilizaran muchos más elementos, cambiando el sistema de ficheros actual por algún tipo de almacenamiento en la nube que permitiera a los usuarios realizar muchos más guardados, aunque también para ello debería de crearse un servidor e implementar un sistema de cuentas para que los usuarios se identifiquen y puedan acceder a sus datos de forma remota.

En cuanto a los añadidos que se han pensado estaría como el lógico el añadido de muchas más plantas y flores, junto con un buscador en el inventario de plantas en el caso de que el número de plantas aumente demasiado para que los usuarios puedan acceder a cada planta de forma simple y rápida en lugar de tener que buscarla en todo el inventario.

Se podría también añadir diferentes vínculos que permitan a los usuarios conocer o incluso comprar por internet las semillas y las plantas que deseen cultivar como se vio en una de las aplicaciones similares estudiadas en el capítulo uno de esta misma memoria.

Por último unos de los cambios estéticos que se podría llegar a añadir es el de poder cambiar el tipo de plano, para permitir al usuario elegir el tipo de terreno que más

le guste, y poder además ver cambios de luz en el plano, simulando días y noches y añadiendo de esta forma iluminación en caso de que sea necesario.

Bibliografía

[1] CARRERAS REGORIGO, ÁLVARO. Route66App. Trabajo de fin de grado. Grado en Ingeniería informática (mención ingeniería del software). 2018.

[2] ESCRIBANO GÓMEZ, CRISTINA. Desarrollo de aplicación para la generación de juegos geográficos a medida. Trabajo de fin de grado. Grado en Ingeniería informática (mención ingeniería del software). 2019.

[3] APARICIO DOMÍNGUEZ, JUAN CARLOS. GuiaMeUVa: Desarrollo de una aplicación web que localiza y guíe a lugares en la Escuela de ingeniería informática. Trabajo de fin de grado. Grado en Ingeniería informática (mención tecnologías de la información). 2019.

[4] LÓPEZ GARCINUÑO, CARLOS. Estudio y aplicación de un juego en red multijugador para dispositivos móviles. Trabajo de fin de grado. Grado en Ingeniería informática (mención ingeniería del software). 2019.

[8] PROJECT MANAGEMENT INSTITUTE. Guía de los fundamentos para la dirección de proyectos: (Guía del PMBOK). Sexta edición. 2017. Editorial Project management institute.

[9] HUGHES, BOB AND COTTERELL, MIKE. Software Project Management. Quinta edición. 2009. Editor London: McGraw-Hill.

[10] LAGUNA, MIGUEL ÁNGEL. Departamento de Informática de la Uva. Apuntes de la asignatura Modelado de software. Curso 2017 -2018.

[15] CRESPO GONZÁLEZ CARVAJAL, YANIA. Departamento de Informática de la Uva. Apuntes de la asignatura Diseño de software. Curso 2017 -2018.

[26] SANZ VILLALÓN, DAVID Y RIVERO HERNÁNDEZ DIEGO. Simulación y optimización vía web de topologías de riego en entornos de cultivo de Castilla y León. Trabajo de fin de carrera. Ingeniería Técnica de Telecomunicación. 2004.

Webgrafía

[5] ECURED. Proceso unificado de desarrollo. Fecha de última consulta 13 de Julio de 2021. https://www.ecured.cu/Proceso_unificado_de_desarrollo

[6] HERNÁNDEZ, HÉCTOR. Los buenos vecinos, malos vecinos en el huerto. Fecha de última consulta 13 de Julio de 2021. http://www.tuinen.es/el-huerto-en-el-jardin/los-buenos-vecinos-malos-vecinos-en-el-huerto

[7] LOS PEÑOTES. Las buenas relaciones en el huerto. Fecha de última consulta 13 de Julio de 2021. https://store.lospenotes.com/llevarse-bien-es-cuestion-de/

[11] UNITY TECHNOLOGIES. La plataforma líder para crear contenido interactivo en tiempo real. Fecha de última consulta 13 de Julio de 2021. https://unity.com/es

[12] WIKIPEDIA. Unity (motor de videojuego). Fecha de última consulta 13 de Julio de 2021. https://es.wikipedia.org/wiki/Unity_(motor_de_videojuego)#Caracter%C3%ADsticas_principales

[13] JOBTED. Jobted. Sueldo del Ingeniero Informático en España. Fecha de última consulta 13 de Julio de 2021. https://www.jobted.es/salario/ingeniero-inform%C3%A1tico

[14] MICROSOFT. Paseo por el lenguaje C#. Fecha de última consulta 13 de Julio de 2021. https://docs.microsoft.com/es-es/dotnet/csharp/tour-of-csharp/

[16] WIKIPEDIA. Astah. Fecha de última consulta 13 de Julio de 2021. https://en.wikipedia.org/wiki/Astah*

[17] WIKIPEDIA. Visual Studio Code. Fecha de última consulta 13 de Julio de 2021. https://es.wikipedia.org/wiki/Visual_Studio_Code

[18] OSWALDO LUGO, SERGIO. ¿Qué es Microsoft Project y para qué sirve? Fecha de última consulta 13 de Julio de 2021. https://www.alpha-consultoria.com/que-es-microsoft-project-y-para-que-sirve/

[19] MICROSOFT. Almacenamiento personal en la nube de OneDrive. Fecha de última consulta 13 de Julio de 2021. https://www.microsoft.com/es-es/microsoft-365/onedrive/online-cloud-storage

[20] WIKIPEDIA. GitLab. Fecha de última consulta 13 de Julio de 2021. https://es.wikipedia.org/wiki/GitLab

[21] PMOINFORMATICA.COM. Pruebas de caja negra: Ejemplos. Fecha de última consulta 13 de Julio de 2021. http://www.pmoinformatica.com/2017/02/pruebas-de-caja-negra-ejemplos.html

[22] UNITY DOCUMENTATION. Flujo de trabajo de los Assets (Asset Workflow). Fecha de última consulta 13 de Julio de 2021. https://docs.unity3d.com/es/530/Manual/AssetWorkflow.html

[23] WIKIPEDIA. Git. Fecha de última consulta 13 de Julio de 2021. https://es.wikipedia.org/wiki/Git

[24] OODESIGN.COM. Factory Method Pattern. Fecha de última consulta 13 de Julio de 2021. https://www.oodesign.com/factory-method-pattern.html

[25] UNIVERSIDAD DE VALLADOLID. PARQUE CIENTÍFICO. Oferta de oficinas llave en mano, ubicados en el Parque científico Uva dentro del Campus Universitario Miguel Delibes (Valladolid). Fecha de última consulta 13 de Julio de 2021. http://www.parquecientificouva.es/Upload/ESPACIOS/CTTA/CTTA_FichaOficinas_Jun2013.pdf

Anexos

Manual de Instalación de la aplicación

A continuación se mostrará paso a paso cómo instalar la aplicación en dispositivos móviles, el proyecto completo se puede consultar de forma remota en: https://github.com/adrvegaac/OrtiDis-TFG.

Paso 1

Lo primero que deberemos hacer es descargar el archivo apk en el dispositivo Android donde deseemos ejecutar la aplicación. Ya que la aplicación no se encuentra en google store deberemos transferirla de forma externa, podemos hacer esto conectando el dispositivo directamente a un pc y transfiriéndola desde ahí si lo deseamos.

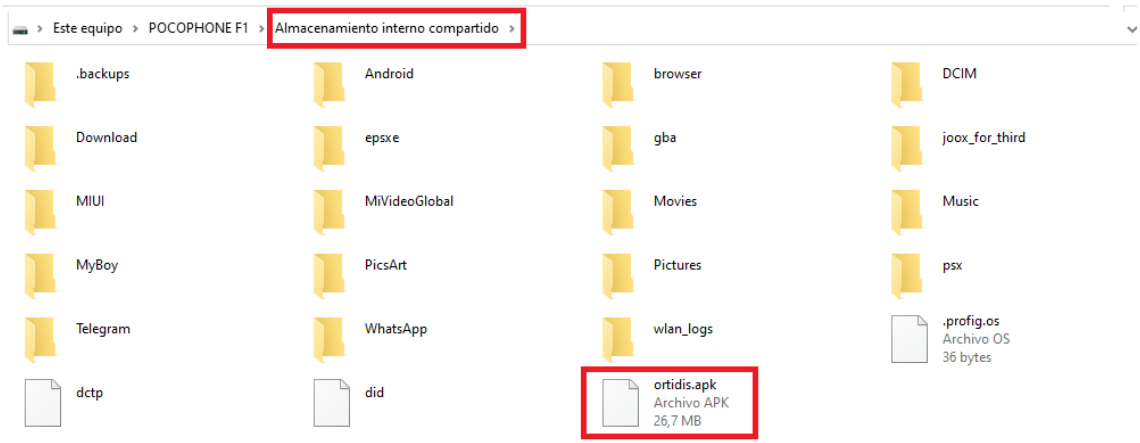

Figura 38: Captura de pantalla de referencia para el paso 1 de la instalación de la aplicación.

Paso 2

Una vez hecho esto abrimos la aplicación de la que disponga nuestro dispositivo de gestor de archivos y buscamos en apks, si esto no aparece aquí podemos buscar la apk desde ajustes y aplicaciones. Una vez localizada la apk pulsamos sobre ella para proceder a su instalación.

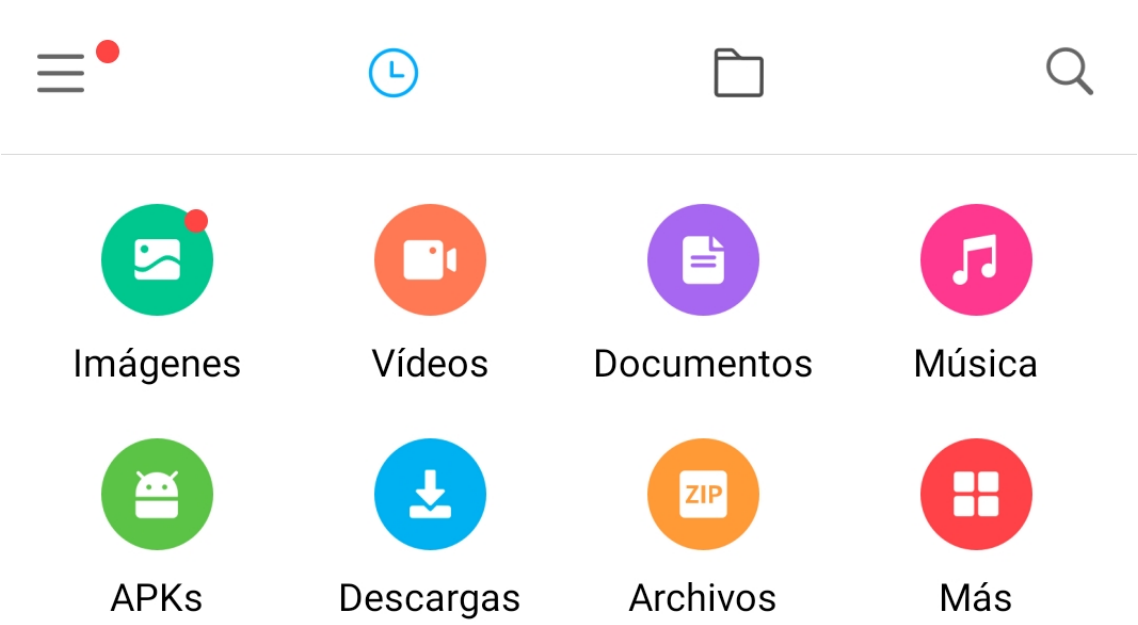

Figura 39: Captura de pantalla número 1 de referencia para el paso 2 de la instalación de la aplicación.

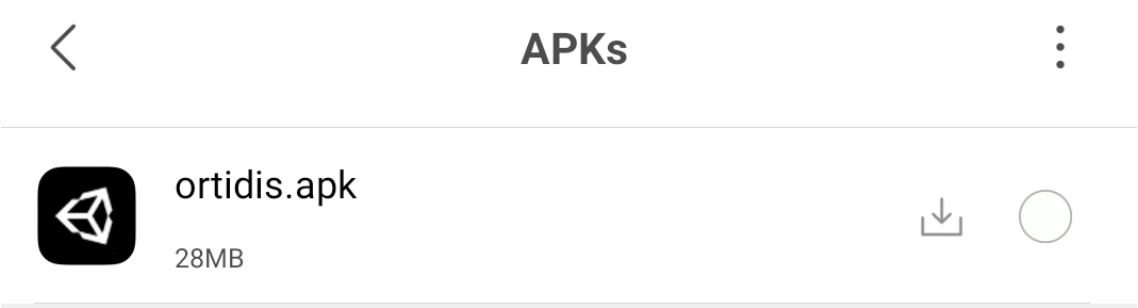

Figura 40: Captura de pantalla número 2 de referencia para el paso 2 de la instalación de la aplicación.

Paso 3

El último paso es instalar la apk, debido a que proviene de una fuente externa a la google store nos aparecerán una serie de avisos y nos pedirá que aceptemos permisos para instalar la aplicación bajo nuestra responsabilidad. Es posible que debamos dar permisos para instalar aplicaciones externas también desde los ajustes de nuestro dispositivo. Además nos aparecerá un mensaje preguntándonos si deseamos enviar la aplicación para analizarla, no debería dar ningún problema si lo hacemos pero se recomienda no enviar.

Deberían aparecernos pantallas similares a estas:

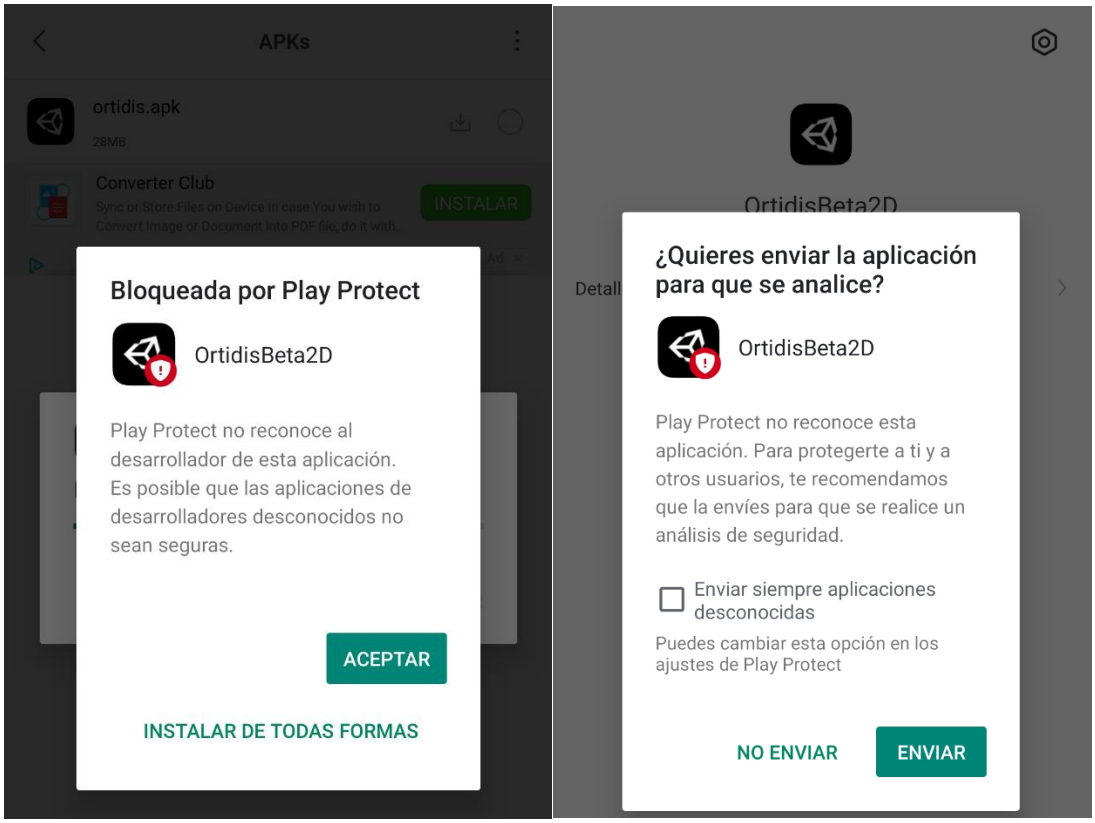

Figura 41: Captura de pantalla de referencia para el paso 3 de la instalación de la aplicación.

Una vez realizados estos pasos la aplicación se instalará y podrá ejecutarse como cualquier otra aplicación de Android.

Manual de Usuario

Se procederá a mostrar el manual de usuario para el uso de la aplicación presentada paso a paso, aunque no necesariamente tiene que seguirse un orden establecido.

Primero se puede observar la primera escena, en un primer momento se encuentra vacía, a medida que el usuario decida guardar los diferentes planos se irá llenando de slots con los diferentes guardados realizados por el usuario.

Figura 42: Captura de pantalla de la escena del menú principal de la aplicación OrtiDis.

Al pulsar el botón de “Crear nuevo Jardín +” se abrirá el menú para diseñar las dimensiones del plano, por supuesto el botón “Salir” cierra la aplicación.

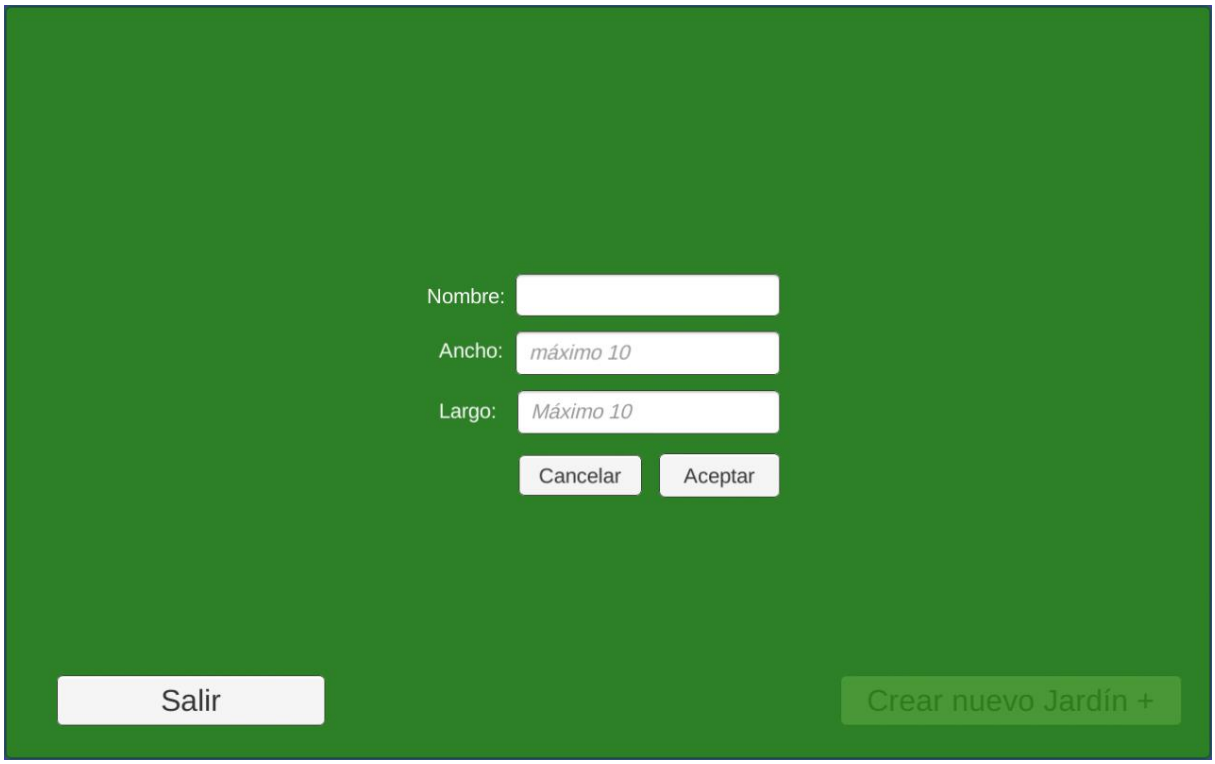

Figura 43: Captura de pantalla del menú de creación de nuevo jardín de la aplicación OrtiDis.

Una vez rellenados los campos con las medidas que se deseen y se pulse el botón aceptar se pasará a la escena del juego y el usuario se encontrará ante una cuadrícula del tamaño que diseñó completamente vacía y una serie de botones, el Botón “Guardar” guarda los progresos del plano, el botón “Menú Principal” también guarda los progresos y devuelve al usuario al menú principal, el botón “Guardar” está para prevenirse de cerrar la aplicación de forma incorrecta, ya que la forma correcta sería cerrar la aplicación desde el menú principal con el botón “Salir”.

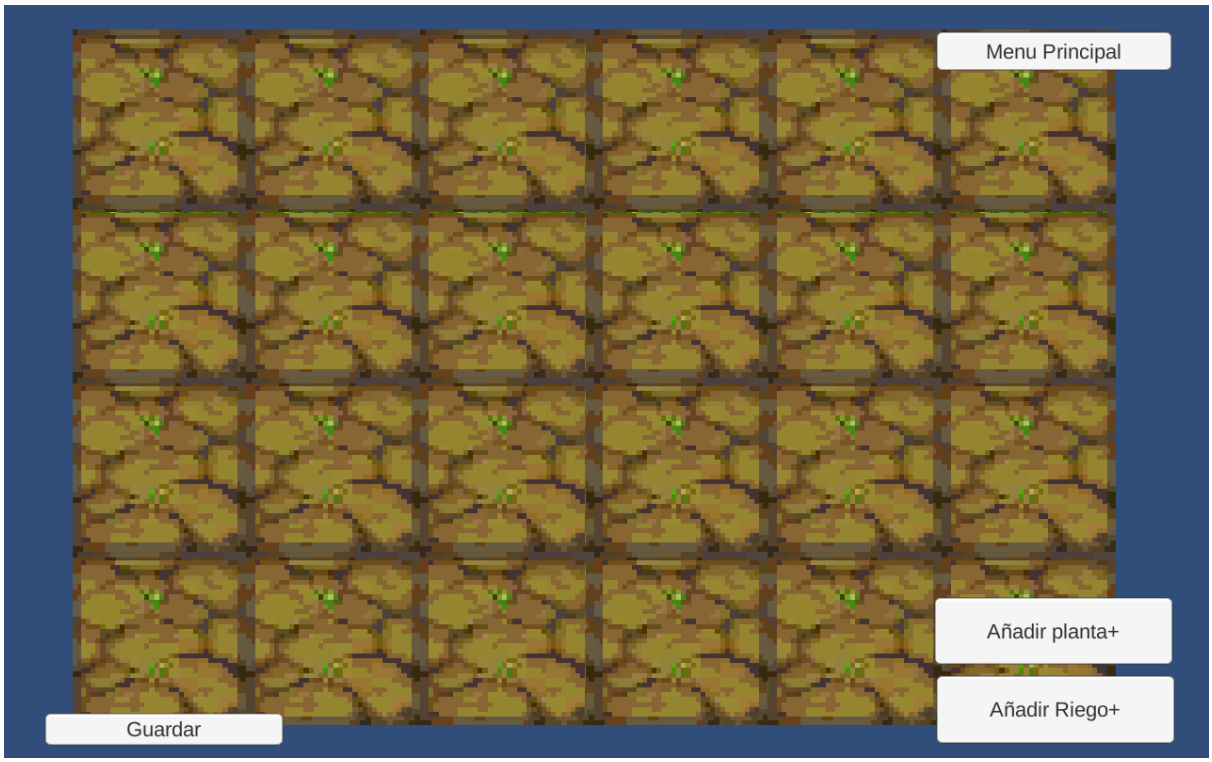

Figura 44: Captura de pantalla de la escena del juego de un plano nuevo de la aplicación OrtiDis.

Al pulsar el botón “Añadir planta+” se abrirá el inventario de plantas, para cerrarlo el usuario deberá pulsar de nuevo el botón “Añadir planta+” o bien pulsar uno de los botones de alguna planta, bien el botón “Plantar” o bien el botón “info”.

Figura 45: Captura de pantalla del inventario de plantas dentro de la escena del juego de la aplicación OrtiDis.

Si el usuario pulsa el botón “Info” del tomate por ejemplo se desplegará la información de esta planta, lo mismo sucede con el resto. Se puede deslizar el inventario hacia abajo para ver el resto de plantas disponibles.

Figura 46: Captura de pantalla de la información del tomate dentro de la escena del juego de la aplicación OrtiDis.

En el caso de que el usuario pulse en “Plantar” en alguna de las plantas el menú se cerrará y aparecerá esa planta seleccionada en el plano, las plantas que se encuentren en casillas colindantes muestran su sinergia, en caso de que sea buena aparecerá una conexión entre ambas plantas de color azul, si es mala la conexión aparecerá de color rojo, en caso de que sea neutra no se mostrará conexión alguna entre las plantas.

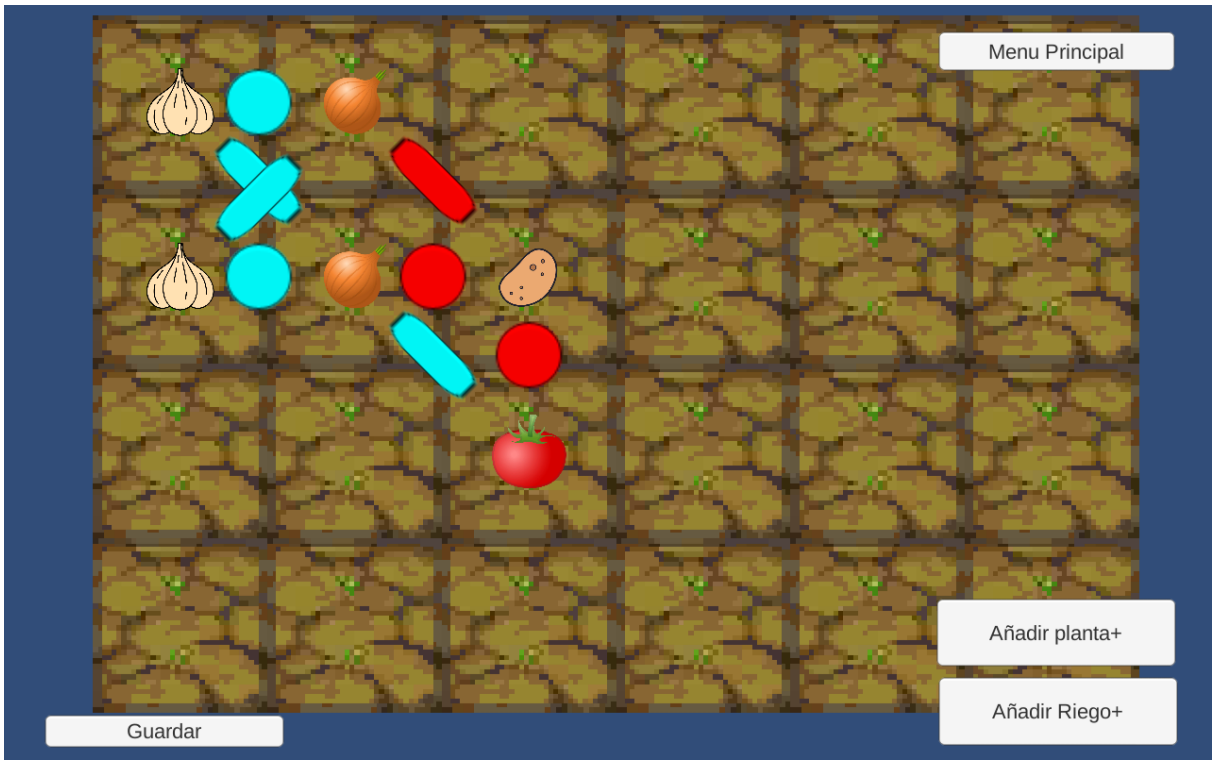

Figura 47: Captura de pantalla de un plano con varias plantas dentro de la escena del juego de la aplicación OrtiDis.

El usuario puede mover las plantas a otras casillas, mientras mantenga una de las plantas pulsada esta cambiará a un color rojizo y además se iluminará la casilla donde se colocará en caso de que el usuario la suelte, si se puede dejar en esa casilla esta se iluminará de color verde azulado, en caso de que no sea posible colocar la planta en la casilla esta se iluminará de color rojo oscuro. Si el usuario desea borrar una planta deberá arrastrarla fuera del plano, esto será visible gracias a que la planta cambiará a un color negro para mostrar que si el usuario la suelta se borrará del plano.

Por último si el usuario pulsa el botón “Añadir Riego+” se activará el sistema de riego óptimo según las dimensiones del plano.

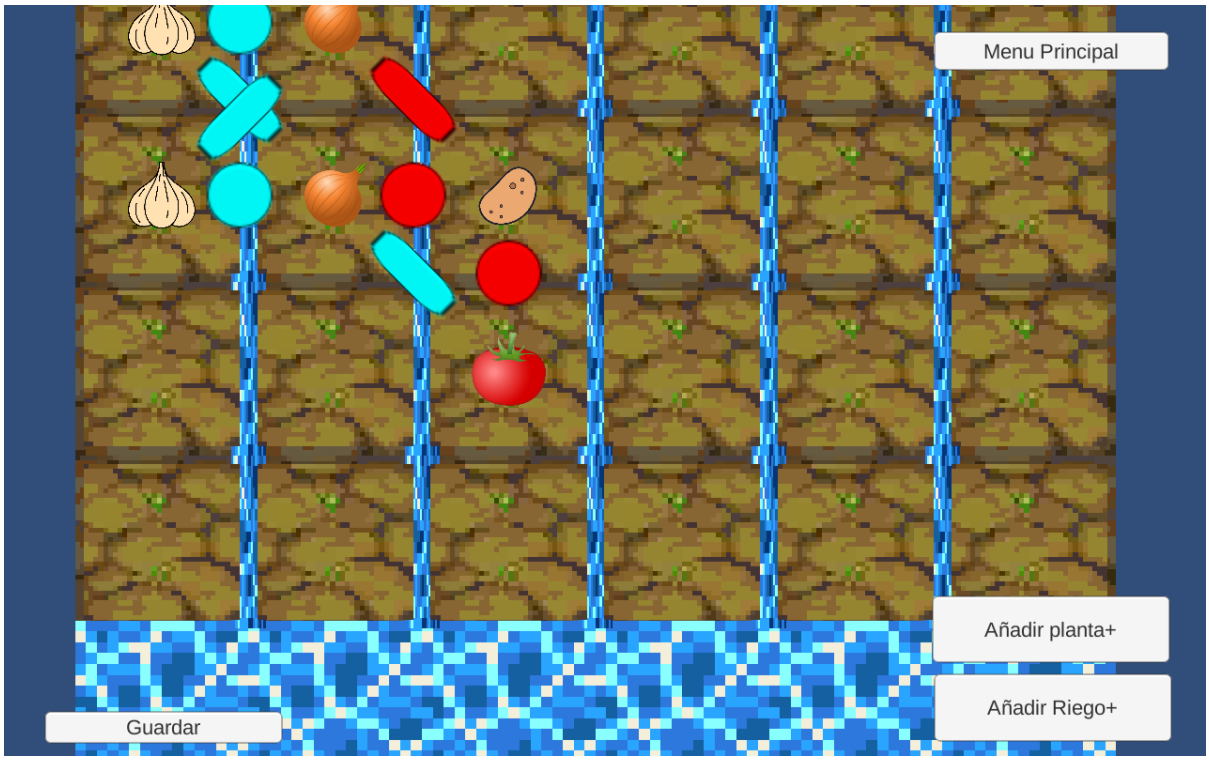

Figura 48: Captura de pantalla de un plano con varias plantas y el riego activo dentro de la escena del juego de la aplicación OrtiDis.

Por último para cerrar la aplicación el usuario deberá volver al menú principal y pulsar el botón “Salir”, podrá comprobar que el progreso ha sido guardado ya que aparecerá el slot con el nombre y las dimensiones del plano que creó, para volver al plano sólo debe de pulsar sobre este slot, el botón “Borrar” borrará el slot que se encuentra a su izquierda.

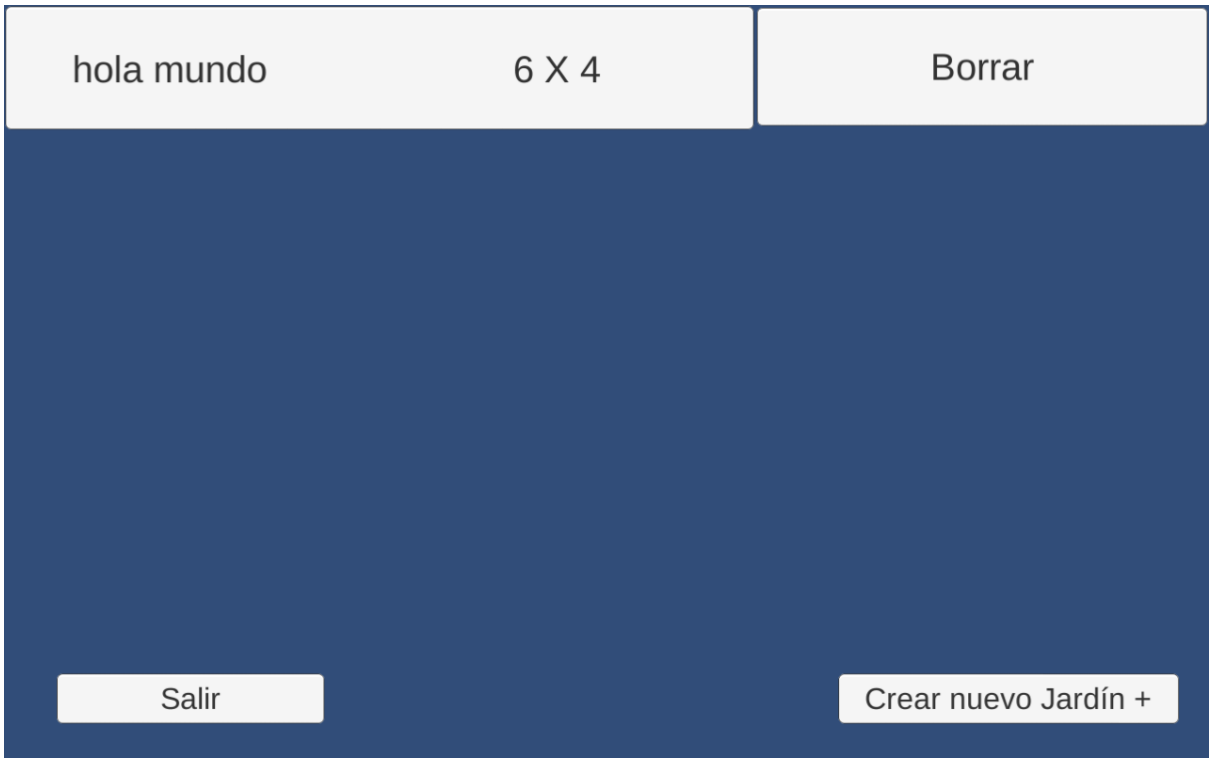

Figura 49: Captura de pantalla del menú principal de la aplicación OrtiDis con un jardín guardado.