E.T.S. de Ingeniería Industrial,
Informática y de Telecomunicación

Desarrollo de aplicaciones móviles para expedir certificados de eficiencia energética de edificios y para levantamiento de planos

Grado en Ingeniería Informática

Trabajo Fin de Grado

Pablo Urriza Iricibar

Edurne Barrenechea Tartas

Pamplona, 18 de junio de 2018

upna Universidad Pública de Navarra
Nafarroako Unibertsitate Publikoa

Índice

1. **Introducción** **1**

1.1 Resumen 1
1.2 Contexto 1

2. **Software utilizado** **3**

2.1 Kivy 3
2.2 Git Extensions 3
2.3 Eclipse 4
2.4 Buildozer 4

3. **Proyectos** **5**

3.1 Certificación energética de edificios 5
3.1.1 Gestión de errores 6
3.1.2 Nuevas funcionalidades 8

3.2 Levantamiento del plano de una habitación 10
3.2.1 Conceptos previos 11
3.2.2 Desarrollo 13

4. **Conclusiones y líneas futuras** **30**

4.1 Conclusiones 30
4.2 Líneas futuras 30

5. **Bibliografía** **31**

1. Introducción

1.1 Resumen

Este trabajo de fin de grado se centra en el desarrollo de dos aplicaciones destinadas a dispositivos móviles. Ambas implementadas en el lenguaje de programación Python, utilizando el framework Kivy (del cual se hablará más adelante), pero cuyos objetivos son muy diferentes.

El primer proyecto se basa en la necesidad de actualizar (y añadir) funcionalidades de una aplicación diseñada para expedir certificados de eficiencia energética de edificios. La motivación principal vino dada por el lanzamiento de la nueva versión de Android (Android 8) en octubre de 2017.

El segundo proyecto es el desarrollo completo de una aplicación cuyo objetivo es permitir al usuario generar el plano de una habitación junto a las medidas de esta mediante técnicas de realidad aumentada.

1.2 Contexto

Este trabajo de fin de grado se enmarca en la realización durante un año, de prácticas curriculares en la empresa Efinovatic. Efinovatic se sitúa dentro del mercado de la eficiencia energética de los edificios.

Esta empresa desarrolló el procedimiento oficial de certificación energética de los edificios existentes, CE3X. La empresa surgió con la idea de proporcionar una herramienta que permitiese realizar de forma eficaz, rápida y cómoda el certificado energético de los edificios. En el año 2005 publicaron la primera versión de dicha herramienta. Se puede ver las clasificaciones de eficiencia energética en la Figura 1.

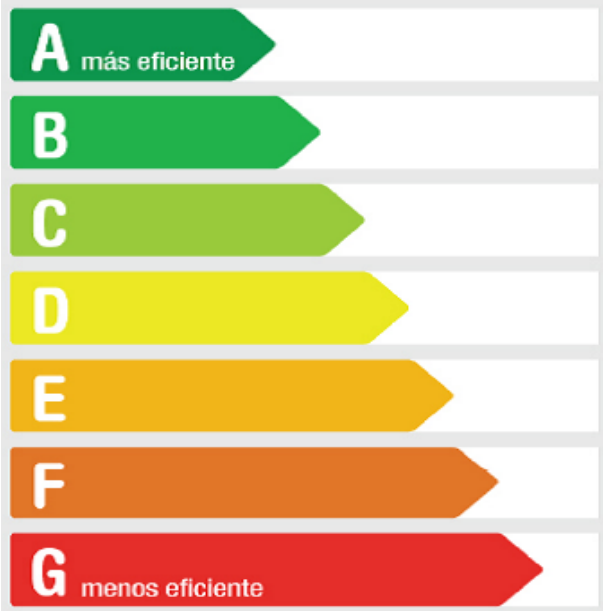


Figura 1. Calificaciones eficiencia energética

Desde el mes de junio del año 2013 cualquier persona que desee vender o alquilar una vivienda debe contar con el certificado energético de dicha vivienda. Es por ello que este mercado ha tenido un gran auge en los últimos años.

Sin ir más lejos, el Ministerio de Energía, Turismo y agenda digital dictaminó que a partir del mes de enero de 2016 solo se iban a admitir por los registros de las comunidades autónomas los certificados de eficiencia energética realizados con tres herramientas, entre las que se encontraba la realizada por Efinovatic, el CE3X.
Las empresas que han diseñado las otras dos herramientas restantes, Lider-Calener y Cerma, son los principales competidores de Efinovatic a la hora de realizar el certificado energético de edificios existentes.

Es por ello que Efinovatic está en proceso de conseguir la validación necesaria para que su programa, CE3X, pueda ser utilizado en la certificación energética de edificios de obra nueva.

Sin embargo, si bien es cierto que Lider-Calener y Cerma disponen de herramientas validadas para poder realizar los certificados de ambos tipos de edificios, el programa CE3X es el más utilizado actualmente en el ámbito de edificios existentes.

Actualmente la empresa se está dedicando en mayor medida a la realización de complementos personalizados a diferentes clientes (empresas principalmente dedicadas al mercado inmobiliario). Estas empresas son en su mayoría nacionales, aunque han realizado algún complemento para empresas extranjeras.
No obstante, Efinovatic se está convirtiendo con el paso de los años en una empresa de soluciones software de todo tipo, saliendo de su mercado habitual de la certificación energética.

El CE3X es un programa mediante el cual un usuario puede generar el certificado de eficiencia energética de cualquier edificio existente. El primer proyecto que se realiza es una actualización de este software mediante la cual se modifican y añaden funcionalidades al mismo.

El segundo proyecto nace con la idea de permitir al usuario crear fácilmente un plano de cualquier habitación que desee y generando las medidas de las paredes de esta sin necesidad de realizar esas mediciones a mano.

1.3 Palabras clave

Aplicación Móvil, Eficiencia Energética, Realidad Aumentada, Planos, Android

2. Software utilizado

En este apartado se explican las principales herramientas software que se han utilizado en la realización de los dos proyectos, instalación y complementación entre ellas.

2.1 Kivy

Kivy es un framework para Python de código abierto y multiplataforma que permite desarrollar aplicaciones móviles. Contiene un gran conjunto de librerías y ejemplos, lo que provoca que sea perfecto tanto para usuarios novatos como para expertos.

Proporciona un API casi idéntico al de Android y la forma de incluir librerías a los proyectos no podría ser más sencilla.

Una de las grandes ventajas de Kivy es la facilidad que da para interactuar con los elementos de la pantalla (botones, cuadros de texto, etc). Esto es debido a los archivos ".kv". Estos archivos sustituyen a los archivos ".xml" del lenguaje Android y es en ellos donde se implementa la estructura de la aplicación (de cada una de sus pantallas).

Kivy ha sido imprescindible en la realización de ambos proyectos ya que constituye el soporte idóneo para realizar aplicaciones móviles utilizando Python.

2.2 Git Extensions

Git Extensions es un conjunto de herramientas diseñadas para que el trabajo con Git en el sistema operativo Windows, sea mucho más intuitivo.

Si bien es cierto que el uso de Git mediante comandos no es excesivamente complicado, el entorno gráfico que proporciona Git Extensions facilita mucho el trabajo en equipo y el control de versiones, sobre todo si se trabajan en varias ramas al mismo tiempo.

La configuración de Git Extensions es muy sencilla, tan sólo hay que indicar dos tipos de repositorios por cada proyecto que se vaya a desarrollar. Un repositorio "local" (en el cual se van a guardar los archivos en el ordenador y en el que realizaremos las acciones de "commit") y un repositorio "remoto" (que se utiliza para sincronizar archivos con el resto de integrantes del equipo o para realizar el control de versiones).

El control de versiones es básico en cualquier proyecto de desarrollo software ya que no sólo facilita el trabajo en equipo si no que se puede emplear como método para generar copias de seguridad. Este hecho es realmente útil a la hora de realizar pruebas en las aplicaciones ya que permite volver a un estado anterior de las mismas de una forma realmente sencilla.

2.3 Eclipse

Eclipse es una plataforma de software compuesta por un conjunto de herramientas de programación de código abierto. Es compatible con múltiples lenguajes de programación y para poder obtener el soporte que ofrece esta plataforma es necesario instalar unos “plugins” o “extensiones” propias de la misma.

Estas extensiones son propias del lenguaje que se vaya a utilizar y entre sus funciones destacan la corrección de errores, optimización del código e importación de librerías externas.

Para la realización de ambos proyectos fue necesaria la instalación de PyDev (compatibilidad con archivos Python .py) y Django (servidor utilizado en el segundo proyecto).

Eclipse dispone, como la gran mayoría de plataformas de software, de un modo debug muy completo. Es muy intuitivo y es necesario para la realización de cualquier desarrollo de software.

2.4 Buildozer

Buildozer es una herramienta que permite el empaquetado de aplicaciones móviles de forma rápida y sencilla. Automatiza el proceso de compilación completo y descarga los requisitos necesarios tal y como Python-for-android, Android SDK, NDK, etc.

Buildozer define un archivo llamado “buildozer.spec” que se asimila al archivo “AndroidManifest.xml” de Android. En él se definen varios parámetros imprescindibles para el proceso de creación del APK como son, la versión Android objetivo, SDK mínimo necesario, NDK mínimo necesario, etc.

El principal inconveniente de esta herramienta se encuentra en la forma de depurar el código de la aplicación. A diferencia de Android Studio (IDE de desarrollo de aplicaciones móviles Android), Buildozer no constituye un IDE como tal si no que realiza el proceso de empaquetado. Esto provoca que si hay algún tipo de error en la aplicación no se puede localizar hasta el momento de la ejecución de la misma (esta significa que ya se ha tenido que instalar en el dispositivo) lo que ralentiza mucho el desarrollo del proyecto.

Otra de los contras de Buildozer es que sólo es posible utilizarlo en sistemas operativos Linux, por lo que fue necesario el uso de una máquina virtual durante el desarrollo de los proyectos.

3. Proyectos

En este apartado se explica detalladamente el desarrollo de los dos proyectos que componen este trabajo de fin de grado. Se sigue en ambos la misma estructura, indicando todas las fases realizadas, así como los problemas y soluciones que han ido surgiendo durante el transcurso de los proyectos.

3.1 Certificación energética de edificios

Este proyecto se enmarca dentro del producto principal de la empresa Efinovatic, en la que, como comenté en el apartado de contextualización, han sido realizadas las prácticas durante este año. El software en cuestión es el CE3X, el cual permite al usuario generar el certificado de eficiencia energética de un edificio.

El CE3X es un programa muy completo en el que se solicita gran cantidad de datos del edificio del que se desee generar el certificado de eficiencia energética y para el cual es necesario disponer de un mínimo de conocimientos en temas de arquitectura, energía y sostenibilidad. En la Figura 2 se muestra el diagrama de casos de uso de esta aplicación.

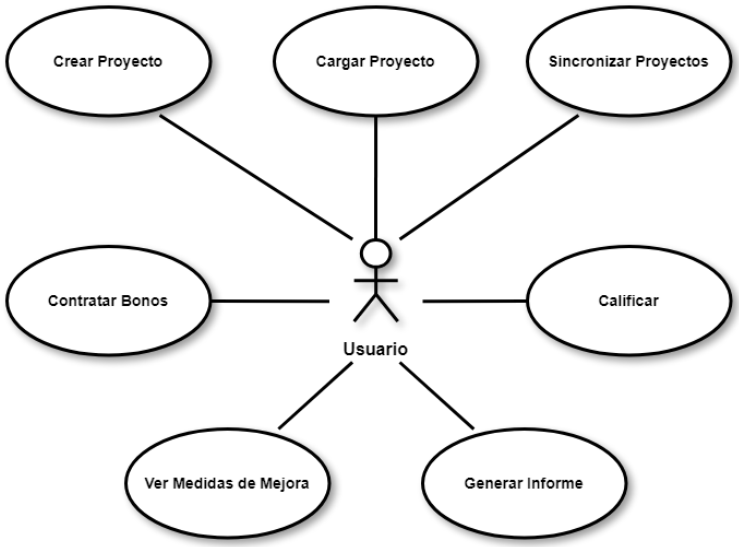


Figura 2. Diagrama de casos de uso del CE3X

El proyecto surge de la necesidad de adaptar las funcionalidades ya existentes del CE3X (y otras nuevas) a la actualización que realiza Android al pasar a su versión 8 en octubre del año 2017.

3.1.1 Gestión de errores

Debido a la importancia de los errores que se habían generado en la aplicación se priorizó su resolución antes de comenzar el desarrollo de las nuevas funcionalidades.

Responsive

El primer problema que supuso esta actualización, que fue uno de los mayores inconvenientes de todo el proyecto como se explicará más adelante, ha sido que la forma que se había implementado para que la aplicación (su aspecto visual) se adaptara a las dimensiones y resolución del dispositivo (hecho denominado como "responsive") en el que se utilizaba, dejó de funcionar.

El proyecto utiliza un archivo de configuración de "widgets" en el que se definen los parámetros generales (tamaño, color, …) de los componentes de la aplicación (botones, imágenes, etc). De este modo, para definir un tamaño tal que la aplicación se visualice correctamente en cualquier dispositivo (móvil o tablet) se utilizaba la cadena de caracteres "%idp" antes de cualquiera de estos parámetros, lo que provocaba que las dimensiones del widget se adecuen a la pantalla en la que se estaba visualizando la aplicación.
La actualización de Android ha provocado que este mecanismo de auto dimensionado deje de funcionar y ha sido necesario buscar una alternativa dada la importancia que tiene este hecho en la usabilidad de la aplicación.
La solución consiste en obtener las dimensiones (largo y ancho) de la pantalla y mediante una relación matemática entre ellas obtener un factor que permita escalar los componentes de la aplicación correctamente, independientemente del dispositivo que se utilice (multiplicando un valor constante por este factor).
En la Figura 3 se puede ver el menú principal del CE3X una vez corregido este problema.

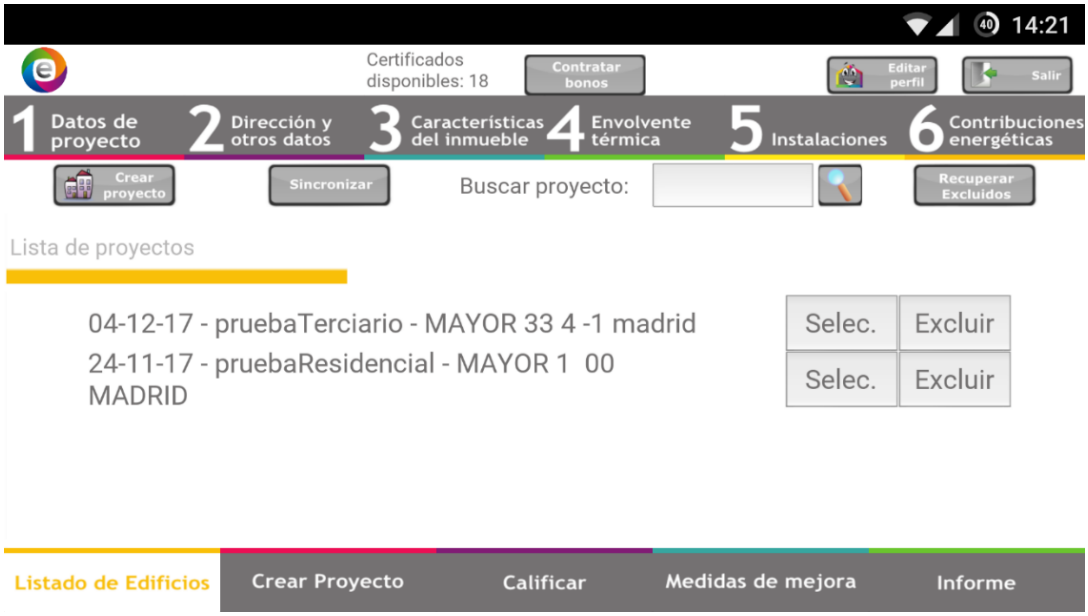


Figura 3. Pantalla Principal CE3X

Almacenamiento

Esta actualización también ha afectado a los permisos que requieren las aplicaciones para acceder a distintas funciones de los dispositivos (cámara, ubicación, etc). En este caso la que ha generado problemas fue el almacenamiento.
El CE3X solicita al usuario imágenes del edificio del cual se esté creando el certificado de eficiencia energética para almacenarlas en el servidor y en el propio dispositivo del usuario (en la carpeta donde se instala la aplicación), de modo que luego accede a estas imágenes para añadirlas al certificado. Estas imágenes se pueden seleccionar directamente (si ya están almacenadas en el dispositivo) o sacarlas directamente con la cámara (este es el caso que generó el problema).
Sin embargo, tal y cómo funciona Buildozer (herramienta de empaquetado de aplicaciones que se ha utilizado, explicada en el apartado "Herramientas"), esta carpeta donde se guardan los archivos básicos del usuario sólo es accesible mediante el ordenador o si el dispositivo se encuentra "rooteado", algo que no es muy común entre los usuarios.
De este modo la aplicación no es capaz de acceder a las imágenes que se sacan con la cámara, ya que ni el propio usuario utilizando el explorador de archivos tiene esa posibilidad.
La solución que se toma consiste en que, a la hora de instalar la aplicación, ésta crea una carpeta en una ruta del dispositivo destinada a datos de aplicaciones instaladas y es ahí donde se almacenan las imágenes capturadas por el usuario.
Esta operación, aunque parece sencilla, no lo es tanto, ya que cada dispositivo tiene una ruta diferente a esta carpeta de datos de aplicaciones. Para que esto sea posible es necesario implementar varias funciones que combinen métodos del API de Android para obtener una ruta adecuada y óptima para crear la carpeta.
De este modo, y una vez que el usuario da los permisos necesarios (cámara y almacenamiento) que son básicos en muchas aplicaciones, el CE3X ya es capaz de guardar y eliminar las imágenes correctamente.

Gestión de permisos

A raíz de este problema con los permisos se ve que es necesario añadir un aviso al iniciar la aplicación que solicite al usuario la activación de dichos permisos. Este hecho, aunque parece bastante trivial es necesario ya que en algunos dispositivos no es posible activar los permisos automáticamente y es el propio usuario el que los debe activar manualmente accediendo desde la opción de Ajustes.
En la mayoría de los dispositivos, tal y cómo está diseñada la aplicación, nada más instalar la aplicación se abre una ventana emergente que permite administrar los permisos de forma más rápida, sin embargo, en versiones antiguas de Android esto no es posible.

Pyjnius

Para poder acceder a las funcionalidades propias del dispositivo, del hardware, se utiliza la librería Pyjnius. Esta librería permite el acceso a clases Java, y para los proyectos que se tratan en este trabajo de fin de grado era indispensable acceder a la clase Hardware.

Esta clase implementa todos los métodos necesarios para obtener la información sobre el dispositivo (sensores, cámara, versión del sistema operativo, rutas de almacenamiento, etc).
Su funcionamiento no es complicado, sin embargo, con la actualización se han modificado algunos métodos de esta clase y ha sido necesario revisar prácticamente la aplicación al completo debido al gran uso que se le daba.

3.1.2 Nuevas funcionalidades

Una vez corregidos todos los errores generados por la actualización de Android se comienza la implementación de nuevas funcionalidades en la aplicación.

Acceso Google Maps

La primera de ellas tiene como objetivo permitir al usuario visualizar la localización del edificio que se estaba certificando, accediendo a la aplicación de Google Maps.
La implementación se basa principalmente en el API de Google ya que tiene gran número de ejemplos y nuestra necesidad se basaba casi al completo en uno de ellos.
Se incluye un botón junto al edificio en cuestión que redirige a Google Maps añadiendo a la url la información con la dirección de dicha vivienda.

Recordar datos

La aplicación dispone de un servicio de gestión de acceso (usuario/contraseña), sin embargo, éste no contaba todavía con la posibilidad de recordar estos datos, lo que permite al usuario no tener que introducirlos cada vez que accede.
Esta posibilidad es muy común en todas las aplicaciones que disponen de un login y parecía necesario implementarla.
Tan sólo ha sido necesario añadir al fichero de configuración que crea Kivy (framework utilizado en los proyectos) la información sobre usuario y contraseña.
De este modo si el usuario lo desea puede recordar las credenciales del último acceso en la pantalla de login de la aplicación, tal y como se ve en la Figura 4.

Figura 4. Pantalla de acceso al CE3X

Patrón de sombras

El patrón de sombras permite definir la sombra arrojada por objetos remotos sobre nuestro edificio. Para poder generarlo es necesario que el usuario introduzca datos sobre las distintas fachadas de este.
Para ello se utiliza la información captada por el luxómetro y el sensor de orientación del dispositivo. Estos datos se procesan y se almacenan formando una gráfica con todos ellos, que será la que definirá el patrón de sombras del edificio que se esté certificando. También se requiere el acceso a la cámara ya que el usuario para poder generar el patrón de sombras debe, como se comentaba anteriormente, introducir información sobre las fachadas del edificio. Para poder procesar cada una de ellas, el usuario debe "apuntar" con la cámara del dispositivo a cada una de las esquinas de la fachada, de este modo se almacena toda la información necesaria combinando los datos que capturan el luxómetro y el sensor de orientación. En la Figura 5 se puede observar un ejemplo de patrón de sombras completo.

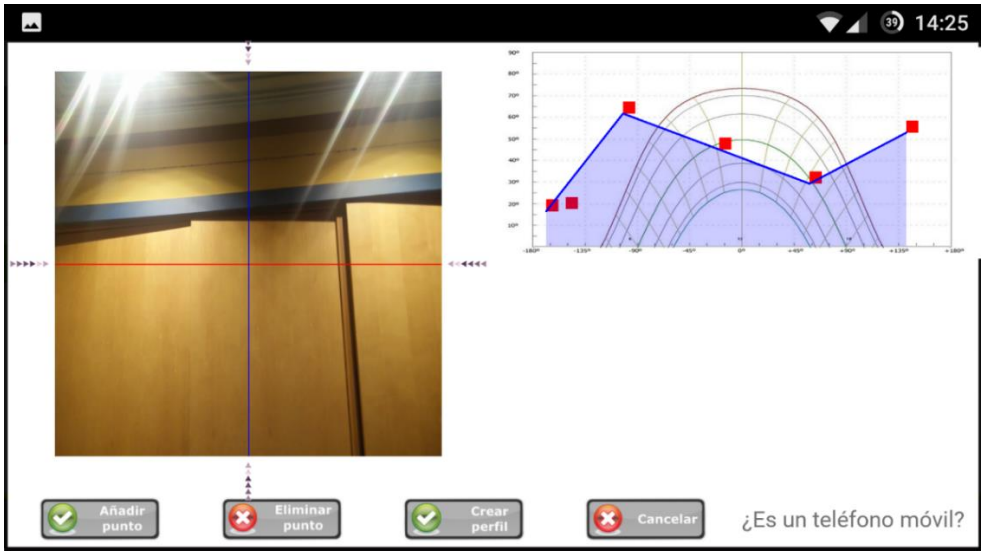


Figura 5. Patrón de Sombras

3.2 Levantamiento del plano de una habitación

Este proyecto surge como un posible complemento para el CE3X pero que finalmente compone una aplicación móvil completa de forma independiente. El propósito de ésta es permitir al usuario generar el plano de una habitación junto a las medidas de las paredes que la componen. Para ello el usuario tan sólo debe introducir las esquinas de las paredes que forman la habitación.

En la Figura 6 se muestra el diagrama de casos de uso de la aplicación. El usuario una vez accede a la misma tiene la opción de añadir las esquinas de la estancia y acto seguido generar el plano de la habitación. Si lo desea puede cargar además planos que ya se hubieran generado anteriormente, con la idea de mostrar en pantalla los planos de toda una vivienda si fuera el caso.

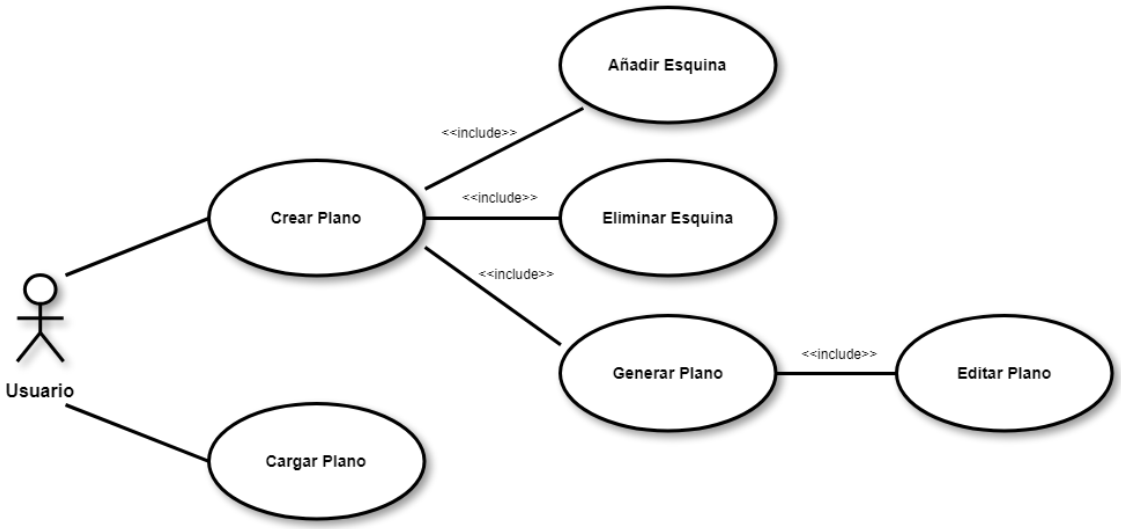


Figura 6. Diagrama de casos de uso

En la Figura 7 se muestra el diagrama de estados de la ejecución principal de la aplicación, que dispone de dos posibles estados finales (salir de la ejecución o guardar el plano generado).

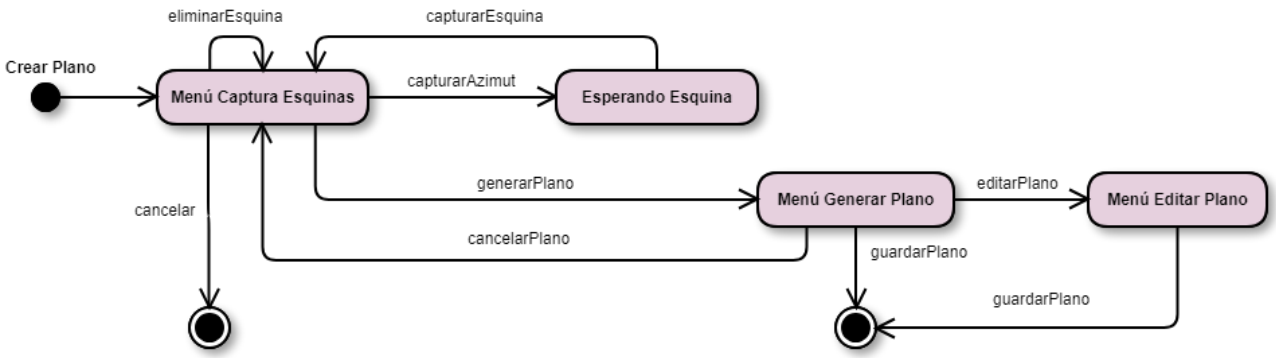


Figura 7. Diagrama de estados de la ejecución principal

3.2.1 Conceptos previos

Antes de comenzar con la explicación del proyecto es necesario aclarar varios conceptos relacionados con los sensores y la cámara de los dispositivos móviles, que son fundamentales para comprender el desarrollo del mismo.

Sensor de orientación

El sensor de orientación de los dispositivos captura tres datos al mismo tiempo, denominados azimut, roll y pitch. El **azimut** es el ángulo que forma el dispositivo con el Norte (de 0 a 360 grados), se puede observar que también se define como el movimiento sobre el eje Z. El **roll** hace referencia al giro sobre el eje Y del dispositivo, por lo que se denomina la "elevación" del mismo (de -90 a 90 grados) y, por último, el **pitch** hace referencia al movimiento sobre el eje X, conocido como la "inclinación" (de -180 a 180 grados). En la figura 8 se puede ver claramente cómo se definen estos datos y los cambios que provocan en los ejes respectivamente:

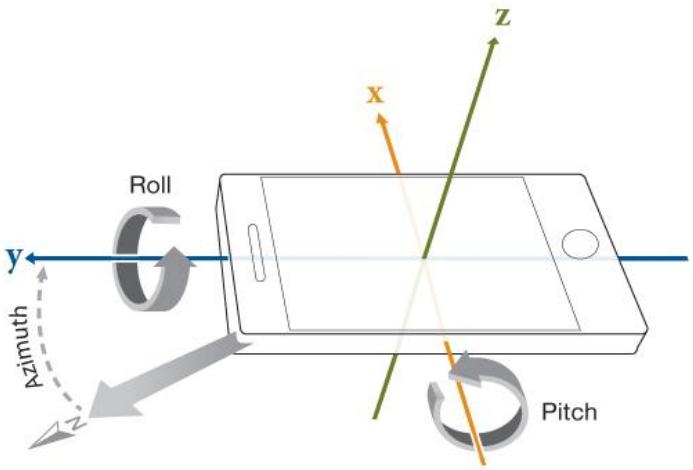


Figura 8. Representación de azimut, roll y pitch

Estos tres valores son imprescindibles para el desarrollo del proyecto, ya que permiten calcular las coordenadas espaciales de los diferentes puntos que introduce el usuario para generar el plano de la habitación. Este cálculo se detalla más adelante junto con el papel que toman el azimut, roll y pitch.

Cámara

Para entender el proceso de la captura de imágenes con la cámara del dispositivo móvil, o con cualquier cámara fotográfica en general, es necesario familiarizarse con los diferentes planos que toman parte en el mismo.

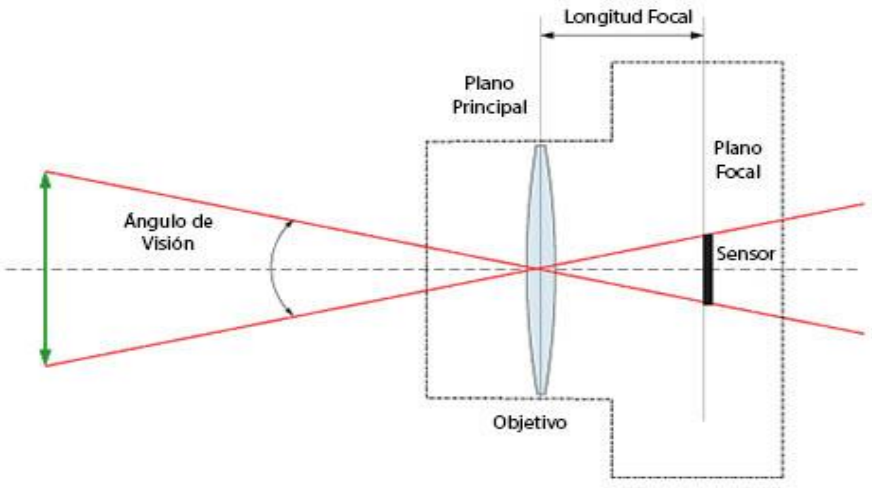


Figura 9. Representación de planos ópticos

En la Figura 9 se observan claramente los tres planos que toman parte en este proyecto.

En primer lugar, nos encontramos con lo que en este proyecto hemos definido como **plano absoluto o plano espacial**, (en la Figura 8 estaría en la posición de la flecha de color verde), en este plano, el objeto (punto o esquina en nuestro caso) tiene las dimensiones (distancias) reales.
Es por ello que si nosotros nos encontramos en el punto (0, 0, 0) absoluto y enfocamos hacia un punto que se encuentra en la posición (4, 3, 2), las **coordenadas absolutas** o espaciales de este punto serán (4, 3, 2). Tomando como medida de distancia **metros**, esto significa que el punto está a 4 metros en el eje X, a 3 metros en el eje Y y a 2 metros de altura (eje Z) con respecto a nuestra posición.

A continuación, tenemos el plano principal, el cual a partir de ahora denominaremos **plano dispositivo**. Como se puede apreciar en la Figura 7, los ejes X e Y del dispositivo móvil se encuentran intercambiados, sin embargo, esto sólo ocurre para los dispositivos cuya orientación por defecto es la vertical (la gran mayoría de teléfono móviles), ya que para las tablets lo más común es que la orientación por defecto sea la horizontal.
Para establecer un criterio común independiente del dispositivo en el que se utilice la aplicación se decide tomar como orientación por defecto la horizontal, de este modo al iniciar la aplicación se comprueba cual es la orientación por defecto del dispositivo y en caso de que fuera vertical (teléfono móvil), se intercambiarían los valores del eje X e Y.

El fundamento del plano dispositivo se basa en comprender cómo se relacionan los ejes **absolutos o espaciales** con los ejes del **dispositivo** cuando el usuario sostiene la tablet mientras utiliza la aplicación, ver Figura 10.

Figura 10. Colocación adecuada de la tablet

Recordamos que los ejes en el dispositivo (hablamos siempre de tablets) se sitúan del modo que se refleja en la Figura 8 intercambiando los ejes X e Y. Por tanto, el usuario al sostener la tablet como se ve en la Figura 10 realiza de forma involuntaria un giro de 90° respecto al eje X. Lo que genera un cambio en los ejes Y y Z.
De este modo la relación entre los ejes espaciales y los del dispositivo quedan de la siguiente manera:

- Eje X Dispositivo = Eje X Absoluto/Espacial
- Eje Y Dispositivo = Eje Z Absoluto/Espacial
- Eje Z Dispositivo = Eje Y Absoluto/Espacial

Esta conversión es tan sólo un cambio en el sistema de referencia, no se aplica de modo directo sobre las coordenadas, el paso de coordenadas absolutas a coordenadas del dispositivo se explica en apartados posteriores.

Por último, nos encontramos con el **plano focal**. El plano focal se define desde el punto donde se encuentre el sensor de la cámara. Este plano se usa como paso intermedio antes de llegar al **plano pantalla**, el cual ya es el que está en contacto directo con el usuario y en el que las distancias se miden en **píxeles**. Esta transformación de metros a píxeles se detalla más adelante.

Una vez definidos los principios básicos del sensor de orientación y del funcionamiento de la cámara de los dispositivos móviles pasamos al desarrollo del proyecto.

3.2.3 Desarrollo

En este apartado se detallan los pasos que debe seguir el usuario para establecer correctamente los datos de la habitación (captura de las esquinas) junto con los cálculos matemáticos necesarios para el procesamiento de estos antes de la generación del plano.

Procesado de datos capturados por el sensor de orientación

Al iniciar la aplicación se activa el sensor de orientación y a través de la clase Hardware de Java y el método autoclass (este método permite acceder a clases desarrolladas en código Java desde Python) de la librería Pyjnius. De este modo se

comienzan a recibir los tres datos explicados en el apartado 3.2.1 el azimut, el roll y el pitch.

Estos datos se capturan sosteniendo el tablet en alto, por tanto, es normal que no mantengan una constancia adecuada para conseguir unos resultados precisos en las medidas. Es por ello por lo que se implementa un filtrado de estos, con el objetivo de que el usuario mantenga una posición fija un breve período de tiempo y así las lecturas del azimut, roll y pitch se regulen.
Este filtrado se basa en calcular el error de 10 medias, utilizando 10 valores por cada media, y en el momento en el que el error es menor a 0.2 se almacena la última lectura. Este cálculo se realiza para cada uno de los tres datos capturados, azimut, roll y pitch.

Captura de azimut

Una vez que se realiza el filtrado de los datos capturados por el sensor se comienza el proceso de almacenamiento de coordenadas espaciales.

No obstante, antes de capturar la esquina como tal, el usuario debe realizar una lectura previa mediante la cual se almacena el azimut correspondiente a la esquina que va a almacenar a continuación.
El valor del azimut (véase la Figura 8) se puede utilizar para conocer la diferencia de grados existente entre esquinas consecutivas de la habitación. Esta diferencia es imprescindible en el cálculo de la medida de la pared como se demuestra más adelante.

En primer lugar, se trata de leer el azimut al mismo tiempo que se captura la esquina, sin embargo, tras varias pruebas se aprecia que el valor del azimut almacenado tiene un error lo suficientemente notorio como para darlo por válido. Esto se debe a que cuando se captura la esquina el dispositivo está girado al menos en 2 ejes espaciales.

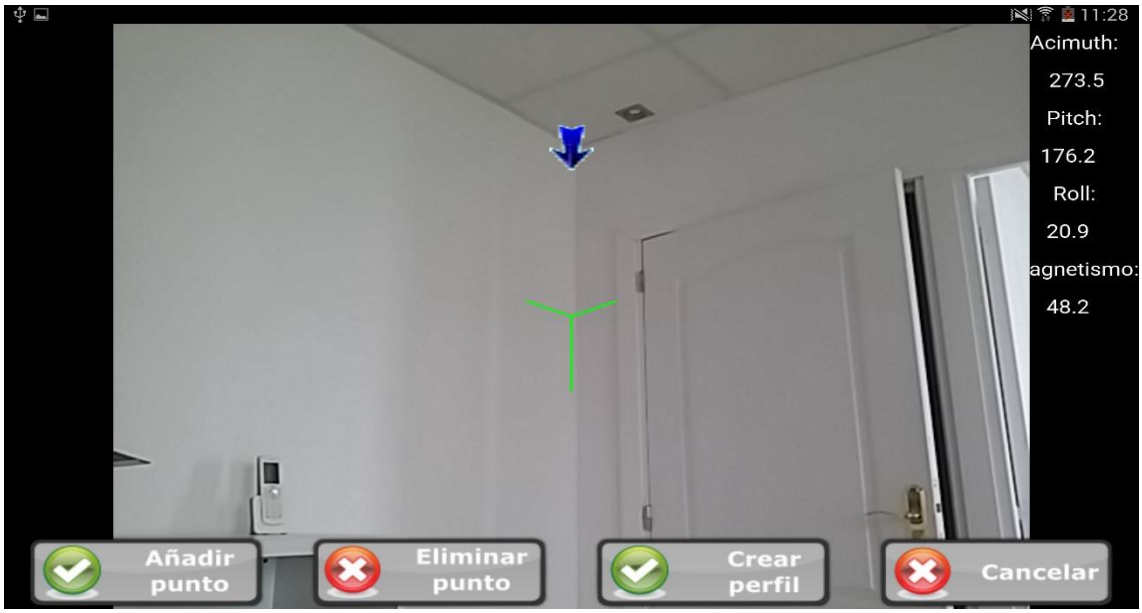


Figura 11. Captura del azimut

La solución que se toma consiste en realizar una lectura previa a la esquina en la cual se fuerza al usuario a tener la tablet en posición perpendicular al suelo de la habitación (roll = 0°) de modo que el valor del azimut que se almacena es mucho más preciso. Para ello se utilizan dos flechas azules (Ver figura 11) que indican al usuario si debe subir o bajar la tablet para llegar al punto en el que el roll sea 0.

En el momento en el que el roll se estabiliza en ese valor, la tablet almacena el azimut automáticamente y para informar al usuario se produce una vibración por parte del dispositivo.

Captura de puntos/esquinas de la habitación

Una vez almacenado el azimut se procede a la captura de la esquina. Dado que el azimut se almacena sin que el usuario interactúe con la pantalla, parece necesario añadir un indicador para que se sepa que hay que "apuntar" a la esquina. Es por ello que se introduce una flecha azul bajo un mensaje "Apunta hacia la esquina del suelo" mediante el cual se informa al usuario de cuál debe ser su siguiente paso. Ver Figura 12.

Figura 12. Captura de una esquina

Llegados a este punto se plantea la opción de que la cruz que se utiliza en la pantalla para que el usuario sepa donde está "apuntando" cambie del color rojo que tiene normalmente a un color verde y que sea en ese instante cuando el usuario pueda pulsar el botón "Añadir punto".
No obstante, esta situación puede causar errores en el cálculo posterior del plano ya que, si el usuario se adelanta a la estabilización de los datos y pulsa el botón, la esquina se almacena en la aplicación. Esto produce que las medidas no se calculen correctamente y que la forma del plano no sea la adecuada.

Es por ello que se decide implementar en la aplicación lo que se ha denominado como "modo experto". Este modo permite, una vez que se han estabilizado las lecturas del sensor, se almacenen automáticamente estos datos y se le comunique al usuario generando una vibración en el dispositivo. Representado en la Figura 13.

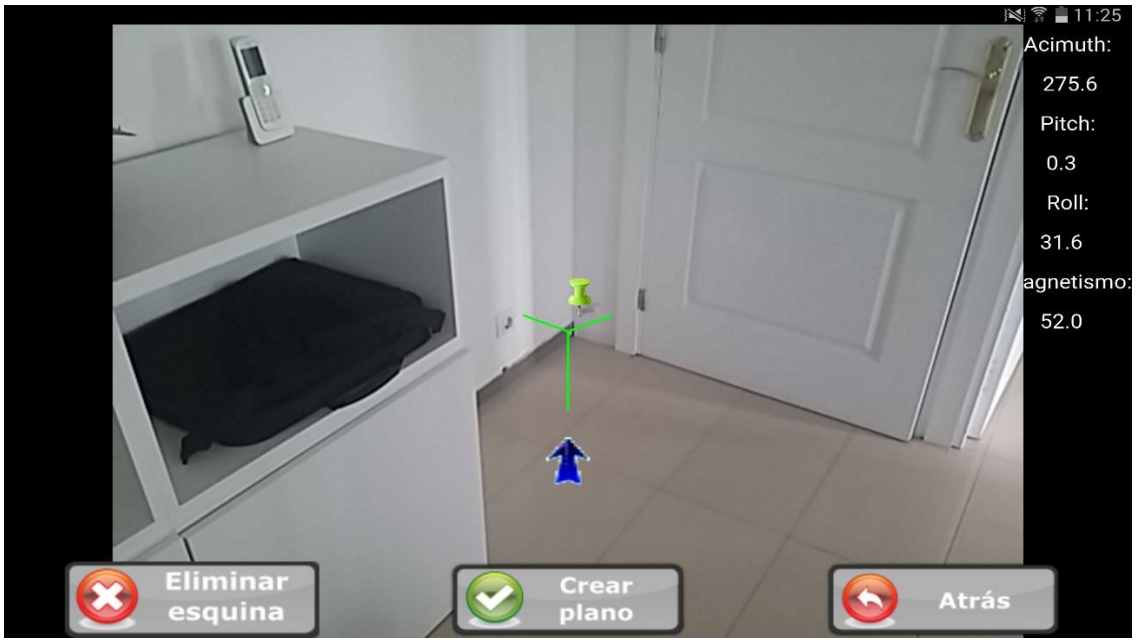


Figura 13. Esquina ya capturada

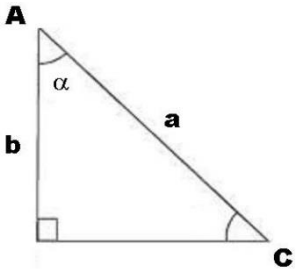
En el momento que se captura una esquina ésta aparece representada en la pantalla con un icono de una chincheta. No obstante, para poder calcular la posición donde se debe mostrar la esquina es necesario realizar el siguiente proceso:

Coord. Iniciales → Coord. Espaciales → Coord. Dispositivo → Coord. Focal → Coord. Pantalla

Esta secuencia conlleva un cálculo matemático bastante extenso y que se pasa a explicar detalladamente a continuación.

Coordenadas Iniciales → Coordenadas Absolutas

El plano dispositivo tiene su origen de coordenadas en el centro de la pantalla (cruz). De este modo, en el instante que se captura una esquina, se generan las coordenadas iniciales de este punto (x, y, z) = (0, 0, "distancia"). Este valor "distancia" hace referencia a la distancia existente entre el dispositivo y la esquina que se acaba de almacenar. Se muestra la representación gráfica de la situación en la Figura 14.



A == Punto dónde se encuentra el dispositivo
C == Esquina que se acaba de capturar
α = Roll (sensor de orientación del dispositivo)
b = Altura del usuario
a = Distancia que se desea calcular

Figura 14. Cálculo distancia a la esquina

Una vez planteado el supuesto parece claro que la distancia (a) que se desea obtener se calcula utilizando el coseno del ángulo α, que no es más que la "inclinación" de la tablet, es decir, el roll.

$$\cos\alpha = \frac{b}{a} \rightarrow a = \frac{b}{\cos\alpha}$$

De esta forma el vector de **coordenadas iniciales** queda definido, (x, y, z) = (0, 0, a). A continuación, se detalla como proseguir en el cálculo de las coordenadas absolutas de la esquina (valores que se mantienen **constantes** durante toda la ejecución, a menos que se elimine la esquina manualmente)**.**

Para poder representar el movimiento que se realiza con el dispositivo al moverlo, ya sea en el momento de capturar una esquina o cuando se pasa de una a otra, es necesario calcular lo que se denomina a partir de ahora **matriz de transformación.** Esta matriz representa la orientación de la tablet en cada instante y se calcula a partir de las rotaciones en los 3 ejes espaciales (X, Y, Z).
Para calcularla se obtiene 1 **matriz de rotación** por cada eje, las cuales se obtienen a continuación.

Para calcular cada una de las tres matrices que definen la rotación de la tablet en cada eje espacial se definen tres vectores unitarios que representan dichos giros:

Vector Unitario Rotación Eje X: (1, 0, 0)
Vector Unitario Rotación Eje Y: (0, 1, 0)
Vector Unitario Rotación Eje Z: (0, 0, 1)

Para poder realizar la demostración del cálculo de estas matrices, éste se realiza sobre un espacio 3D en el cual se mantiene constante el eje sobre el que se está produciendo la rotación, de modo que se puede simplificar el escenario a un tratamiento sobre 2D.

Para ello se supone un punto P inicial y un punto Q final, con sus respectivos ángulos α y θ, inicial y final respectivamente. Los escenarios se han conformado de forma que el vector unitario referente al eje sobre el que se rota se sitúa en perpendicular al plano que forman los otros dos ejes.

Las representaciones gráficas de las rotaciones sobre el eje Z, eje Y y eje X se pueden ver en las Figuras 15, 16 y 17 respectivamente. Junto al proceso de cálculo de las matrices de rotación para cada uno de los tres ejes.

[Recordatorio: el roll hace referencia al eje X, el pitch al eje Y y el azimut al eje Z]

Rotación Eje Z

Se define R como el radio de la circunferencia que simboliza el giro.

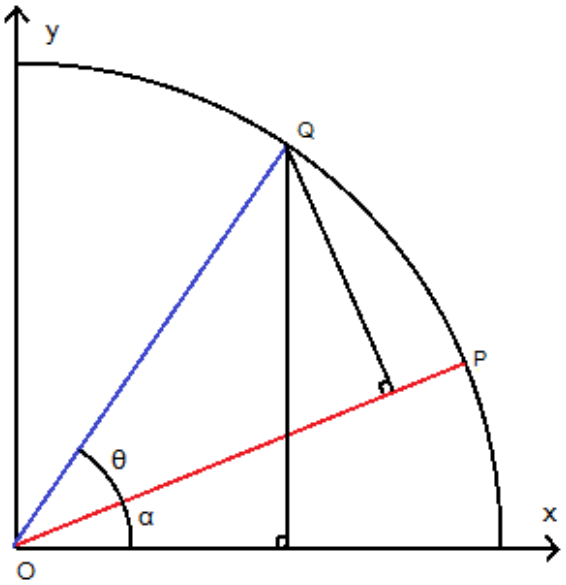


Figura 15. Representación rotación sobre Eje Z

P == Punto Inicial (x, y, z)

Q == Punto Rotado (x', y', z')

α = Ángulo Inicial

θ = Ángulo de rotación (**azimut**)

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} R\cos(\alpha) \\ R\sin(\alpha) \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} R\cos(\alpha+\theta) \\ R\sin(\alpha+\theta) \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} R\,[\cos(\alpha)\cos(\theta) - \sin(\alpha)\sin(\theta)] \\ R\,[\cos(\alpha)\sin(\theta) + \sin(\alpha)\cos(\theta)] \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x\cos(\theta) - y\sin(\theta) \\ x\sin(\theta) + y\sin(\theta) \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

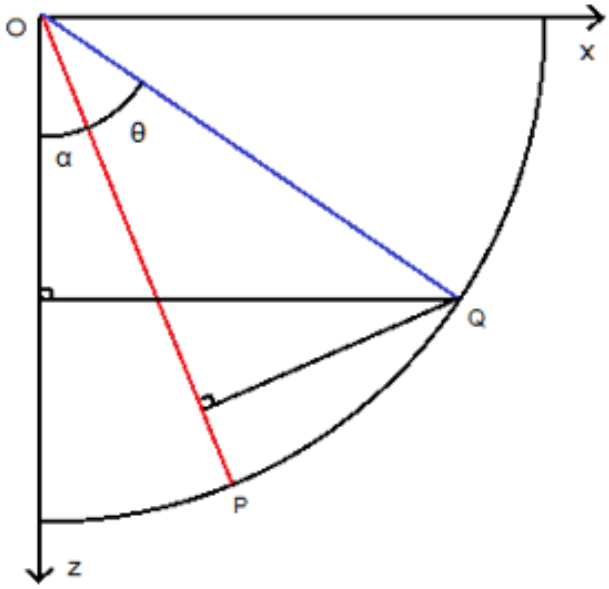
Matriz de rotación Z:

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

Al realizarse la rotación sobre el eje Z → z' = z

Rotación Eje Y

Se define R como el radio de la circunferencia que simboliza el giro.



P == Punto Inicial (x, y, z)

Q == Punto Rotado (x', y', z')

α = Ángulo Inicial

θ = Ángulo de rotación (**pitch**)

Figura 16. Representación rotación sobre Eje Y

$$\begin{bmatrix} x \\ z \end{bmatrix} = \begin{bmatrix} R\cos(\alpha) \\ -R\sin(\alpha) \end{bmatrix}$$

$$\begin{bmatrix} x' \\ z' \end{bmatrix} = \begin{bmatrix} R\cos(\alpha+\theta) \\ -R\sin(\alpha+\theta) \end{bmatrix}$$

$$\begin{bmatrix} x' \\ z' \end{bmatrix} = \begin{bmatrix} R\left[\cos(\alpha)\cos(\theta)-\sin(\alpha)\sin(\theta)\right] \\ -R\left[\cos(\alpha)\sin(\theta)+\sin(\alpha)\cos(\theta)\right] \end{bmatrix}$$

$$\begin{bmatrix} x' \\ z' \end{bmatrix} = \begin{bmatrix} x\cos(\theta)+z\sin(\theta) \\ -x\sin(\theta)+z\sin(\theta) \end{bmatrix}$$

$$\begin{bmatrix} x' \\ z' \end{bmatrix} = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} x \\ z \end{bmatrix}$$

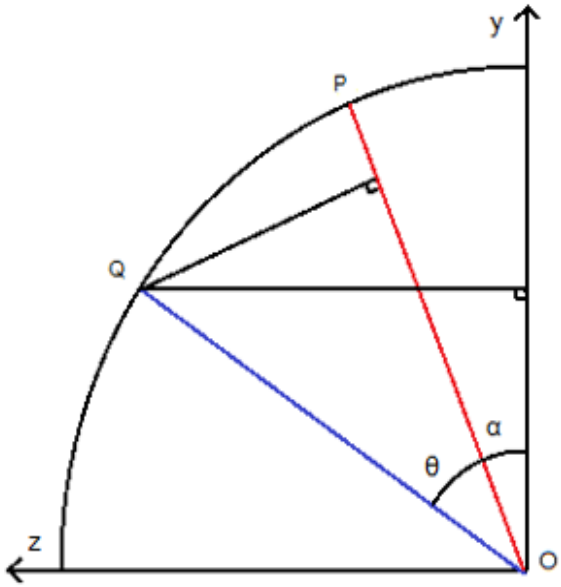
Matriz de rotación Y:

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

Al realizarse la rotación sobre el eje Y → y' = y

Rotación Eje X

Se define R como el radio de la circunferencia que simboliza el giro.



P == Punto Inicial (x, y, z)

Q == Punto Rotado (x', y', z')

α = Ángulo Inicial

θ = Ángulo de rotación (**roll**)

Figura 17. Representación rotación sobre Eje X

$$\begin{bmatrix} y \\ z \end{bmatrix} = \begin{bmatrix} R\cos(\alpha) \\ R\sin(\alpha) \end{bmatrix}$$

$$\begin{bmatrix} y' \\ z' \end{bmatrix} = \begin{bmatrix} R\cos(\alpha+\theta) \\ R\sin(\alpha+\theta) \end{bmatrix}$$

$$\begin{bmatrix} y' \\ z' \end{bmatrix} = \begin{bmatrix} R\left[\cos(\alpha)\cos(\theta)-\sin(\alpha)\sin(\theta)\right] \\ R\left[\cos(\alpha)\sin(\theta)+\sin(\alpha)\cos(\theta)\right] \end{bmatrix}$$

$$\begin{bmatrix} y' \\ z' \end{bmatrix} = \begin{bmatrix} y\cos(\theta)-z\sin(\theta) \\ y\sin(\theta)+z\sin(\theta) \end{bmatrix}$$

$$\begin{bmatrix} y' \\ z' \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} y \\ z \end{bmatrix}$$

Matriz de rotación X:

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) \\ 0 & \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

Al realizarse la rotación sobre el eje X → x' = x

De este modo obtenemos las tres matrices y la matrizTransformacion:

$$\text{matrizRotacionX} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha & 0 \\ 0 & \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$ Donde α = roll

$$\text{matrizRotacionY} = \begin{bmatrix} \cos\alpha & 0 & \sin\alpha & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\alpha & 0 & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$ Donde α = pitch

$$\text{matrizRotacionZ} = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 & 0 \\ \sin\alpha & \cos\alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$ Donde α = azimut

matrizTransformacion = matrizRotacionY · matrizRotacionZ · matrizRotacionX

Para la realización de cálculos entre matrices y vectores se utiliza la librería matemática **Numpy**. Numpy ofrece una extensa variedad de clases y métodos que facilitan el tratamiento de datos y los cálculos matriciales.

De modo que se obtienen las coordenadas absolutas de la esquina capturada:

coordenadasAbsolutas = matrizTransformacion · coordenadasIniciales

Estas coordenadas definen la posición de la esquina en el mundo “real”, se almacenan y se mantienen constantes.

Coordenadas Absolutas → Coordenadas Dispositivo

Este paso sólo tiene sentido en el proceso de reposicionamiento de esquinas ya capturadas en tiempo real. Esto se debe a que al añadir una nueva esquina las coordenadas del dispositivo se generan automáticamente, tal y como se explica en el paso anterior. Mientras que, al reposicionar esquinas ya existentes, es necesario recalcular sus coordenadas del dispositivo a partir de las coordenadas absolutas almacenadas.

Este cálculo se realiza de forma inversa al explicado en el paso explicado anteriormente:

coordenadasDispositivo = inversa(matrizTransformacion) · coordenadasAbsolutas

Coordenadas Dispositivo → Coordenadas Focales

Una vez que tenemos calculadas las coordenadas del dispositivo se obtienen las coordenadas focales. Para ello el primer paso es comprender la relación existente entre el plano dispositivo y el plano focal, es aquí donde toma parte un concepto fundamental, la **distancia focal.**
La distancia focal es una característica propia de cada cámara, representa la longitud existente entre el sensor y el objetivo, se mide en mm. Ver Figura 18.

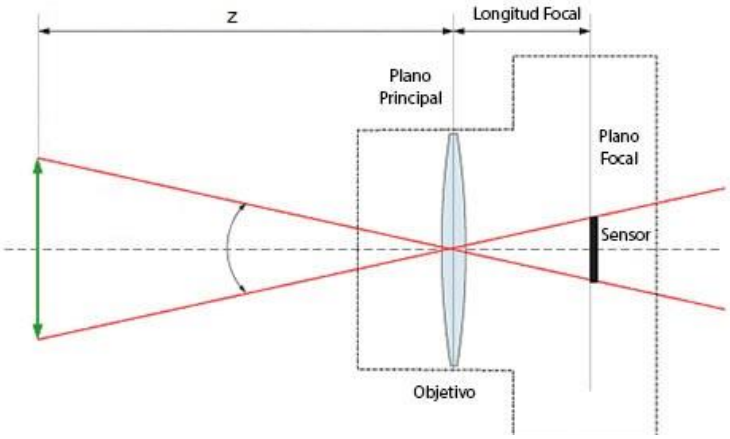


Figura 18. Relación plano focal y plano dispositivo

La distancia entre el plano principal y el plano real (z), no es más que la coordenada z de las coordenadas del dispositivo (distancia calculada en el paso 1)**.**

De este modo, se obtiene la escala entre el plano del dispositivo y el plano focal:

$$\text{factorEscalado} = \frac{distancia\ Focal}{coordenadaZDispositivo}$$

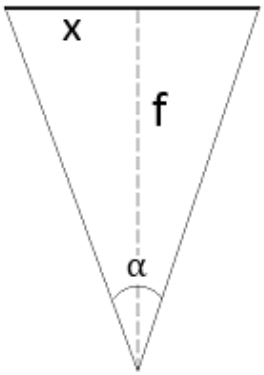
coordenadasFocales = coordenadasDispositivo · factorEscalado

Coordenadas Focales → Coordenadas Pantalla

El último paso consiste en obtener las coordenadas en la pantalla del dispositivo (píxeles) a partir de las coordenadas focales. Es importante resaltar que a partir del plano focal las coordenadas están compuestas sólo por componente x y componente y.

Para este proceso es necesario recurrir a dos datos característicos de la cámara del dispositivo denominados **ángulos de visión horizontal y vertical**. En este momento es necesario acceder, a través de la librería Window de Kivy, a las dimensiones (en píxeles) de los dos ejes de la pantalla.
Con estos datos es posible calcular la escala que hay que aplicar a las coordenadas focales para pasar a coordenadas pantalla. Esta escala, una diferente para cada eje, es la relación entre la dimensión del eje X de la pantalla y el tamaño máximo que permite enfocar el ángulo de visión, horizontal, en este caso.

En la Figura 19 se puede observar la representación del cálculo de la escala en el eje X (la escala del eje Y se obtendría de forma análoga).



α = Ángulo de visión horizontal

f = Distancia focal

Figura 19. Cálculo escalado pantalla

De este modo las escalas resultan ser:

$$\text{EscalaX} = \frac{dimensionEjeXPantalla}{2 \cdot f \cdot \tan(\alpha X/2)} \qquad \text{EscalaY} = \frac{dimensionEjeYPantalla}{2 \cdot f \cdot \tan(\alpha Y/2)}$$

$$\text{coordenadasPantalla} = \text{coordenadasFocales} \cdot \begin{bmatrix} \text{EscalaX} \\ \text{EscalaY} \end{bmatrix}$$

En el momento que se obtienen finalmente las coordenadas pantalla de la esquina, se muestra en pantalla el punto capturado y se procesa su comportamiento. Este comportamiento se detalla en el siguiente apartado junto al proceso para llevarlo a cabo.

Realidad aumentada e interacción con el usuario

Una vez finalizado el proceso de captura de esquinas pasamos a mejorar el diseño gráfico y la experiencia del usuario. Es necesario que el usuario pueda ver en todo momento donde se encuentran los datos que ya ha capturado o, mejor dicho, si regresa a una esquina ya capturada que le aparezca en pantalla la chincheta encima.

Además, parece buena idea implementar un método mediante el cual se generen automáticamente las paredes desde el momento que el usuario introduce 2 o más esquinas.
Se expone a continuación el proceso de implementación de ambas funcionalidades:

Reposicionamiento de esquinas en tiempo real

Cada vez que movemos el dispositivo se realiza el proceso indicado anteriormente, mediante el cual se recalculan las coordenadas de la pantalla de cada una de las esquinas almacenadas utilizando en cada momento la matriz de transformación. Esta matriz de transformación se calcula constantemente utilizando los datos capturados por el sensor de orientación (azimut, roll y pitch).

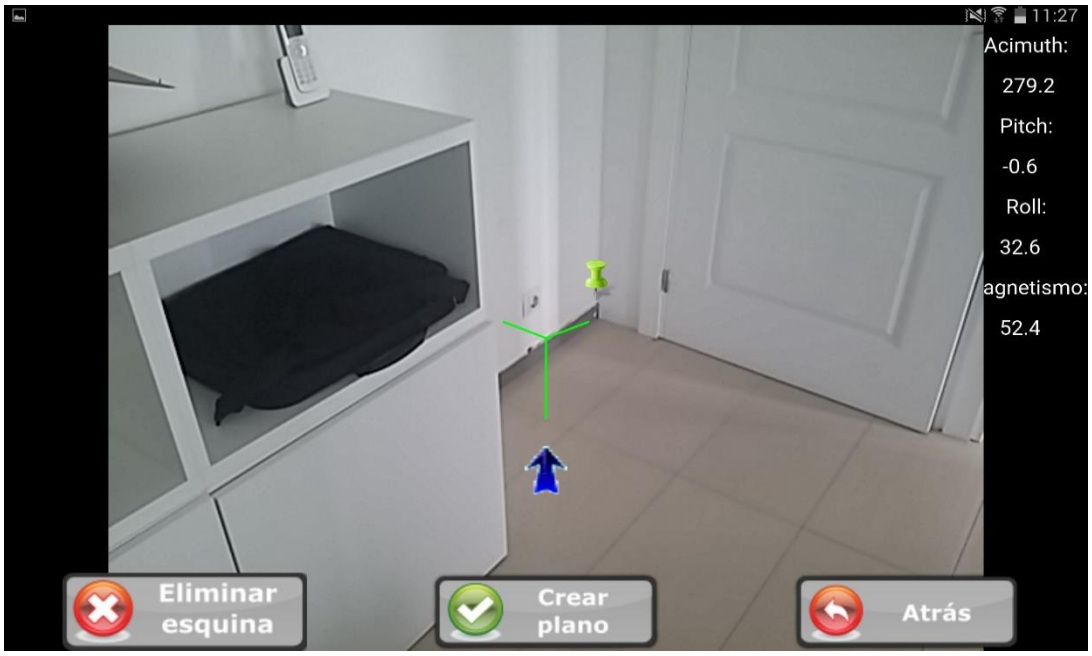


Figura 20. Reposicionamiento en tiempo real

Tal y como se puede ver en la Figura 20, una vez que se introduce una esquina ésta se reposiciona en tiempo real según hacia donde mire la tablet. Al ser un cálculo en tiempo real hay un pequeño error de calibrado que se corrige automáticamente si se deja quieta la tablet. El usuario puede conocer cuándo los datos que se muestran en pantalla son fiables gracias al color de la cruz (rojo si el error de los cálculos es alto o verde si se han estabilizado y son correctos).

Generación automática de paredes

En el momento que el usuario introduce 2 o más esquinas, la aplicación comienza a generar automáticamente las paredes entre las esquinas consecutivas.

El proceso es relativamente sencillo, se comprueba qué puntos aparecen en la pantalla y acto seguido se dibujan las líneas entre dichos puntos y las esquinas consecutivas (aparezcan o no en la pantalla). El framework Kivy ofrece una gran variedad de opciones gráficas y en este caso se utiliza la clase Line, definida por su inicio y final.
Se puede ver un ejemplo de esta funcionalidad en la figura 21.

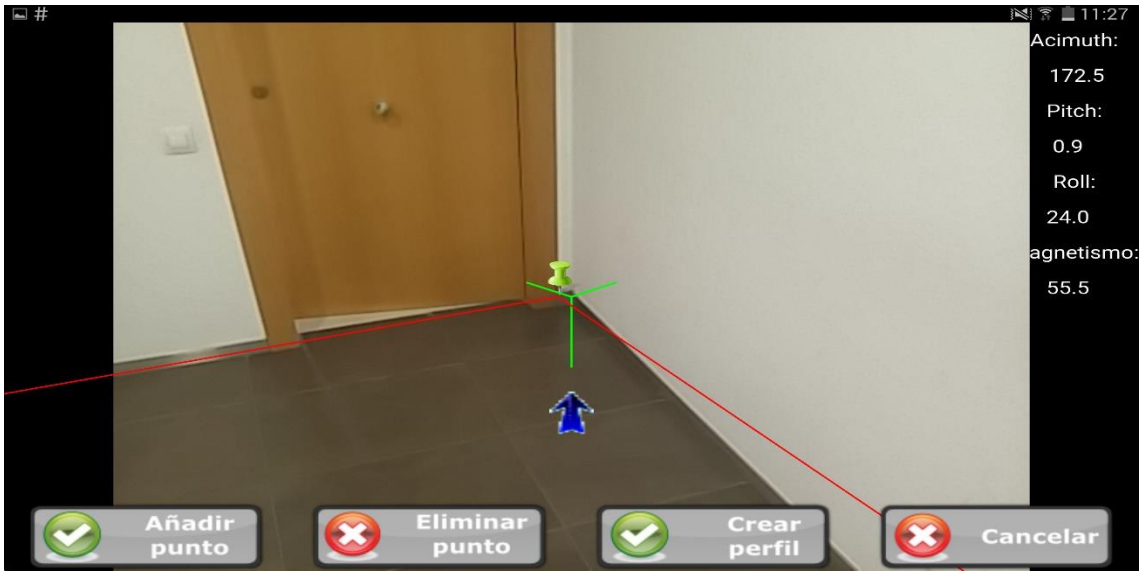


Figura 21. Generación automática de paredes

Creación del plano

Una vez que el usuario ha capturado todas las esquinas de la estancia tiene la posibilidad de generar el plano de ésta. Para ello solo debe pulsar la opción de "Crear Plano".
Este proceso consta de tres partes que se explican detalladamente a continuación.

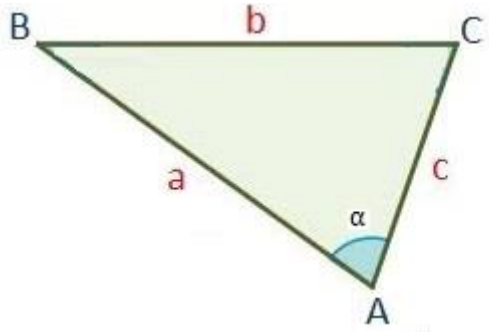
Escalado

Durante el proceso de captura de las esquinas se generan las paredes de la estancia automáticamente. Estas paredes internamente siempre están presentes, aunque por pantalla el usuario tan sólo pueda ver aquellas que son visibles dependiendo de la orientación de la tablet.

Es por ello que para generar el plano como tal tan sólo es necesario aplicar un factor de escalado a cada una de las esquinas que se obtiene relacionando las dimensiones de la pantalla. Este factor es diferente en cada dispositivo, pero se aplica directamente sobre los datos almacenados de la estancia, esto provoca que las paredes se redimensionen automáticamente, permitiendo que el plano se muestre al completo por pantalla.
Cabe destacar que el plano no aparece centrado, ya que en esta pantalla se da la opción al usuario de cargar más planos (que se hubieran almacenado previamente) para poder conformar, si así lo desea, el plano de la vivienda al completo.

Cálculo de medidas de las paredes

Para realizar el cálculo de las medidas de las paredes es necesario utilizar los datos referentes al azimut que se almacenan antes de capturar cada una de las esquinas de la estancia. El cálculo de cada medida se representa en la Figura 22.



A == Usuario
B == Esquina 1
C == Esquina 2
a == Distancia a la Esquina B
c == Distancia a la esquina C
α = Diferencia entre azimuts
b == Medida de la pared

Figura 22. Representación cálculo de las medidas

Las distancias a y c son en realidad las **coordenadas z** del dispositivo de las esquinas B y C respectivamente, que las tenemos almacenadas desde el momento que se capturan las esquinas.
Por otro lado, es necesario obtener el valor del ángulo α, para ello se emplean los datos de los azimuts guardados justo antes de almacenar los datos de cada esquina (ver Figura 8).

α = azimut2 – azimut1 (Para las esquinas 2 y 1)

Una vez que tenemos α, a y b podemos calcular la medida de la pared (b):

$$b = \sqrt{a^2 + c^2 - 2\,a \cdot b \cdot \cos\alpha}$$

El resto de las paredes, se calculan de forma análoga, utilizando en cada caso las esquinas que definen la pared, junto a sus coordenadas z del dispositivo y el valor de α correspondiente.

Optimización

El último paso antes de mostrar al usuario el plano generado con las medidas es una **optimización**. Esta optimización se encarga de corregir posibles errores a la hora de capturar las esquinas, mal calibrado de los sensores, etc.

Para la realización de los cálculos se utiliza la librería científica de Python, SciPy. En concreto la clase optimize. Scipy permite definir una función “error” a optimizar, añadir restricciones, condiciones de salida, etc. En general, una librería muy completa y que cumplía perfectamente con el objetivo actual del proyecto.

Sin embargo, SciPy utiliza código compilado en C, esto provoca que la compatibilidad con los dispositivos móviles se vea gravemente comprometida. Es posible ayudarse de Fortran para compilar la librería en el propio dispositivo, pero se decide tomar la siguiente opción: utilizar un servidor con Django, realizar los cálculos de la optimización, devolver las nuevas coordenadas y generar el plano en ese momento.

Django es un framework de desarrollo web de código abierto, escrito en Python. Su configuración es relativamente sencilla y permite crear aplicaciones web sencillas en poco tiempo. Esto juntado con el hecho de que para este proyecto no es necesaria una aplicación web si no simplemente utilizar el servidor como motor de cálculo, facilita mucho su configuración.
En el momento que el usuario pulsa la opción de "Generar Plano", se envía una petición al servidor, éste la procesa, realiza el proceso que se detalla a continuación y devuelve a la aplicación los datos optimizados.

El método optimize de SciPy, permite añadir restricciones (constraints) en el proceso de optimización. Esto permite mantener una coherencia en los datos iniciales generados por la aplicación dentro de un margen de error.
Se decide que la función a optimizar controle que los ángulos existentes entre las paredes de la estancia se aproximen a 90°, para ello se modificarán las **coordenadas absolutas** de las esquinas. Por lo tanto, es un proceso de **minimización**.
Si bien es cierto que en algunos casos esta optimización es contraproducente, estimaciones realizadas sugieren que el número de casos en los que este proceso generará mejores resultados será mucho mayor que en los que los empeorará.

Función a minimizar:

$$\text{Error} = \sqrt{\sum_{i=0}^{numeroEsquinas} (angulo[i]^2)}$$

Se fijan las siguientes restricciones:

1. La optimización se considera válida desde el instante en el que el **error** entre las coordenadas generadas en la iteración i y las coordenadas generadas en la iteración i-1 sea menor o igual que **0.001**
2. Los **ángulos optimizados** no pueden tener más de un error del **10%** respecto a los **ángulos iniciales**.
3. Las **medidas** de las paredes **optimizadas** no pueden tener más de un error del **20%** respecto a las **medidas iniciales**.

Una vez establecidos todos los parámetros de la función optimize se comienza el proceso de optimización. Cabe destacar que la duración del proceso se reduce al realizarla a través de un servidor. Las pruebas realizadas en servidores locales alternaban valores entre 5 y 10 segundos. En el estado actual de la aplicación la generación del plano (escalado, cálculo de medidas y optimización) es prácticamente instantánea.

Se plantea un escenario como ejemplo, en el cual se genera el plano de una estancia con cuatro paredes.

De este modo, se obtienen las coordenadas optimizadas (Figura 24) y se pueden apreciar las diferencias sobre las coordenadas iniciales (Figura 23):

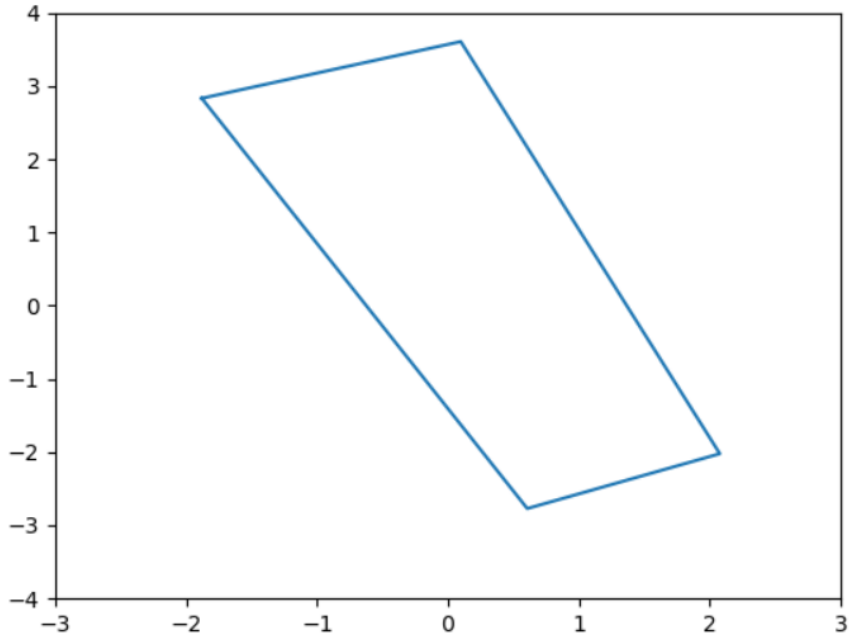


Figura 23. Plano Inicial

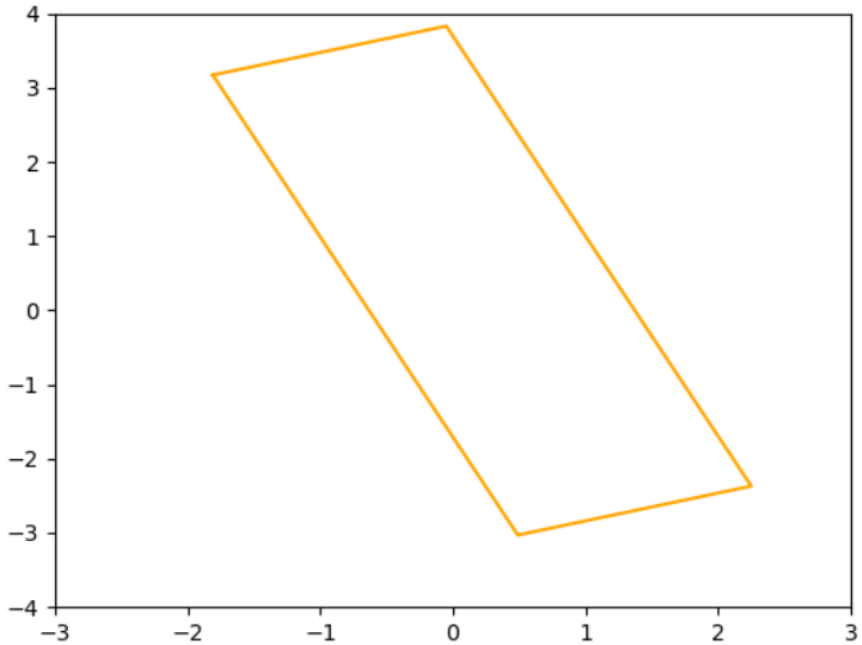


Figura 24. Plano Optimizado

Se exponen los datos iniciales del supuesto junto a los datos generados y sus correspondientes errores, mostrando en todos los casos si se han cumplido las restricciones fijadas en el proceso de optimización.

Medidas de las paredes (Error máximo 20%):

	PARED 1	PARED 2	PARED 3	PARED 4
INICIAL (m)	2.1253	5.9658	1.6507	6.1292
OPTIMIZADA (m)	1.8796	6.6128	1.8796	6.6128
ERROR (%)	11.56	9.78	12.17	7.31

Figura 25. Resultados optimización – Medidas paredes

Ángulos (Error máximo 10%):

	ÁNGULO 1	ÁNGULO 2	ÁNGULO 3	ÁNGULO 4
INICIAL (°)	87.4471	87.9875	82.3892	86.9545
OPTIMIZADO (°)	89.7971	89.7971	89.7971	89.7971
ERROR (%)	2.61	2.01	8.24	3.16

Figura 26. Resultados optimización - Ángulos

Como se puede apreciar en las Figuras 25 y 26, todos los resultados obtenidos, tanto las medidas de las paredes como los ángulos, se encuentran dentro de los valores del error indicados al inicio del proceso.

En la Figura 27 se comparan gráficamente los resultados obtenido, combinando las Figuras 23 y 24.

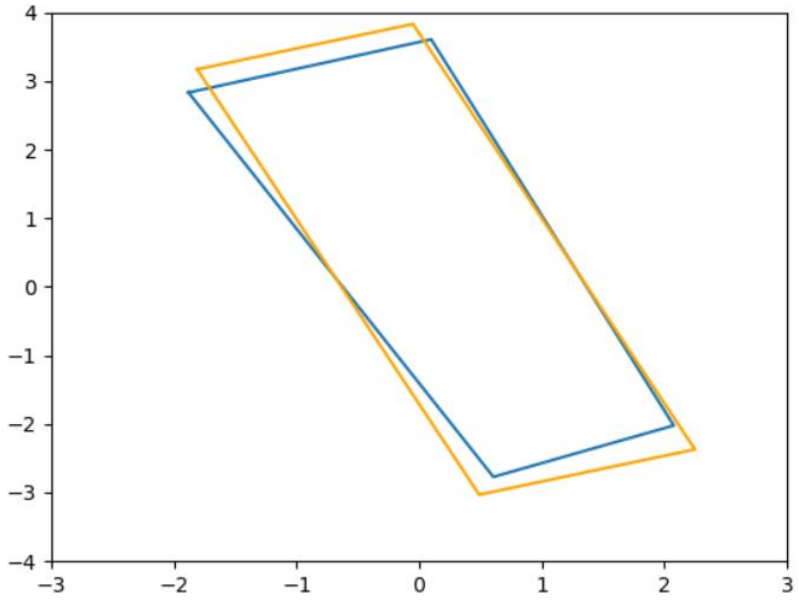


Figura 27. Comparación entre planos

4. Conclusiones y líneas futuras

4.1 Conclusiones

El primer proyecto desarrollado se realiza sobre una aplicación consolidada en el mercado y con un gran número de usuarios.

La mayor parte de los problemas solucionados y de las nuevas funcionalidades añadidas se inspiran en comentarios que de un modo u otro los usuarios de la aplicación han hecho llegar a sus desarrolladores.

Todos los sistemas operativos están en constante evolución y es imprescindible que todas las empresas conozcan y dediquen recursos humanos y materiales a los procesos de actualización y renovación de sus aplicaciones.

Por otra parte, el segundo proyecto nace debido a la demanda de reducción de costes en un determinado sector industrial. Se intentan utilizar métodos de programación avanzados y acordes al mundo actual, en el que el mundo de la Inteligencia Artificial y la Realidad Aumentada toman cada vez más fuerza.

4.2 Líneas futuras

Son diversas las mejoras que se pueden introducir en ambas aplicaciones. En este apartado se van a exponer varias de las que se dejan abiertas como futuras líneas de trabajo.

Aplicación CE3X

La compatibilidad actual de la aplicación actualizada en el primer proyecto con Android 8 es prácticamente del 100%. No obstante, varias funcionalidades están suspendidas debido a que la forma de implementarlas a sufrido demasiados cambios lo que ha provocado que su rediseño no tenga cabida dentro de este proyecto.

Entre ellas se encuentran: opción de editar perfil de usuario, notificaciones al registrar proyectos nuevos, optimización de los cálculos de las medidas de mejora, etc.

Aplicación para el levantamiento de planos

El segundo proyecto se ha diseñado y desarrollado desde 0. Es por ello que parte del código realizado necesita someterse un proceso de refactorización. Principalmente para facilitar futuras actualizaciones o cambios en las funcionalidades.

Por otro lado, parece necesario dedicar tiempo a mejorar la estructura gráfica y la usabilidad de la aplicación (menú principal, más opciones de edición, etc).

5. Bibliografía

SOFTWARE

- **Kivy:** https://kivy.org/#home
- **Buildozer:** http://buildozer.readthedocs.io/en/latest/index.html

IMÁGENES

- **Figura 4:** Mathworks. Capturing Azimuth, Pitch and Roll Example. Obtenido de https://www.mathworks.com/matlabcentral/mlc-downloads/downloads/submissions/40876/versions/8/previews/sensorgroup/Examples/html/CapturingAzimuthRollPitchExample.html?access_key=
- **Figura 5 y 11**: Carrillo R. (2014). El Factor de Multiplicación de la Distancia Focal. Obtenido de http://todo-fotografia.com/2014/el-factor-de-multiplicacion-de-la-distancia-focal/
- **Figura 6:** Obtenido de https://www.freepik.es/fotos-premium/manos-sosteniendo-simulacros-de-tablet-pc-en-la-mesa-de-madera_2133004.htm
- **Figura 10:** Arias E. (2013). Triángulo Rectángulo Isósceles. Obtenido de https://matematicasmodernas.com/triangulo-rectangulo-isosceles/

CÁLCULOS

- **Cálculo de matrices de rotación de los tres ejes en 3D:** Basado en https://www.taringa.net/posts/ciencia-educacion/19170197/Matrices-de-Rotacion.html