Integración de un editor de expresiones matemáticas en un entorno educativo basado en dispositivos móviles

UPC - FIB

Diego Lao Tebar
Ponente: Cristina Gómez Seoane
Director: Juan Lao Tebar
Profesor de GEP: Joan Carles Gil Martin
Empresa: Maths for More S.L.
Grado en Ingeniería informática
Especialidad: Ingeniería del Software

05-07-2019

Resumen

Existen diferentes herramientas que ayudan a crear libros, presentaciones o memorias de proyectos. Sin embargo, pocas ayudan en la tarea de creación de fórmulas matemáticas.

Maths for More S.L. junto a su producto *MathType* proporciona una herramienta para la creación de fórmulas matemáticas haciendo uso de estándares muy populares en el mercado como *MathML* o LaTeX y añadiendo facilidad de uso para su edición.

Los últimos años la compañía ha ido expandiéndose a plataformas como la suite de *Microsoft Office* o la plataforma de aprendizaje online *Moodle*. Por ello, han seguido buscando nuevos ecosistemas donde expandirse y, después de un estudio de mercado, se ha detectado una necesidad de expansión hacia la plataforma *iOS*.

Este proyecto es la expansión de *MathType* en dicha plataforma, integrando el producto en este nuevo ecosistema, mejorando la experiencia de uso y sobretodo proponiendo una arquitectura pensada desde el punto de vista de la mantenibilidad y reusabilidad.

Resum

Existeixen diferents eines que ajuden a crear llibres, presentacions o memòries de projectes. Però, poques ajuden a la tasca de creació de fòrmules matemàtiques.

Maths for More S.L. junt al seu producte *MathType* proporciona una eina per a la creació de fòrmules matemàtiques fent ús d'estàndards molt populars dins del mercat com *MathML* o LaTeX i afegint facilitats d'ús per a la seva edició.

Els últims anys, la companyia ha anat expandint-se a plataformes com la suite de *Microsoft Office* o la plataforma d'aprenentatje *Moodle*. Degut a això, han anat cercant nous ecosistemes on integrar-se i, després d'un estudi de mercat, s'ha detectat una necessitat d'expansió cap a la plataforma *iOS*.

Aquest projecte es la expansió de *MathType* dins d'aquesta plataforma, integrant el producte, millorant l'experiència d'ús i sobretot proposant una arquitectura raonada des del punt de vista de la mantenibilitat i reusabilitat.

Abstract

Nowadays, there are a lot of tools to help people in tasks like the writing process of books, presentations or theses. However, only some of these tools help to write mathematical formulas.

Maths for More S.L. with its product *MathType* provides a tool for the creation of mathematical formulas using popular standards as *MathML* or LaTeX and adding facilities of use.

The last years the company has been expanding its business to third platforms like *Microsoft Office* or the learning management platform *Moodle*. Due to that, they have been searching new ecosystems until a necessity of this kind of software in *iOS* has been found.

This project is the expansion of *MathType* to *iOS*. Integrating the product in this new ecosystem improving user experience and specially considering the architecture from the point of view of maintainability and reusability.

Agradecimientos

Ya está, después de un intenso cuatrimestre he terminado, presento mi primer *TFG*.

Estoy bastante contento con el resultado, pese a que me hubiera gustado más haber dedicado un cuatrimestre o dos de manera exclusiva en un tema relacionado con el motociclismo eléctrico. Sin embargo, me lo he pasado bien.

Este capítulo está para agradecer a aquellas personas que me ayudaron a hacerlo. Primero de todo mi familia más directa: mi hermano y director de proyecto, Juan, por proponerme realizar este proyecto en la empresa *Maths for More S.L.* y criticar mi trabajo; mi madre María, por apoyarme siempre y ayudarme económicamente desde pequeño en los distintos proyectos que he ido realizando por muy bobos que fueran; y mi padre Juan (sí, otro Juan), por inculcarme la curiosidad y estudiar o trabajar en algo sólo por el mero gozo, sin todo ello no sería nada.

Por otra parte, también me gustaría agradecer a Ana, mi novia, la tarea de aguantarme y leer todos los capítulos que le mandé de las primeras versiones del proyecto para después criticarlos, si eso no es amor *que baje Dios y lo vea*. Además, reconozco que no se me da del todo bien escribir, pero gracias a ella este trabajo puede ser mínimamente leído, aunque sea el principio.

Finalmente, también agradecer a mi tutora Cristina el trabajo de leerse todas mis entregas y contestar hasta en días festivos con alguna crítica y propuesta de mejora. En lo personal, es una de las mejores profesoras que se pueden encontrar en la *FIB*.

Sé que me dejo a familia y amigos, pero si sigo así tengo que darle las gracias hasta al autobusero de la línea *H8*.

Y para terminar, y a riesgo de tacharme de ególatra, me doy las gracias a mí, so cabezota. No cambies.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción y contexto

Este proyecto es un *Trabajo de Final de Grado (TFG)* de la especialidad de Ingeniería del Software de la Facultat d'Informàtica de Barcelona (Universitat Politècnica de Barcelona) en modalidad B. Concretamente se trabajará en la empresa *Maths for More S.L.* .

Maths for More S.L. es una empresa dentro del sector de las matemáticas y la educación. Desarrollan productos bajo el nombre de *WIRIS* y *MathType* para grandes plataformas *LMS (Learning Management System)* como *Moodle* y distintos editores web.[31]

Actualmente, uno de sus productos principales es *MathType*, un editor de fórmulas matemáticas dirigido a profesores, alumnos y editores generales de documentos matemáticos que permite la edición de fórmulas tanto usando una barra de herramientas como de forma manuscrita (*handwriting*).

MathType destaca por su integración en plataformas de terceros muy populares como *Google Docs*[23], *Moodle*[37] o *Microsoft Office*[32] y también por la facilidad de integración en nuevas plataformas. Debido a esto, se ha decidido empezar su expansión a otros sistemas operativos de escritorio y móvil. Este TFG pretende ser la expansión de *MathType* a iOS y lo hará bajo el nombre de *MathType for iOS*.

1.1. Actores implicados

Este proyecto tiene distintos actores que se ven afectados por su desarrollo. En las siguientes secciones se detallarán y se explicarán.

1.1.1. Maths for More S.L.

Empresa donde se realiza el proyecto. El proyecto afecta a toda la empresa debido a que es una inversión por parte de ésta para expandir su producto en el mercado. Por ello, también se ven afectados todos los departamentos como el comercial, el de soporte u otros departamentos técnicos de la empresa.

1.1.2. Equipo de desarrollo y evaluador del TFG

Es el equipo que tendrá en cuenta que el proyecto será evaluado como TFG, aportarán diferentes puntos de vista y lo desarrollarán.

Por un lado estará Diego Lao Tebar, el estudiante responsable que trabaja en *Maths for More S.L.* y que realizará el proyecto tomando las decisiones para terminarlo de la mejor forma posible. Y por otro lado están: Juan Lao Tebar, el director del proyecto; y Cristina Gómez Seoane, la ponente del proyecto. Su cometido es el de guiar y supervisar el TFG.

1.1.3. Los usuarios

Los usuarios a los que va destinado *MathType* son principalmente alumnos, profesores y editores que trabajan con fórmulas matemáticas y necesitan una herramienta para editar y visualizar dichas fórmulas matemáticas. Este proyecto les ofrece esas funcionalidades desde dispositivos iOS.

1.1.4. Los usuarios clientes de *Maths for More S.L.*

Ya hay usuarios que son clientes que utilizan el editor de fórmulas matemáticas *MathType*. Este proyecto complementará la versión de escritorio o web y les permitirá el uso de *MathType* en el sistema operativo iOS, pudiendo realizar el trabajo que hacen desde dispositivos móviles iOS.

1.1.5. Usuarios "validadores"

Los usuarios "validadores" son aquellos usuarios que se escogen para realizarles pruebas de validación del producto. Reportan posibles problemas del producto para así poder revisar cambios que ayuden a mejorar.

1.1.6. Otros stakeholders

Existen otros actores implicados como por ejemplo programadores u otros sistemas de la competencia iguales o parecidos. El código de las integraciones

de *MathType* es de código abierto, por lo que puede ser revisado por éstos actores con el fin de aprender o copiar el código.

1.2. Estado del arte

Dentro del mundo de la edición de fórmulas matemáticas, existen diferentes estándares para la representación de fórmulas. Los principales son *MathML*[7] y LaTeX[36]. Estos estándares han provocado que los sistemas de edición de fórmulas sean heterogéneos y que no puedan comunicarse entre sí de forma sencilla, ya que cada uno implementa su propio estándar o elige uno ya existente de forma arbitraria. Sin embargo, han facilitado que dos editores que hayan implementado el mismo estándar puedan comunicarse entre sí.

Este problema de heterogeneidad ha provocado que el exportar fórmulas sea también complicado debido a que hay que responder a la pregunta de "¿En qué formato exportar?". La solución por parte de algunos de los editores ha sido no dejar exportar, exportar a diferentes estándares o simplemente exportar en un formato genérico aceptado por casi todos, una imagen.

Insertar en un editor de texto una imagen es algo trivial hoy en día. Por lo que es una de las soluciones que mejor funcionan en aquellos editores de fórmulas que buscan exportar sus fórmulas a otros programas, con la seguridad de que podrán ser utilizadas.

A parte del problema de heterogeneidad comentado, ha existido otro problema crítico, la creación de un sistema simple para editar las fórmulas. Los estándares de representación de fórmulas matemáticas son muy extensos y esto se traduce en que una interfaz de usuario no puede enseñar tanta información a la vez. Por ello, muchos de los editores existentes en el mercado tienen interfaces poco usables y además muy diferentes entre sí.

Actualmente existe un programa software llamado *MathType* que intenta solucionar estos problemas mediante el uso del estándar *MathML* y el soporte del estándar LaTeX. De esta forma puede ser compatible con la mayor cantidad de productos del mercado.

Además, ha invertido recursos en la creación de distintas formas de creación de fórmulas, ya sea de forma manuscrita o con ayuda de una barra de herramientas. Por lo que la comunidad ha agradecido este avance.

Finalmente, ha creado un ecosistema de aplicaciones de escritorio y web donde se utilizan imágenes como elementos genéricos para exportar sus fórmulas.

1.2.1. Comparativa de productos similares

Por lo tanto, *MathType* no es el único software que permite crear y editar fórmulas matemáticas, pero sí que es uno de los más completos y hay motivos que justifican la creación de un software que extienda de él.

Primero de todo tenemos el ecosistema de *MathType*. *MathType* ofrece ya dos versiones de su editor: una versión desktop y una versión web, por lo cual crear una versión para la plataforma iOS amplía el ecosistema ya existente.

El segundo motivo es que no existen aplicaciones en iOS que permitan crear y editar fórmulas manuscritas.

El tercer motivo es que no existen aplicaciones pensadas para crear y editar fórmulas y exportar a otras aplicaciones la imagen resultante con facilidad. Esto limita al usuario si quiere utilizar un editor en particular y complementarlo con un software externo.

El cuarto motivo es que en general los editores de ecuaciones existentes en iOS no son cómodos y/o no aprovechan características del sistema y del dispositivo. No están pensados para dividir la pantalla en dos y utilizar dos aplicaciones a la vez. Por otra parte las funcionalidades táctiles que podrían ser útiles, como insertar fórmulas manuscritas, son inexistentes.

De los motivos explicados se han extraído características que se creen relevantes para los usuarios y se ha realizado una tabla comparativa entre los diferentes programas existentes en el mercado. Además, se muestran las características que debería cumplir *MathType for iOS* una vez terminado el proyecto.[31][35][26][54][4]

Tabla 1.1: Comparativa de productos.

	MathType for iOS	MathType	Microsoft Word para iOS	iWork	MathPad	MathMagic
Creación y edición de fórmulas	✓	✓	✓(Sólo iPad)	✓	✓	✓
Permite importar de otras aplicaciones las fórmulas de las imágenes creadas	✓	✓	✗	✗	✗	✗
Permite exportar a otras aplicaciones las imágenes de las fórmulas creadas	✓	✓	✓	✓	✓	✗
Soporte del formato MathML	✓	✓	✗	✓	✗	✓
Soporte del formato LaTeX	✓	✓	✓	✓	✗	✓
Handwriting	✓	✓	✗	✗	✗	✗
Interfaz para edición	✓	✓	✓	✗	✓	✓
Soporte para iOS	✓	✗	✓	✓	✓(Sólo iPad)	✗
Pensada para funcionar en modo pantalla dividida en iOS	✓	✗	✓	✓	✗	✗

Microsoft Word y *iWork* pese a no ser programas pensados desde un inicio para la inserción de fórmulas matemáticas han ganado buena fama extendiendo sus funcionalidades. Sin embargo, ofrecen herramientas muy simples o a veces incómodas de usar cuando se quiere introducir una fórmula compleja. Todo esto debido a que tienes que navegar por demasiados menús. Pese a ello, se sigue trabajando y en las últimas versiones de *Microsoft Word*[34] se están realizando avances para soportar estándares como LaTeX.

En cuanto a *MathPad*, es un editor de documentos centrado en poder escribir texto con la capacidad de insertar también fórmulas. Funciona realmente bien para crear fórmulas complejas pero no aprovecha las capacidades del sistema como es la multiventana. Además está muy limitado debido a que está pensado para trabajar en él y no para ser el complemento de otros programas, por lo que no cuenta con el soporte de formatos como *MathML* o LaTeX.

Finalmente, *MathMagic* es el programa más parecido a *MathType* en su versión de escritorio. Debido a esto tiene grandes contras como el de no tener soporte para plataformas móviles y estar restringido a ser utilizado como complemento de programas específicos como *Microsoft Word* o *Adobe InDe-*

sign y no como identidad propia o complemento genérico.

1.3. Formulación del problema

1.3.1. Problema

Actualmente hay muchos usuarios en iOS que editan documentos desde su iPad -y según las estadísticas de uso internas de *MathType* seguirá creciendo. Sin embargo, cuando necesitan insertar fórmulas matemáticas es difícil crearlas y editarlas debido a que no hay editores matemáticos para el programa específico o tienes que usar editores alternativos.

En cuanto a cambiar de editor, hay muchas empresas que trabajan con herramientas como *Microsoft Word*[35] y no están dispuestas a cambiar de software solamente porque en uno se pueda o sea más fácil insertar fórmulas matemáticas. Entonces acaban por recurrir a tener documentos en distintos editores, separados entre sí o sencillamente no insertar todo lo que les gustaría insertar.

Además, es habitual que los editores de fórmulas matemáticas presenten interfaces incómodas que no se adaptan a la pantalla táctil de estos dispositivos. Tampoco existe la posibilidad de trabajar en modo multiventana (también se le puede llamar "ventana partida") debido a las mismas carencias en la interfaz de usuario.

1.3.2. Objetivos

Debido a esto, este proyecto busca crear una aplicación para el sistema operativo iOS. Esta aplicación debe permitir la creación y edición de fórmulas, que permita una vez creadas poder exportar la imagen a otra aplicación de edición de texto, como por ejemplo *Microsoft Office*.

Además, debe tener una interfaz adaptada a las características que ofrece tanto el sistema operativo iOS como las capacidades del dispositivo. Es decir, hacer uso de la capacidad multiventana y tener una interfaz usable teniendo en cuenta que es un dispositivo táctil y con dimensiones más reducidas que un monitor de ordenador.

Capítulo 2

Alcance

Este proyecto busca extender el ecosistema de *MathType* al sistema operativo iOS, para su uso principal en iPad, pero dando soporte también a iPhone. De esta manera, los usuarios contarán con una herramienta para poder trabajar en la edición de fórmulas matemáticas desde dispositivos iOS.

El resultado del proyecto debe ser una aplicación iOS que contenga todas las funcionalidades de *MathType* Web, compartiendo su código por temas de mantenimiento y aplicándole los elementos necesarios para extender sus funcionalidades y así ser más cómodo y útil en iOS.

Los elementos que extenderán dichas funcionalidades de *MathType* son elementos de pantalla para mejorar el copia y pega de las fórmulas, ajustes de imagen y herramientas, una sección de "fórmulas recientes", la posibilidad de exportar e importar imágenes de las fórmulas generadas mediante el uso de Drag&Drop y por último, extender el estilo del editor para poder ser utilizado en pantalla dividida junto a otra aplicación. Esta última característica, solamente podrá ser utilizada en dispositivos iPad debido a que son los únicos dispositivos con la funcionalidad de multiventana.

2.1. Posibles riesgos y soluciones

En el transcurso del desarrollo del proyecto pueden surgir diferentes problemas. Los más relevantes son los siguientes:

1. **Tiempo insuficiente.** El tiempo del que se dispone para la realización es limitado. Durante la planificación del proyecto se tendrá en cuenta de manera constante. Una de las soluciones es hacer un sistema modular donde una vez realizada la integración de *MathType Web* en iOS cada funcionalidad sea un módulo independiente. De esta forma, en caso de

no tener el tiempo suficiente, es posible no implementar ciertos módulos y aún así todo el proyecto puede seguir en funcionamiento.

2. **Problemas al conectar diferentes lenguajes de programación entre sí.** Este proyecto hace uso de diferentes lenguajes de programación que pueden llegar a realizar tareas complejas, pero se podrían ver limitados en la comunicación entre sí. Una solución es desacoplar los diferentes módulos de la aplicación para que así la comunicación sea lo más simple posible. Cada módulo deberá trabajar de forma independiente y comunicar únicamente los resultados.

3. **Ignorancia en la temática de diseño.** Durante la carrera no hay una gran enseñanza del diseño de interfaces y por lo tanto podría provocar que la aplicación no fuera tan fácil de usar como se quiere o que se gaste demasiado tiempo en ello. Una solución es aprovechar los recursos humanos de la empresa *Maths for More S.L.* , ya que existe un departamento específico de documentación y diseño.

4. **Aprendizaje de nuevas tecnologías.** No se tiene una gran experiencia en el lenguaje de programación. En caso de tener problemas, existen muchos recursos en Internet y habrá que invertir tiempo del proyecto para investigar. Se tendrá entonces en cuenta este tiempo extra durante la planificación temporal del proyecto.

2.2. Metodología y rigor

Durante la carrera se han explicado diferentes metodologías para afrontar proyectos, entre las cuales las más destacadas han sido *Waterfall*[47] y *Agile*[48].

Como se nos ha explicado en asignaturas como GPS (Gestión de Proyectos Software), para saber cuál escoger hace falta observar la naturaleza del proyecto. Éste es un proyecto con los requisitos bien definidos pero que puede presentar muchas variaciones en el tiempo de ejecución debido al uso de tecnologías nuevas para el desarrollador y también el cambio de interfaz a una interfaz móvil.

Además y debido a la limitación de recursos al haber un sólo desarrollador, se necesita tener el sistema funcionando lo antes posible para recibir *feedback* y cambiar las cosas que sean necesarias.

Por estos motivos, y siguiendo los principios del *Manifiesto Agile*[2] de dar más valor al software funcionando respecto a la documentación exhaustiva y

la respuesta ante el cambio sobre seguir el plan, se ha decidido escoger *Agile* como la metodología para este proyecto.

2.2.1. Método de trabajo

Una vez se ha decidido la metodología de trabajo hace falta especificar qué método se acabará utilizando.

Existen diferentes implementaciones de la metodología Agile, pero en este proyecto se utilizará *SCRUM*[42] ya que es el que utiliza *MathType* y se cree conveniente que el proyecto se sincronice desde un principio con la línea general de productos de la empresa.

SCRUM lo que propone a grandes rasgos es definir bloques temporales llamados *sprints* o iteraciones donde al final de cada bloque hay una entrega parcial. Una entrega parcial sería una mejora del producto que ayuda a estar más cerca del objetivo del proyecto, pero que no tiene por qué conseguir que se cumpla el objetivo en sólo esa entrega.

SCRUM[43] permite definir sprints máximos de 4 semanas, que es el que utiliza MathType y por lo tanto el que usará este proyecto. Dentro de cada sprint se puede dividir principalmente en las siguientes fases:

- **Planificación del sprint.** Se realiza al principio de cada sprint.
  - Selección de los requisitos a cumplir en el sprint.
  - Planificación de las tareas a realizar para alcanzar los requisitos propuestos.
- **Daily Meetings.** Reuniones que se hacen idealmente cada día, pero que en el caso de este proyecto se hará de forma semanal mediante email para mantener informado tanto al ponente como al director del proyecto. Y, debido al carácter semanal, durante el transcurso del proyecto se llamarán a este tipo de reuniones *Weekly Meetings*.
- **Refinamiento.** Durante todo el sprint se realiza un análisis del tiempo y esfuerzo dedicado a cada una de las tareas y se vuelven a aclarar los requisitos si hiciera falta.
- **Revisión del sprint.** Reunión al final del sprint donde se muestra a toda la empresa el desarrollo que se ha llevado a cabo durante la iteración y a poder ser va acompañado de una demostración en vivo.

- **Retrospectiva.** Se realiza al final del sprint y se analiza qué se ha hecho mal, qué se ha hecho bien y qué inconvenientes se han encontrado que no han permitido avanzar como se había planificado en un inicio.

2.2.2. Herramientas de seguimiento

Existen muchas herramientas de seguimiento para un proyecto, para este en específico se han escogido varias dependiendo del contexto.

A nivel de código se realizará un seguimiento utilizando *Git*[9], una tecnología que permite separar el código en las llamadas "ramas" para así mantener los desarrollos nuevos independientes de la parte estable del sistema. Además permite llevar un control de los cambios del código.

A nivel de planificación de tareas se hará uso de *Trello*[52], una plataforma web que permite la creación de tareas y llevar un control del estado de las mismas.

Finalmente, a nivel temporal se utilizará la herramienta *Microsoft Project*[33], que permite llevar un seguimiento de las tareas en las diferentes fases del proyecto junto a los recursos utilizados. También permite la creación de artefactos como pueden ser los *diagramas de Gantt*.

2.2.3. Método de validación

SCRUM plantea el uso de tests como método de validación del código generado[44][45]. Cada incremento del código que se genera por cada tarea tiene su conjunto de tests asociado. Estos tests tendrán que pasar correctamente antes de la entrega final. De esta manera es posible validar que los requisitos se cumplen y que no se ha dado el caso de poder haber hecho que otro requisito que antes se cumplía ahora no se cumpliera.

Para el caso de las validaciones que no puedan llevarse a cabo con tests, como por ejemplo "que la interfaz sea bonita", se realizará una validación durante las reuniones finales de revisión del sprint y retrospectiva.

Y finalmente, se informará de manera semanal en los *Weekly Meetings* al director y al ponente del proyecto TFG para que opinen sobre el estado del proyecto. No obstante, también se realizarán reuniones espontáneas en caso de tener dudas que necesiten ser resueltas lo antes posible.

Capítulo 3

Análisis de requisitos

El análisis de requisitos es una de las partes más importantes de un proyecto software. Esto es debido a su impacto directo en la elección de las características que debe tener el sistema a construir y que con las cuales se cumplirán dichos requisitos, alcanzando así los objetivos propuestos.

Existen diferentes definiciones de qué es un *requisito software*, como por ejemplo la que contiene el *Glosario IEEE de Términos de Ingeniería del Software* y la definición que se encuentra en *La Guía SWEBOK*.

El *Glosario IEEE de Términos de Ingeniería del Software*[12] define *requisito software* como:

- Condición o capacidad requerida por un usuario para solucionar un problema o conseguir un objetivo.
- Condición o capacidad que tiene que ser cumplida por un sistema o un component de un sistema para satisfacer un contrato, estándar, especificación o cualquier otro documento formal.
- Representación documentada sobre una condición o capacidad como 1 o 2.

Por otra parte, La Guía SWEBOK define requisito software como:

> Un requisito software es la propiedad que un software desarrollado o adaptado debe tener para resolver un problema concreto.[19]

Además, los requisitos se pueden categorizar en requisitos funcionales y en requisitos no funcionales.

La Guía SWEBOK define los requisitos funcionales como:

> Un requisito funcional especifica una función que un sistema o componente de un sistema debe ser capaz de llevar a cabo.[20]

Mientras que define los requisitos no funcionales como:

> Los requisitos no funcionales son aquellos que especifican aspectos técnicos que debe incluir el sistema, y que pueden clasificarse en restricciones y calidades.[20]

Entonces, debido a la importancia de dichos requisitos, en las siguientes dos secciones se mostrarán los diferentes requisitos de la aplicación MathType for iOS así como la justificación de cada requisito.

3.1. Requisitos funcionales

En esta sección se expondrán los diferentes requisitos funcionales que requiere *MathType for iOS*. En cada uno de los requisitos se les dará una enumeración junto a un nombre. Posteriormente se hará una descripción detallada del caso de uso junto a una justificación del porqué es necesario dicho caso de uso. Finalmente, se explicará el criterio de satisfacción que se debe cumplir para poder dar por satisfecho el caso de uso.

Requisito funcional 001: Creación/Edición de imágenes de fórmulas matemáticas.
Descripción: *MathType for iOS* permitirá la creación y edición de fórmulas matemáticas en el sistema operativo iOS.
Justificación: Los usuarios usan *MathType for iOS* para crear y/o editar fórmulas matemáticas, por lo que poder utilizar MathType dentro de la aplicación pueden hacerlo.
Criterio de satisfacción: Los usuarios pueden utilizar todas las funcionalidades que ofrece MathType para webs.

Requisito funcional 002: Exportar fórmulas desde *MathType for iOS* a otra aplicación.
Descripción: *MathType for iOS* podrá exportar las fórmulas creadas como imágenes a otras aplicaciones que permitan insertar imágenes.
Justificación: Los usuarios de *MathType for iOS* quieren crear fórmulas para insertarlas en otras aplicaciones. Por lo tanto, este caso de uso les permite exportar las formulas que han creado.
Criterio de satisfacción: El usuario puede exportar de *MathType for iOS* a una aplicación diferente que acepta imágenes de terceras aplicaciones.

Requisito funcional 003: Importar desde otra aplicación a *MathType for iOS*.
Descripción: *MathType for iOS* permitirá importar imágenes con fórmulas de otras aplicaciones para editarlas.
Justificación: Los usuarios de *MathType for iOS* crean fórmulas para insertarlas en otras aplicaciones. Por lo tanto, este requisito permite al usuario importar esas fórmulas de nuevo para así editarlas y no tener que crearla desde cero.
Criterio de satisfacción: El usuario puede importar desde *MathType for iOS* una fórmula desde una aplicación diferente.

Requisito funcional 004: Cambio de formato de imagen [1].
Descripción: *MathType for iOS* permitirá la elección del formato de imagen (PNG o SVG) de las fórmulas creadas.
Justificación: Las diferentes aplicaciones existentes no soportan cualquier tipo de formato de imagen. Permitir elegir entre los dos formatos más utilizados y con mejores características técnicas ayuda al usuario a obtener el mejor resultado posible dependiendo de qué formato de imagen soporte el software al que quiere exportar las imágenes.
Criterio de satisfacción: El usuario puede obtener imágenes ya sea en formato PNG o SVG.

Requisito funcional 005: Detección de fórmula en imagen.
Descripción: *MathType for iOS* permitirá tomar una fotografía con la cámara del móvil o desde la biblioteca de fotos del mismo móvil y reconocer la fórmula para posteriormente poder visualizarla y/o editarla.
Justificación: Muchas veces los usuarios son profesores que utilizan libros físicos de matemáticas donde encuentran fórmulas que les gustaría utilizar en sus papers o en los tests para sus alumnos. Esta funcionalidad les da una facilidad extra a la hora de hacerlo.
Criterio de satisfacción: El usuario puede realizar o seleccionar una imagen con una fórmula no manuscrita para ser procesada y obtener así la fórmula para ser editada en *MathType for iOS*.

Requisito funcional 006: Mostrar fórmulas recientes.
Descripción: *MathType for iOS* permitirá consultar las últimas fórmulas creadas y/o editadas.

[1] El requisito funcional estaba contemplado durante el diseño y la toma de requisitos de la aplicación. Sin embargo, debido a limitaciones del sistema, se ha eliminado. En los capítulos de implementación y planificación se explica con más detalles el porqué de su eliminación.

Justificación: Hay veces que los usuarios quieren editar una fórmula que han creado recientemente pero debido a que la han borrado tienen que volver a crear la fórmula desde cero. Para mejorar la experiencia de uso se pueden mostrar las fórmulas recientes y permitir así editar desde su última edición.
Criterio de satisfacción: El usuario puede crear una fórmula, borrar la fórmula, crear otra nueva y entonces poder seleccionar la fórmula antigua para ser editada de nuevo desde el último momento antes de borrarla.

Requisito funcional 007: Seleccionar fórmula reciente.
Descripción: MathType for iOS permitirá editar una fórmula creada recientemente.
Justificación: El usuario habitualmente quiere volver a editar una fórmula creada recientemente para modificarla y así ahorrar tiempo al no tener que crearla de nuevo desde el principio.
Criterio de satisfacción: El sistema puede enviar notificaciones informativas al usuario.

Requisito funcional 008: Recibir notificaciones informativas.
Descripción: *MathType for iOS* permitirá la creación de diálogos informativos al usuario para explicar errores que hayan podido surgir durante su ejecución.
Justificación: Existen situaciones donde hay que mantener informado al usuario del estado del sistema. Por ejemplo, si ha habido un error.
Criterio de satisfacción: El sistema puede enviar notificaciones informativas al usuario.

Requisito funcional 009: Cambio de toolbar.
Descripción: *MathType for iOS* permitirá escoger qué barra de herramientas se muestra en el editor de entre las ya ofrecidas por *MathType Web*: normal, simple y chemistry.
Justificación: Cada barra de herramientas hace que ciertas herramientas sean más accesibles, dependiendo de qué fórmulas realice el usuario le conviene una u otra para ir más rápido.
Criterio de satisfacción: El usuario puede escoger una barra de herramientas y el sistema es capaz de cambiarla.

Requisito funcional 010: Cambio de idioma.
Descripción: *MathType for iOS* permitirá escoger en qué idioma está la aplicación.
Justificación: El usuario se siente más cómodo usando la lengua que él prefiere o entiende.
Criterio de satisfacción: El usuario puede escoger en qué lengua está el

sistema y el sistema es capaz de adaptarse a dicha elección cambiando los textos al idioma elegido.

3.2. Requisitos no funcionales

Para terminar, detallaremos los requisitos no funcionales de *MathType for iOS*. Para este tipo de definición se ha utilizado una plantilla que separa los diferentes requisitos no funcionales por categorías explicando el nombre del requisito, la categoría que se le asigna dentro de The Volere Requirements Specification Template[41], la descripción detallada de cada requisito, la justificación de por qué es necesario dicho requisito y finalmente la condición de satisfacción que se tiene que cumplir para dar por alcanzado el requisito.

3.2.1. Requisitos de interfaz "Look And Feel"

Requisito no funcional 001: Diseño atractivo
Según Volere: 10a
Descripción: El sistema tiene que ser atractivo para los usuarios.
Justificación: Un diseño que gusta hace crecer el número de usuarios.
Condición de satisfacción: Se toman personas independientes al proyecto y se hace una prueba de la aplicación. Si el 70 % o más de las valoraciones son positivas, se considera que el requisito ha sido satisfecho.

Requisito no funcional 002: Estilo autoritario
Según Volere: 10b
Descripción: El sistema tiene que seguir los estilos de diseño adecuados por los cuales dar una sensación de fiabilidad al ser utilizada.
Justificación: Hace falta obtener un diseño por el cual los usuarios no se sientan obligados a buscar otras aplicaciones debido al mal diseño de esta.
Condición de satisfacción: Se toman personas independientes al proyecto y se hace una prueba de la aplicación. Si el 70 % o más de las valoraciones son positivas, se considera que el requisito ha sido satisfecho.

3.2.2. Requisitos de usabilidad y humanidad

Requisito no funcional 003: Usabilidad
Según Volere: 11a
Descripción: El sistema tiene que ser intuitivo.
Justificación: El sistema tiene que seguir los estándares de diseño a los que los usuarios están acostumbrados con tal de permitir que sea fácil para ellos

utilizar el sistema.
Condición de satisfacción: Se toman personas independientes al proyecto y se hace una prueba de la aplicación. Si el 70 % o más de las valoraciones son positivas, se considera que el requisito ha sido satisfecho.

Requisito no funcional 004: Lengua
Según Volere: 11b
Descripción: El sistema tiene que permitir visualizar la información de la aplicación en el idioma del dispositivo. Si el idioma no estuviera disponible, por defecto se usa el inglés.
Justificación: La adaptación a diferentes lenguas provoca que más usuarios puedan utilizar el sistema.
Condición de satisfacción: El sistema muestra la información en el idioma del dispositivo.

Requisito no funcional 005: Aprendizaje
Según Volere: 11c
Descripción: El sistema tiene que permitir que los usuarios utilicen el sistema sin tener formación previa.
Justificación: Saber utilizar el sistema sin formación previa provoca que el usuario empiece a utilizar el sistema más rápido y que cometa menos errores.
Condición de satisfacción: Se toman personas independientes al proyecto y se hace una prueba de la aplicación. Si los usuarios pueden crear una fórmula y arrastrarla a otra aplicación en menos de 5min, se considera que el requisito ha sido satisfecho.

3.2.3. Requisitos de rendimiento

Requisito no funcional 006: Velocidad y latencia
Según Volere: 12a
Descripción: El sistema tiene que invertir la menor cantidad de tiempo en la comunicación con los servicios externos.
Justificación: Un gasto menor en la cantidad de tiempo que lleva la comunicación con servicios externos genera una sensación de eficiencia hacia el usuario debido a que el tiempo de respuesta disminuye.
Condición de satisfacción: Se realizan 10 pruebas de escritura (handwriting) en la integración de MathType de la fórmula $x = \frac{-b \pm \sqrt{b^2-4ac}}{2a}$. Si ninguna prueba sobrepasa los 10s, se da por satisfecho el requisito.

Requisito no funcional 007: Disponibilidad
Según Volere: 12d

Descripción: El sistema tiene que estar disponible y funcional el mayor tiempo posible.
Justificación: Mantener un sistema funcionando en todo momento aumenta la confianza del usuario hacia el sistema.
Condición de satisfacción: El sistema en ningún momento puede estar más de 24h inutilizado.

3.2.4. Requisitos operacionales y de entorno

Requisito no funcional 008: Sistemas Adyacentes
Según Volere: 13c
Descripción: El sistema tiene que poder trabajar correctamente con diferentes servicios adyacentes.
Justificación: El sistema integra MathType, software que depende directamente de servicios externos. Si el sistema no logra comunicarse de forma correcta con los servicios externos quedará inutilizado.
Condición de satisfacción: Es posible crear la imagen de una fórmula introducida.

3.2.5. Requisitos de mantenimiento y soporte

Requisito no funcional 009: Mantenimiento
Según Volere: 14a
Descripción: El sistema tiene que poder actualizarse dependiendo del feedback de los usuarios, los bugs encontrados o posibles nuevas funcionalidades.
Justificación: El sistema integra MathType, un software en contínuo desarrollo con actualizaciones mensuales. Además, el sistema operativa para el que se está desarrollando, iOS, también está en contínuo desarrollo.
Esto provoca que el sistema tenga que actualizarse cada cierto tiempo para adaptarse a los nuevos cambios de sus dependencias.
Condición de satisfacción: Se pueden implementar nuevas features o arreglar bugs encontrados en un tiempo igual o menos a un mes.
Además, debe contar con un build y tests automatizados haciendo uso del software Jenkins.

Requisito no funcional 010: Adaptabilidad
Según Volere: 14c
Descripción: El sistema tiene que poder ser ejecutado y ser 100 % funcional en dispositivos iPad con una versión 12 de iOS como mínimo y ser compilado con Xcode 10 o posterior.

Justificación: La ejecución del sistema en dispositivos con las últimas versiones del sistema operativo iOS aumenta el número máximo de potenciales usuarios.
Condición de satisfacción: La aplicación es compatible y 100 % funcional desde iOS 12.0 hasta la última versión estable publicada.

Capítulo 4

Especificación

En ingeniería del software, se utilizan diferentes artefactos para capturar y llevar la información del proyecto. Estos artefactos ayudan a cumplir los objetivos del proyecto modelando sus diferentes partes, como por ejemplo los requisitos que tiene que cumplir el software, el comportamiento de éste en situaciones específicas o también el cómo interpreta el sistema la realidad.
A continuación se expondrán en las diferentes secciones los artefactos de este trabajo, donde se ha escogido el uso de casos de uso para la especificación de qué acciones tiene que realizar el sistema MathType for iOS.

4.1. Diagrama de los casos de uso

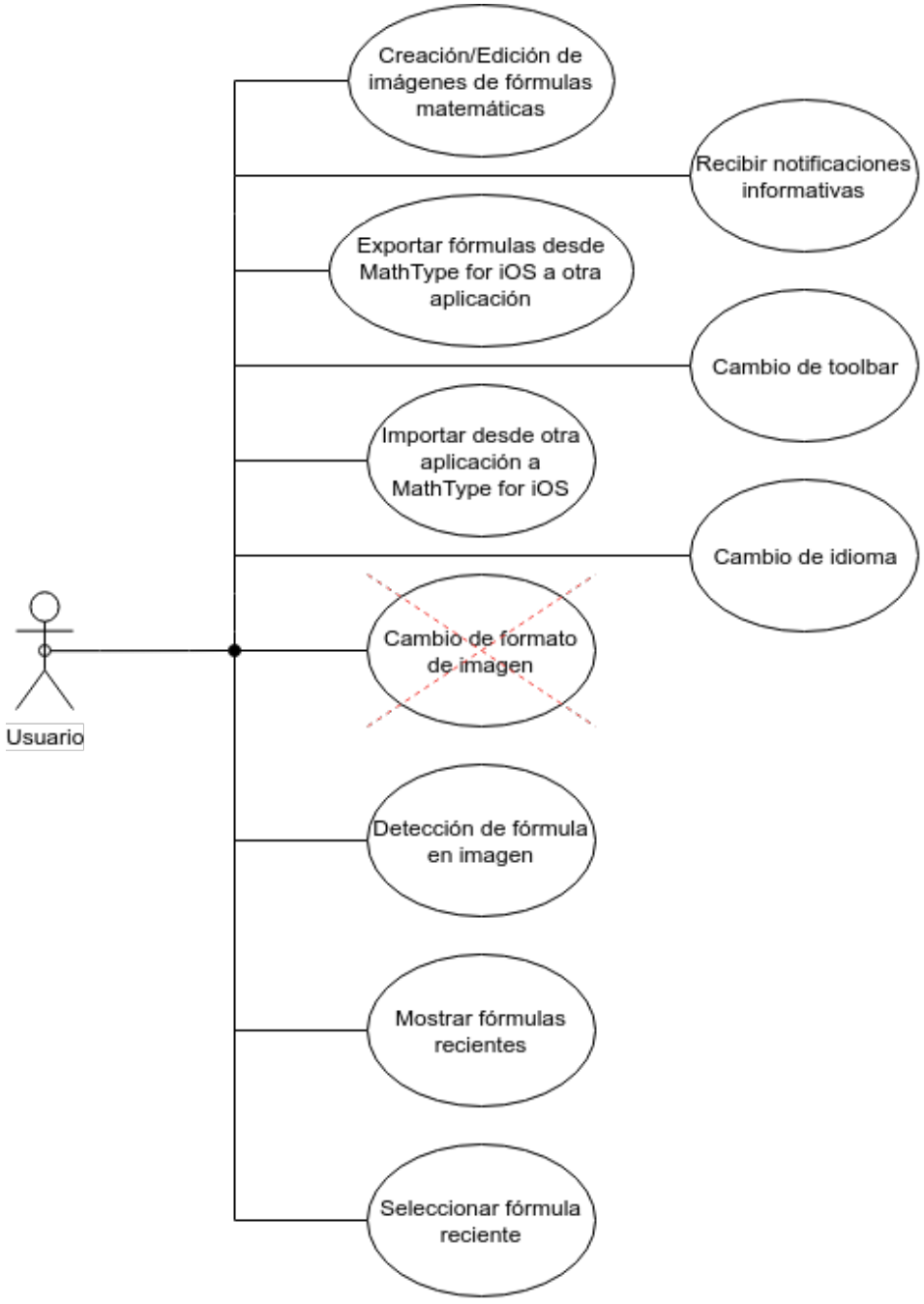


Figura 4.1: Diagrama de casos de uso.

1

[1]Como se ha comentado en los capítulos anteriores, el cambio de formato de imagen estaba contemplado durante el diseño y la toma de requisitos de la aplicación. Sin embargo, debido a limitaciones del sistema, se ha eliminado.

4.2. Especificación de los casos de uso

Para la especificación de los siguientes casos de uso, se ha utilizado una plantilla por la cual mostrar el nombre del caso de uso, el actor principal que se ve afectado, las precondiciones (si hubiese), el disparador que provoca la ejecución de dicho caso de uso, el escenario de éxito común y las extensiones (flujos de ejecución alternativos al escenario principal).

Caso de uso 001: Creación/Edición de imágenes de fórmulas matemáticas.
Actor principal: Usuario
Precondición: Ninguna.
Disparador: El usuario tiene MathType for iOS en primer plano.
Escenario principal de éxito

1. En cualquier momento se puede ir a la [Extensión 1] o [Extensión 2].
2. El usuario puede escribir y/o editar en cualquier momento una fórmula matemática.
3. El usuario obtiene dicha fórmula como imagen cada vez que inserta una fórmula.

Extensiones

1. Cierre total de la aplicación.
   1.1. El usuario indica que quiere acabar el caso de uso.
   1.2. El sistema acaba el caso de uso.
2. No es posible crear la imagen de la fórmula.
   2.1. El sistema va al [Caso de uso 007] e informa al usuario que no ha sido posible crear la fórmula.

Caso de uso 002: Exportar fórmulas desde MathType for iOS a otra aplicación.
Actor principal: Usuario
Precondición: La aplicación distinta a MathType for iOS tiene permite importar imágenes.
Disparador: El usuario quiere exportar una fórmula a otra aplicación
Escenario principal de éxito

1. En cualquier momento se puede ir a la [Extensión 1].

2. El usuario exporta una fórmula creada en MathType for iOS.
3. El usuario importa la fórmula exportada desde MathType for iOS a la otra aplicación abierta.
4. La fórmula queda insertada en la aplicación distinta a MathType for iOS.

Extensiones

1. Cierre total de la aplicación.
    1.1. El usuario indica que quiere acabar el caso de uso.
    1.2. El sistema acaba el caso de uso.

Caso de uso 003: Importar desde otra aplicación a MathType for iOS.
Actor principal: Usuario
Precondición: La aplicación distinta a MathType for iOS permite exportar imágenes.
Disparador: El usuario quiere importar una fórmula en MathType for iOS
Escenario principal de éxito

1. En cualquier momento se puede ir a la [Extensión 1].
2. El usuario exporta la imagen de la fórmula en la aplicación distinta a MathType for iOS.
3. El usuario importa la fórmula exportada de la segunda aplicación a MathType for iOS.
4. MathType for iOS muestra la fórmula arrastrada y permite su edición.

Extensiones

1. Cierre total de la aplicación.
    1.1. El usuario indica que quiere acabar el caso de uso.
    1.2. El sistema acaba el caso de uso.

Caso de uso 004: Detección de fórmula en imagen.
Actor principal: Usuario
Precondición: Ninguna.
Disparador: El usuario quiere detectar una fórmula en una fotografía.
Escenario principal de éxito

1. En cualquier momento se puede ir a [Extensión 1].
2. MathType for iOS permite escoger entre hacer una foto o escoger una de la biblioteca de fotos del propio dispositivo.
    2.1. Si escoge hacer una foto, se ejecuta la [Extensión 2].
    2.2. Si no, se ejecuta la [Extensión 3].
3. El sistema reconoce la fórmula en la imagen y permite editarla. Puede ejecutarse la [Extensión 4].

Extensiones

1. Cierre total de la aplicación.
    1.1. El usuario indica que quiere acabar el caso de uso.
    1.2. El sistema acaba el caso de uso.
2. Se escoge hacer una fotografía.
    2.1. El usuario puede volver en cualquier momento al principio del caso de uso.
    2.2. MathType for iOS permite al usuario hacer una fotografía con el dispositivo.
    2.3. El usuario realiza la fotografía.
    2.4. MathType for iOS guarda la fotografía en la biblioteca de fotos del dispositivo
3. Se escoge acceder a la biblioteca de fotos para seleccionar una imagen.
    3.1. El usuario puede volver en cualquier momento al principio del caso de uso.
    3.2. MathType for iOS abre la biblioteca de fotos y espera a que el usuario seleccione una imagen.
4. No se ha podido detectar la fórmula matemática dentro de la imagen proporcionada.
    4.1. El sistema no muestra ninguna fórmula matemática.
    4.2. El sistema informa al usuario de que no ha podido detectar ninguna fórmula en la imagen proporcionada ejecutando así el [Caso de uso 007].

Caso de uso 005: Mostrar fórmulas recientes.
Actor principal: Usuario
Precondición: Ninguna.
Disparador: El usuario tiene MathType for iOS en primer plano.
Escenario principal de éxito

1. En cualquier momento se puede ir a [Extensión 1].
2. El sistema muestra las fórmulas creadas y/o editadas previamente. Puede saltar la [Extensión 2]

Extensiones

1. Cierre total de la aplicación.
   1.1. El usuario indica que quiere acabar el caso de uso.
   1.2. El sistema acaba el caso de uso.
2. Anteriormente no se han creado ni editado fórmulas.
   2.1. El sistema no muestra ninguna fórmula creada/editada previamente.

Caso de uso 006: Seleccionar fórmula reciente.
Actor principal: Usuario
Precondición: Ninguna.
Disparador: El usuario selecciona una fórmula reciente.
Escenario principal de éxito

1. En cualquier momento se puede ir a [Extensión 1].
2. El sistema permite editar la fórmula seleccionada.

Extensiones

1. Cierre total de la aplicación.
   1.1. El usuario indica que quiere acabar el caso de uso.
   1.2. El sistema acaba el caso de uso.

Caso de uso 007: Recibir notificaciones informativas.
Actor principal: Usuario
Precondición: Ninguna.
Disparador: El sistema quiere informar al usuario.
Escenario principal de éxito

1. El sistema envía una notificacion con información al usuario.

Caso de uso 008: Cambio de toolbar.
Actor principal: Usuario
Precondición: Ninguna.
Disparador: El sistema pregunta al usuario qué toolbar, entre las que ofrece, quiere el usuario que sea mostrada en el editor.
Escenario principal de éxito

1. En cualquier momento se puede ir a [Extensión 1].
2. El usuario selecciona una de las toolbars ofrecidas por el sistema.
3. El editor cambiar su toolbar por la seleccionada.

Extensiones

1. Cierre total de la aplicación.
    1.1. El usuario indica que quiere acabar el caso de uso.
    1.2. El sistema acaba el caso de uso.

Caso de uso 009: Cambio de idioma.
Actor principal: Usuario
Precondición: Ninguna.
Disparador: El sistema pregunta al usuario en qué idioma, entre los ofrecidos, quiere que esté *MathType for iOS*.
Escenario principal de éxito

1. En cualquier momento se puede ir a [Extensión 1].
2. El usuario selecciona uno de los idiomas ofrecidos por el sistema.
3. *MathType for iOS* pone todos sus textos en el idioma especificado.

Extensiones

1. Cierre total de la aplicación.
    1.1. El usuario indica que quiere acabar el caso de uso.
    1.2. El sistema acaba el caso de uso.

4.3. Esquema conceptual.

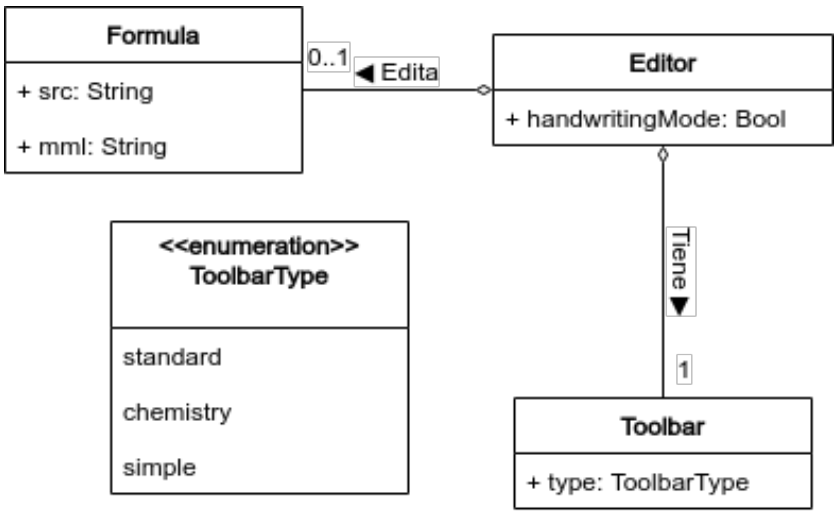


Figura 4.2: Esquema conceptual de los controladores del sistema.

Para terminar de hablar de la especificación del sistema, la figura anterior muestra el esquema conceptual del mismo.

Debido a que *MathType for iOS* es una integración de un software ya existente, pero adaptado para ser utilizado en *iOS*, el esquema se encuentra muy simplificado. La importancia recae durante la implementación en el diagrama lógico de la capa de presentación, es decir, la forma en la que se muestran los datos y se interaccionan con ellos.

Desde un punto de vista global, el esquema conceptual de la aplicación es un editor que puede tener o no una fórmula creada y que tiene una toolbar de herramientas.

Entrando ya en detalles específicos de cada clase, los atributos de la clase *Formula* son dos: *src* hace referencia a la url de la imagen desde donde se buscará para ser cargada. Por otro lado el atríbuto *mml* hace referencia a la representación según el estándar *MathML* de dicha fórmula.

En cuanto a la clase *Toolbar*, contiene el atributo *type* que puede tomar un valor entre las tres opciones ofrecidas por el enumeration *ToolbarType*. Estas tres opciones, han sido elegidas de manera arbitraria y son los nombres que se le han dado a las toolbars más utilizadas por los clientes de *Maths for More S.L.* .

Finalmente, la clase *Editor* contiene el atributo *handwritingMode* con un valor booleano que es verdadero cuando se encuentra en este modo o falso en el caso contrario.

Capítulo 5

Diseño

En este capítulo se tratará la justificación del diseño de *MathType for iOS*. Se entiende como diseño el paso previo a la implementación donde se detalla cómo tiene que ser ser el sistema y que provocará como consecuencia que se satisfagan los requisitos previamente descritos.

El diseño engloba diferentes aspectos que se irán tratando durante el capítulo, debido a que requisitos como el de la importancia de la interfaz de usuario requieren un estudio a priori del diseño externo.

Por otro lado, también existe un diseño arquitectónico o diseño interno desde el punto de vista del código para que éste cumpla con características como la mantenibilidad y reusabilidad.

Finalmente, se hablará también del diseño físico de la aplicación para explicar dónde se ejecuta cada parte del conjunto software.

5.1. Diseño externo

Para el diseño de la interfaz se ha realizado un prototipo de la aplicación por la cual poder navegar entre las diferentes pantallas.

En el caso de *MathType for iOS* se ha tenido en cuenta diferentes criterios para el diseño. Por ejemplo, el número de elementos que se muestran al usuario. Estos han sido ordenados de mayor a menor importancia dependiendo de la frecuencia de uso.

- Editor de fórmulas matemáticas con toolbar de herramientas
- Editor de fórmulas matemáticas en modo handwriting
- Fórmulas recientes

- Detección de fórmulas en una imagen
- Ajustes

Además, durante esta parte del diseño se han tenido en cuenta de los siguientes requisitos:

- Requisito no funcional 001: Diseño atractivo
- Requisito no funcional 002: Estilo autoritario
- Requisito no funcional 003: Usabilidad
- Requisito no funcional 005: Aprendizaje

Pero también ha sido importante la interacción con estos elementos. Por ejemplo, el cómo se quiere poder ejecutar la aplicación y cómo se quiere importar/exportar una fórmula.

En cuanto a la ejecución, se sabe de antemano que *iOS* permite los modos conocidos como *Split View* y *Slide Over*. Estos lo que hacen es mostrar en pantalla varias aplicaciones al mismo tiempo, con la diferencia de que *Split View* divide en dos la pantalla para mostrar a cada lado una aplicación diferente y *Slide Over* superpone la vista de una de las aplicaciones sobre otra.

Esto de cara al uso de *MathType for iOS* es una buena forma de mejorar la productividad debido a que ésta no tiene sentido de existencia por sí misma, sino que es una herramienta que complementa a otras aplicaciones.

Por la parte de importar/exportar una fórmula también se ha realizado un estudio previo y se sabe que *iOS* permite implementar la funcionalidad Drag&Drop. Ésta ayuda a eliminar elementos en pantalla que sugieran el importar o exportar, como añadir un botón, para así agarrar la imagen previsualizada y moverla a donde se quiere tener.

Un primer diseño final, sin tener en cuenta colores ni la guía de estilos del sistema operativo iOS ha sido el de la siguiente figura. En el apéndice A se pueden observar las capturas restantes de la aplicación.

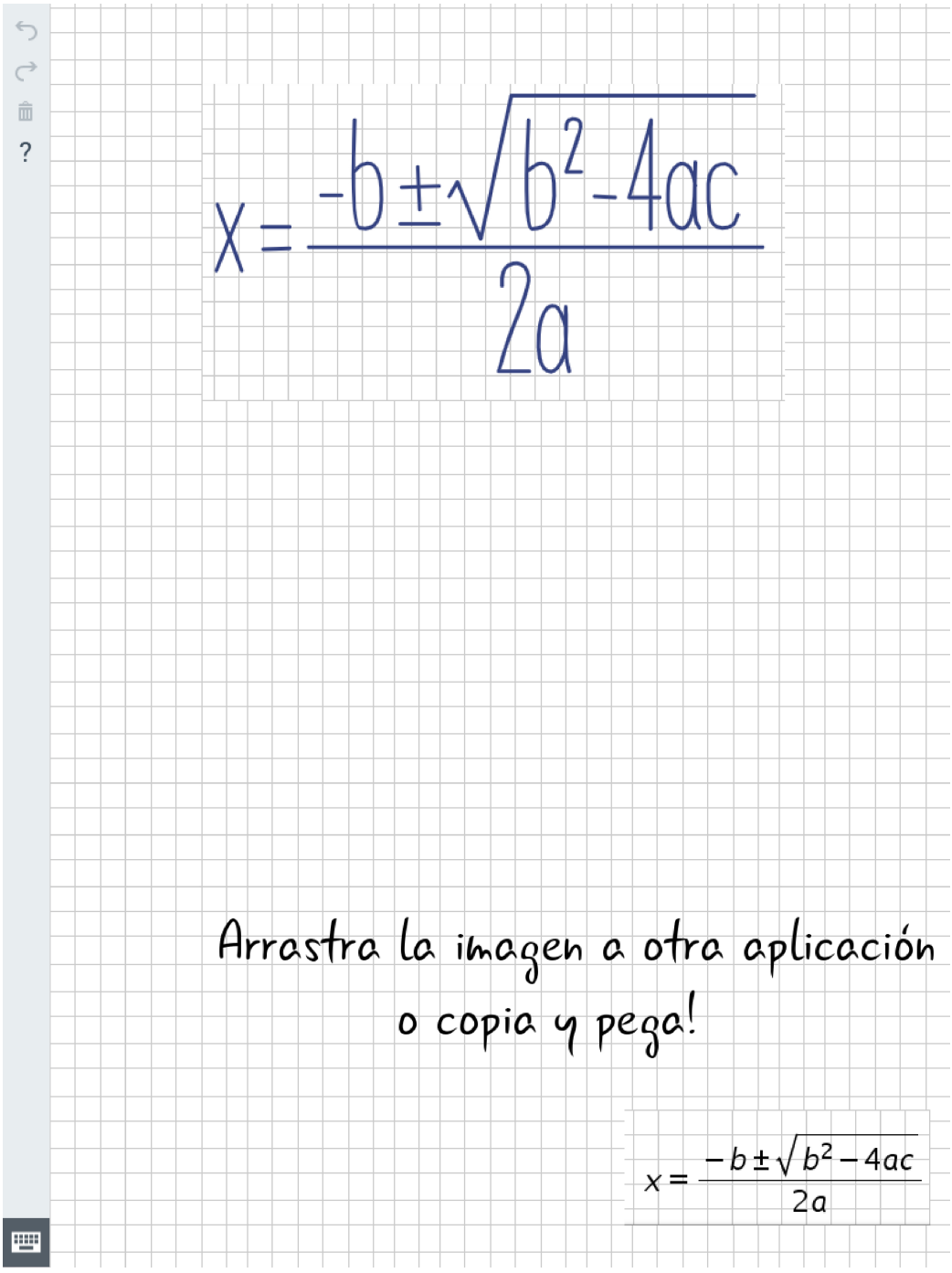


Figura 5.1: Pantalla principal de MathType for iOS.

Los motivos del uso de este layout para la aplicación son varios:

Primero de todo y como se ha comentado anteriormente es el uso que el usuario hace de la funcionalidad. Los usuarios de *MathType for iOS* utilizan el

modo manuscrito para la creación de fórmulas. Por lo tanto, que la pantalla inicial sea el modo handwriting del editor genera un aumento de la productividad y además reduce la curva de aprendizaje para utilizar la característica principal. Y como se puede ver en relación al Drag&Drop, la previsualización en este modo se espera que pueda ser arrastrada a otras aplicaciones.

Por otra parte nos encontramos que la edición de fórmulas, sobretodo las manuscritas, requiere de una superfície grande para sentirse cómodo, por lo que hay que optimizar el espacio. Además, hay que tener en cuenta que si se activa el modo Split View o Slide Over la superfície disminuye hasta un 70 %. Esto provoca que haya que aprovechar al máximo el espacio de trabajo.

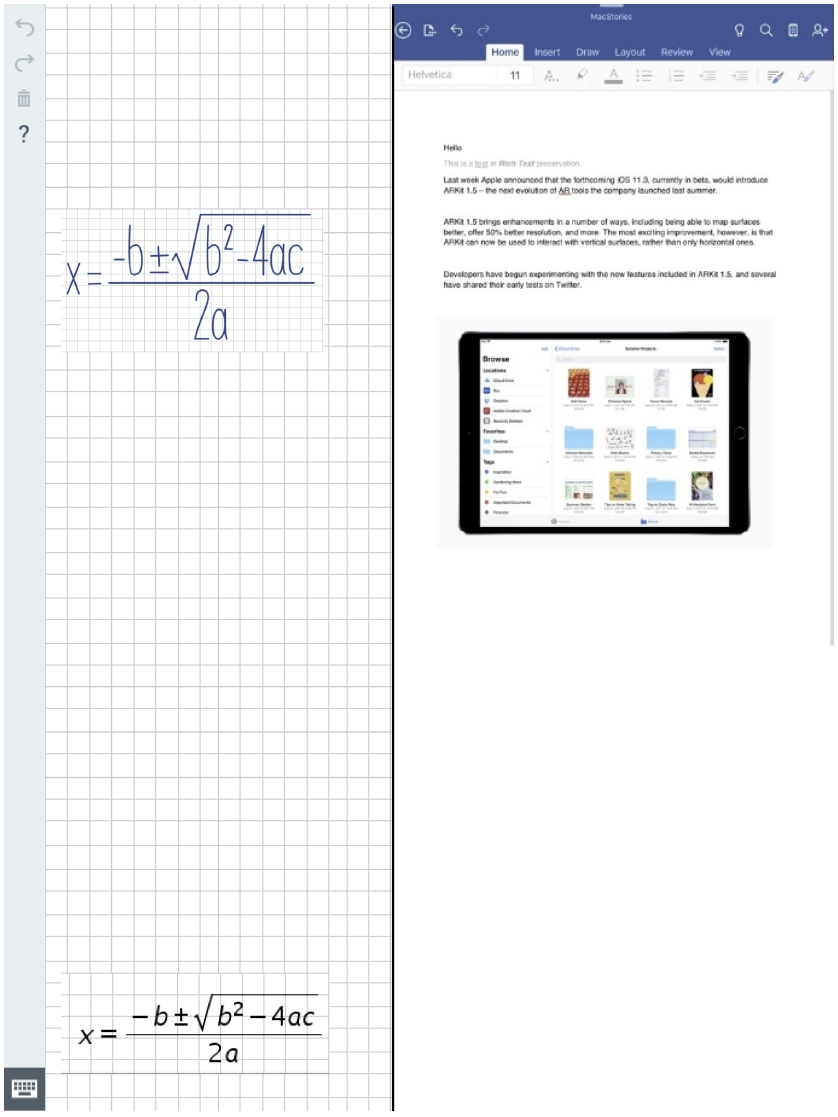


Figura 5.2: MathType for iOS en modo Split View.

Para resolver este problema, se ha decidido tener un menú que en principio no es visible hasta que el usuario lo necesita y lo activa mediante un gesto o posiblemente mediante un botón en una Navigation bar en la parte superior de la pantalla.

Finalmente, se ha decidido mantener el menú de ajustes lo más escondido posible debido a que un usuario no suele editar para cada fórmula los ajustes de la aplicación. Mantener estas opciones menos accesibles nos ayuda en puntos como: reducir la curva de aprendizaje, ya que el usuario no tiene constancia hasta más tarde que esas opciones existen; ofrecer flexibilidad, sólo si se requiere; y a mantener el menú lo más claro posible para que se puedan visualizar las "Fórmulas recientes".

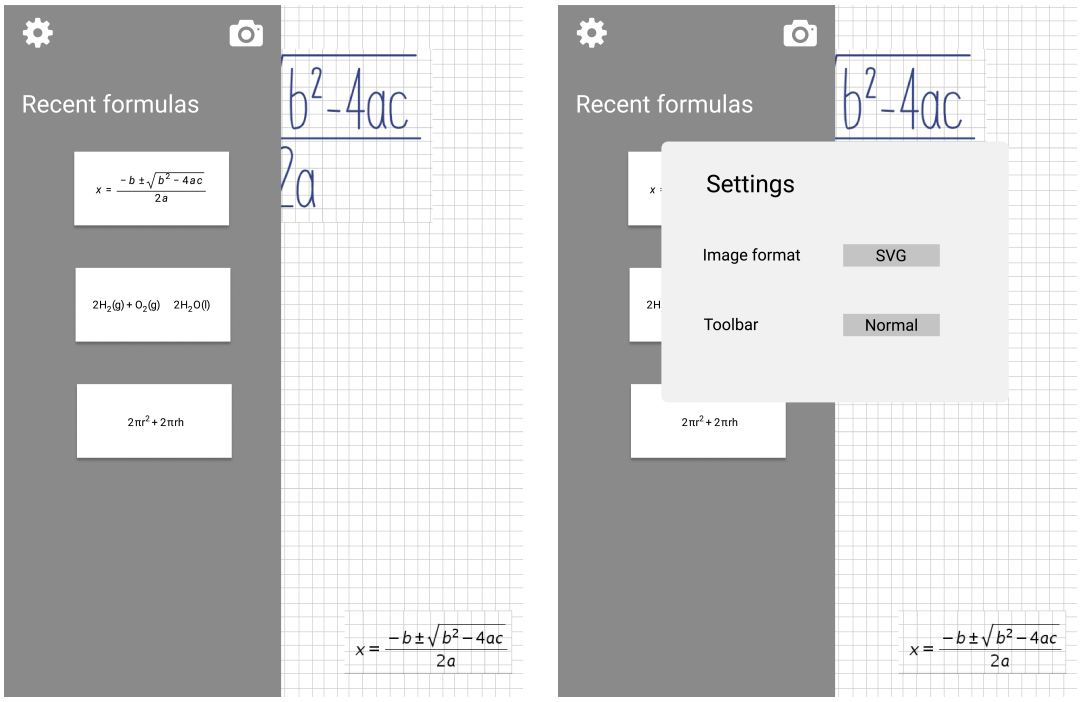


Figura 5.3: Menú de MathType for iOS con las fórmulas recientes, el acceso a ajustes y cámara.

5.2. Diseño interno

5.2.1. Complejidad del diseño

Las funcionalidades que quiere llegar a ofrecer *MathType for iOS* pueden ser sencillas a simple vista, sin embargo existe una complejidad escondida debido a diferentes factores como: el uso de distintos lenguajes de programación, la comunicación entre éstos o la inclusión de *MathType for iOS* en el ciclo de desarrollo y testeo de la empresa *Maths for More S.L.* que condicionan la arquitectura de la aplicación.

Modularidad

Para este proyecto, y como se explica en la planificación, existe un riesgo a la hora de estimar el número de horas. Con esto en mente es necesario ofrecer una solución que permita satisfacer los plazos de entrega de un producto terminado. Para ello se requiere pensar en tener un producto útil lo más pronto posible y después basarse en la idea de *construcción modular* para añadir a la aplicación todas aquellas funcionalidades extras que mejoren el producto pero que no sean necesarias en sí mismas.

Esto hace que la arquitectura requiera un bajo nivel de acoplamiento entre las diferentes componentes y así sea sencillo desactivar las que no den tiempo a terminar y que el producto siga en funcionamiento.

Limitaciones del sistema iOS

El sistema operativo donde se desarrolla la aplicación presenta limitaciones que afectan a la arquitectura del programa.

Primero de todo la gestión de la memoria por parte del garbage collector[27] se hace mediante las llamadas referencias fuertes y referencias débiles. Esto genera que la creación de las vistas tengan que guardar en algunos casos referencias fuertes para no perder el acceso.

Por otro lado, el sistema también se encuentra limitado en algunas funciones como las de comunicación entre aplicaciones. iOS separa la información entre aplicaciones por seguridad y eso genera una dificultad añadida para compartir información, que en el caso de MathType for iOS sería exportar e importar las fórmulas entre aplicaciones. Sin embargo, se permite usar ciertas clases como las de tipo *Imagen* para sortear esta dificultad.

Aún así, esto último supone un cambio en la forma de tratar la importación y exportación de fórmulas que ya realiza *MathType Web*. Dicha estrategia

lo que hace es mantener atributos con información relativa a la fórmula en formato *MathML* para así saber en todo momento qué formula se muestra en la imagen. Pero en este caso no es posible, ya que no está permitido compartir atributos adicionales ni en parámetros opcionales ni en la propia imagen dado que el sistema también elimina los campos de metadatos que puedan haber.

Simplicidad para cambiar las vistas

Actualmente la empresa *Maths for More S.L.* está trabajando en una nueva versión mayor de su editor MathType Web. Dicha versión implementará importantes cambios de interfaz, por lo que la arquitectura se ve condicionada de forma que debe aislar cada una de las vistas que presenta al usuario y facilitar su cambio en caso de ser necesario. De esta forma, ante un cambio repentino de interfaz es posible cambiar la vista asociada de manera simple y rápida.

Comunicación entre Swift y JavaScript

Los dos lenguajes que utiliza *MathType for iOS* son Swift[22] y JavaScript[15]. Swift es el idioma principal que se utiliza para el desarrollo nativo de la aplicación. Sin embargo, debido a que uno de los requisitos es la integración y extensión de MathType Web es necesario usar JavaScript para ello.

Esta decisión requiere entonces del uso de un sistema de comunicación entre ambos lenguajes para transferir la información entre el editor de fórmulas y la aplicación. Además del aprendizaje vinculado a ambos lenguajes.

5.2.2. Arquitectura interna

Patrón MVC

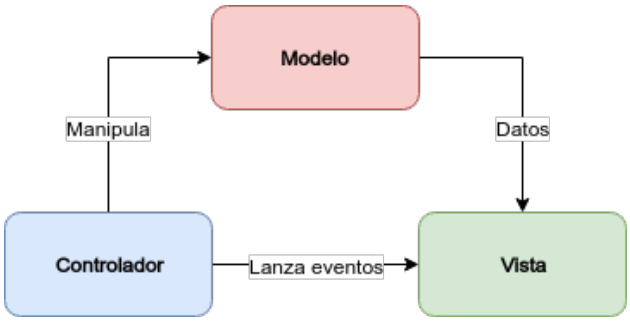


Figura 5.4: Diagrama de funcionamiento del patrón MVC.

Debido a la complejidad de la que se ha hablado anteriormente, el patrón de diseño escogido es el de Modelo-Vista-Controlador, abreviado como MVC[8].

Dicho patrón mejora la extensibilidad y mantenibilidad. Sus principales ventajas para este proyecto son:

- **La implementación se realiza de forma modular.** Esto ayuda al desarrollo modular comentado anteriormente. Dicha característica es una de las más importantes debido a que gracias a ella es posible añadir y quitar funcionalidades rápidamente y que el proyecto pueda seguir en funcionamiento, por lo que se hace interesante de cara al *TFG*, por si la falta de tiempo sugiere que ciertas funcionalidades deben ser eliminadas; y de cara a *Maths for More S.L.* por si requieren de lanzar versiones de prueba donde observar si una nueva funcionalidad es útil para sus usuarios o no. En las siguientes secciones se explica con más detalles el cómo se consigue.
- **La modificación de las vistas no afecta al modelo del dominio.** Esto hace que la vista tenga un acoplamiento bajo respecto al modelo y por lo tanto es más sencillo cambiar la vista por otra en caso de ser necesario, es decir, solamente la representación de los datos varía y no los datos.
- **La modificación del modelo solamente afecta a éste.** Esto implica que en caso de cambio en el modelo bastaría con modificar las interfaces que puedan haber.

A cambio, nos supondría tener las siguientes desventajas:

- **Mayor tiempo inicial de desarrollo.** La implementación del MVC es costosa al principio, provocando que los tiempos de desarrollo iniciales aumenten. Esto afecta de manera directa a la planificación del proyecto del Capítulo 8.
- **Requiere de un mecanismo de eventos.** Dichos eventos serán utilizados para responder de manera específica ante situaciones concretas.
- **El número de clases e interfaces es mayor**. En otros patrones no son necesarias grandes cantidades de clases, esto a la larga podría llegar a provocar problemas de gestión de archivos para contener dichas clases y/o repetición de código si no se sabe de la existencia de todas las clases del sistema.

Diseño estático del patrón MVC

A continuación se explican los detalles más importantes del diseño estático del patrón MVC. Cada párrafo explica uno de los componentes (*Modelo*, *Vista* o *Controlador*) y muestra de manera aislada el diagrama lógico asociado a dicho componente. Para ver el diagrama lógico global del sistema, hay que ir al *Apéndice C*.

Como aclaración previa, aquellas clases que contienen prefijos "NS", "UI" o "WK" son clases de Swift y por lo tanto no se han especificado sus atributos y/o métodos, solamente se ha hecho una referencia a ellos.

Modelo

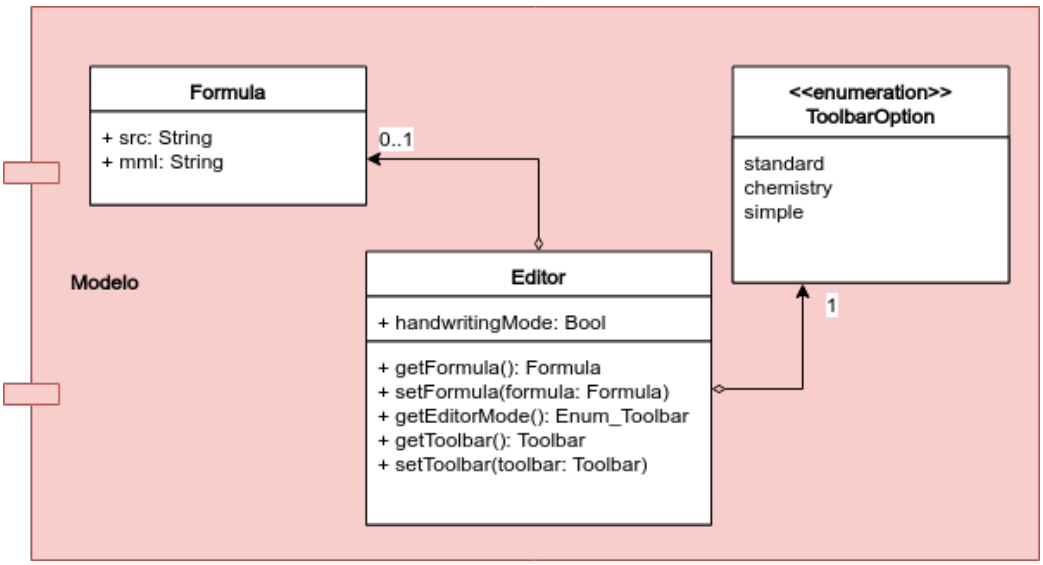


Figura 5.5: Diagrama lógico del modelo.

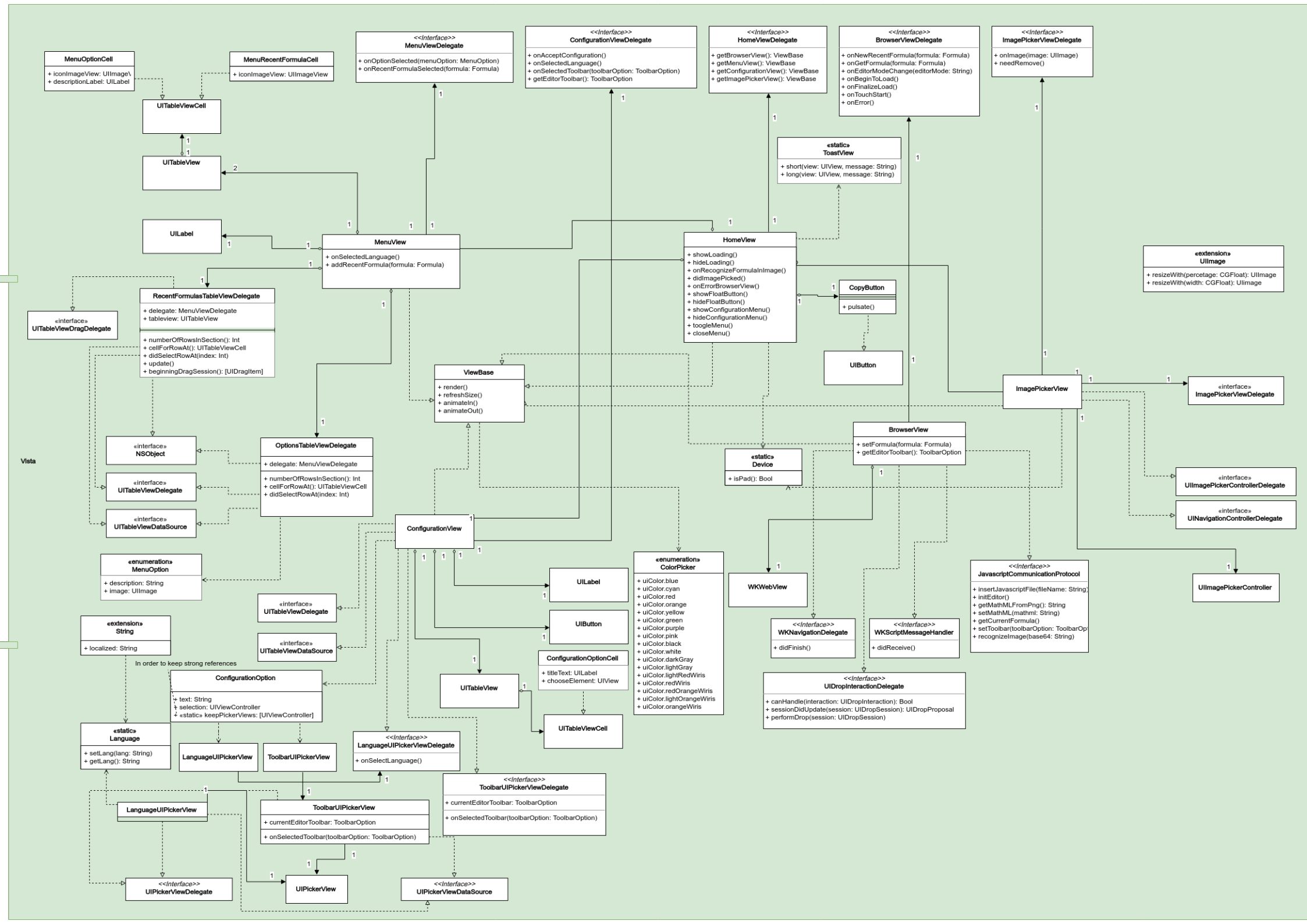
El sistema a implementar por *MathType for iOS* no requiere de la gestión de ninguna base de datos. Sin embargo, el modelo se ocupará de crear la estructura interna de los objetos que se utilizan en el modelado conceptual. Esto quiere decir que en él estarán todas las clases que requieran de una semántica, como las fórmulas o el propio editor.

El motivo de hacer esto es para crear una capa de abstracción y estandarización de las estructuras durante la comunicación entre capas. Si se pasaran primitivas se podría dar el problema de tener funciones con muchos parámetros; y si se pasaran objetos básicos (clase de la que heredan todas las clases) el mantenimiento del código se reduce ya que o bien se documenta qué propiedades y atributos tiene que cumplir dicho objeto, o se generan inconsistencias

como acceder a una propiedad inexistente. Además, esto permite tratar los datos si así se requiere solamente cambiando el valor de retorno de estas clases.

Finalmente, tanto las *vistas* como los *controladores* acceden al modelo y lo utilizan en sus propias llamadas, por lo que se crean relaciones de dependencia hacia estas clases del modelo.

Vista


Vista
MenuOptionCell
+ iconImageView: UIImage\
+ descriptionLabel: UILabel
MenuRecentFormulaCell
+ iconImageView: UIImageView
UITableViewCell
UITableView
UILabel
<<Interface>>
MenuViewDelegate
+ onOptionSelected(menuOption: MenuOption)
+ onRecentFormulaSelected(formula: Formula)
<<Interface>>
ConfigurationViewDelegate
+ onAcceptConfiguration()
+ onSelectedLanguage()
+ onSelectedToolbar(toolbarOption: ToolbarOption)
+ getEditorToolbar(): ToolbarOption
<<Interface>>
HomeViewDelegate
+ getBrowserView(): ViewBase
+ getMenuView(): ViewBase
+ getConfigurationView(): ViewBase
+ getImagePickerView(): ViewBase
<<Interface>>
BrowserViewDelegate
+ onNewRecentFormula(formula: Formula)
+ onGetFormula(formula: Formula)
+ onEditorModeChange(editorMode: String)
+ onBeginToLoad()
+ onFinalizeLoad()
+ onTouchStart()
+ onError()
<<Interface>>
ImagePickerViewDelegate
+ onImage(image: UIImage)
+ needRemove()
«static»
ToastView
+ short(view: UIView, message: String)
+ long(view: UIView, message: String)
MenuView
+ onSelectedLanguage()
+ addRecentFormula(formula: Formula)
HomeView
+ showLoading()
+ hideLoading()
+ onRecognizeFormulaInImage()
+ didImagePicked()
+ onErrorBrowserView()
+ showFloatButton()
+ hideFloatButton()
+ showConfigurationMenu()
+ hideConfigurationMenu()
+ toogleMenu()
+ closeMenu()
CopyButton
+ pulsate()
UIButton
«extension»
UIImage
+ resizeWith(percetage: CGFloat): UIImage
+ resizeWith(width: CGFloat): UIimage
RecentFormulasTableViewDelegate
+ delegate: MenuViewDelegate
+ tableview: UITableView
+ numberOfRowsInSection(): Int
+ cellForRowAt(): UITableViewCell
+ didSelectRowAt(index: Int)
+ update()
+ beginningDragSession(): [UIDragItem]
«interface»
UITableViewDragDelegate
ViewBase
+ render()
+ refreshSize()
+ animateIn()
+ animateOut()
ImagePickerView
«interface»
ImagePickerViewDelegate
«interface»
NSObject
OptionsTableViewDelegate
+ delegate: MenuViewDelegate
+ numberOfRowsInSection(): Int
+ cellForRowAt(): UITableViewCell
+ didSelectRowAt(index: Int)
«interface»
UITableViewDelegate
«interface»
UITableViewDataSource
BrowserView
+ setFormula(formula: Formula)
+ getEditorToolbar(): ToolbarOption
«static»
Device
+ isPad(): Bool
«interface»
UIImagePickerControllerDelegate
«interface»
UINavigationControllerDelegate
ConfigurationView
«enumeration»
MenuOption
+ description: String
+ image: UIImage
«enumeration»
ColorPicker
+ uiColor.blue
+ uiColor.cyan
+ uiColor.red
+ uiColor.orange
+ uiColor.yellow
+ uiColor.green
+ uiColor.purple
+ uiColor.pink
+ uiColor.black
+ uiColor.white
+ uiColor.darkGray
+ uiColor.lightGray
+ uiColor.lightRedWiris
+ uiColor.redWiris
+ uiColor.redOrangeWiris
+ uiColor.lightOrangeWiris
+ uiColor.orangeWiris
WKWebView
<<Interface>>
JavascriptCommunicationProtocol
+ insertJavascriptFile(fileName: String)
+ initEditor()
+ getMathMLFromPng(): String
+ setMathML(mathml: String)
+ getCurrentFormula()
+ setToolbar(toolbarOption: ToolbarOp
+ recognizeImage(base64: String)
UIImagePickerController
«extension»
String
+ localized: String
In order to keep strong references
ConfigurationOption
+ text: String
+ selection: UIViewController
+ «static» keepPickerViews: [UIViewController]
ConfigurationOptionCell
+ titleText: UILabel
+ chooseElement: UIView
<<Interface>>
WKNavigationDelegate
+ didFinish()
<<Interface>>
WKScriptMessageHandler
+ didReceive()
<<Interface>>
UIDropInteractionDelegate
+ canHandle(interaction: UIDropInteraction): Bool
+ sessionDidUpdate(session: UIDropSession): UIDropProposal
+ performDrop(session: UIDropSession)
«static»
Language
+ setLang(lang: String)
+ getLang(): String
LanguageUIPickerView
ToolbarUIPickerView
<<Interface>>
LanguageUIPickerViewDelegate
+ onSelectLanguage()
<<Interface>>
ToolbarUIPickerViewDelegate
+ currentEditorToolbar: ToolbarOption
+ onSelectedToolbar(toolbarOption: ToolbarOption)
LanguageUIPickerView
ToolbarUIPickerView
+ currentEditorToolbar: ToolbarOption
+ onSelectedToolbar(toolbarOption: ToolbarOption)
<<Interface>>
UIPickerViewDelegate
UIPickerView
<<Interface>>
UIPickerViewDataSource

MathType for iOS utiliza las llamadas "vistas" para darle una representación a los datos que serán visualizados.

Una de las preocupaciones más grandes con dichas vistas es la mejora de su reusabilidad y modularidad. Para ello se llevan a cabo una serie de restricciones, algunas se reflejan en el diagrama lógico en forma de clases o relaciones y otras son restricciones textuales que afectan sobretodo al diagrama de secuencia de algunas de las acciones.

Las restricciones que se han llevado a cabo son las siguientes:

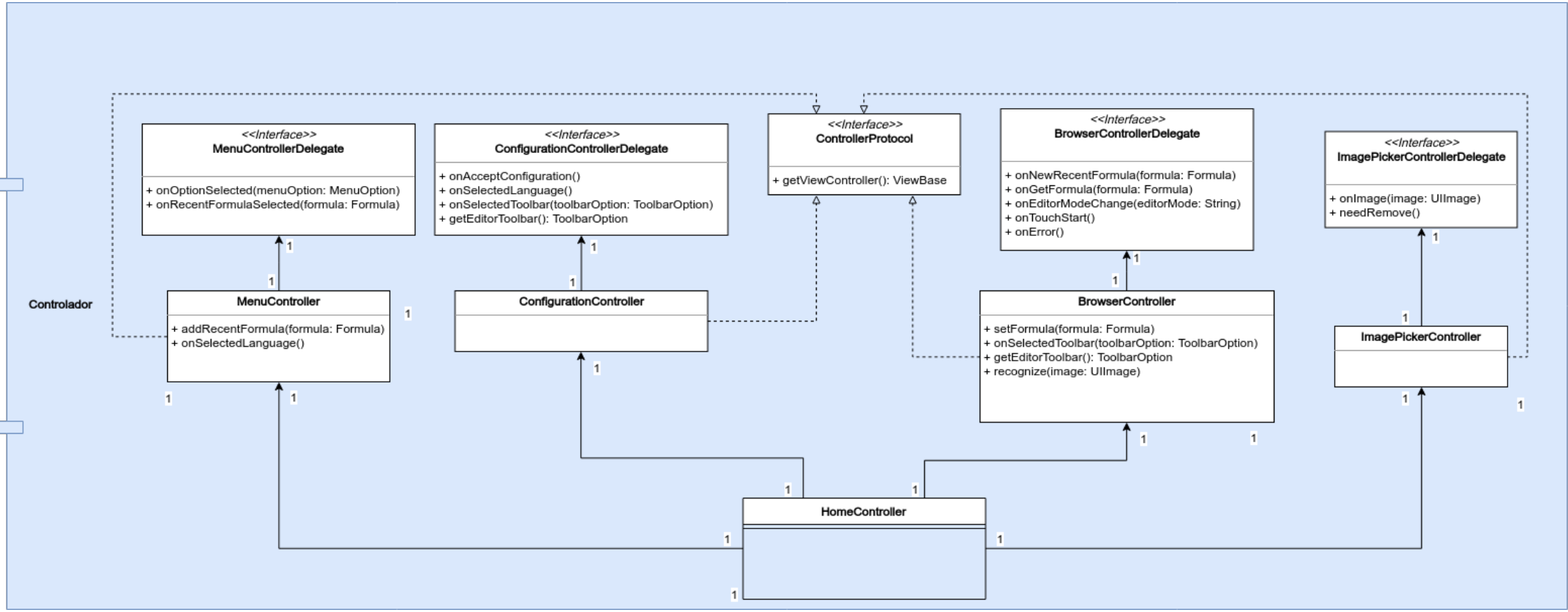
- **Solamente una vista puede decidir qué y cómo representar información al usuario.** Ya que como se ha dicho antes, se delega el trabajo de representación de los datos a esta capa.
- **Las vistas tienen referencias a interfaces en vez de a sus propios controladores.** Debido a que una vista tiene un potencial de cambio mayor que el de un controlador, se requiere una capa de abstracción como lo puede ser una interfaz para disminuir el acoplamiento.
- **Los límites de la vista los define la vista padre.** Esto quiere decir que una vista le proporciona un espacio visual a la subvista donde tiene permitido renderizarse. De otra manera, podríamos encontrar problemas de solapamiento de vistas y contención de las mismas. Por ejemplo si dos subvistas deciden ocupar la mitad del espacio, ya que en este caso habría que ir a cada subvista y configurarlas para ocupar sólo la mitad. En cambio, si ocupan todo el espacio de renderizado y el padre es el encargado de decidir que quiere la mitad para cada una, otra vista que tenga esas mismas subvistas puede ponerlas en la distribución que le apetezca sin crear una subclase de dichas subvistas para cambiar la configuración de cuánto espacio van a ocupar. Por lo tanto se mejora la mantenibilidad y reusabilidad del código.
- **Cada vista solamente puede comunicarse con su propio controlador.** Esto aumenta la cantidad de código ya que muchas veces se crean largas cadenas de llamadas. Por ejemplo, si Vista A realiza una acción que afecta a la vista B, se crea la cadena "Vista A llama a Controlador A - Controlador A llama a Controlador B - Controlador B llama a Vista B". Sin embargo, se crean menos dependencias entre vistas y/o controladores, por lo que con ello se consigue aislar funcionalidades. De esta manera, es posible mantener el sistema por módulos donde, desactivando un controlador de una funcionalidad específica en

la clase principal, se puede eliminar dicha funcionalidad. Análogamente, si activamos un controlador nuevo y añadimos su vista, es posible añadir una nueva funcionalidad.

En cuanto al segundo punto, afecta de manera directa al diagrama lógico tal y como se ve con la clase *HomeView*. Cada vista tiene una relación con su propia interfaz delegada que sigue el patrón de nombre "[NombreDeLaVista]Delegate", cosa que le permite obtener vistas genéricas del tipo *ViewBase* para ser añadidas. De esta forma, se desacoplan las subvistas y el delegado (el controlador que implementa dicha interfaz) es quien se ocupa de la lógica.

En cuanto a los demás puntos, se tratan de restricciones textuales y dependen de manera directa del programador, que es responsable de no incumplirlas para no romper la arquitectura del sistema.

Controlador


Controlador
<<Interface>>
MenuControllerDelegate
+ onOptionSelected(menuOption: MenuOption)
+ onRecentFormulaSelected(formula: Formula)
<<Interface>>
ConfigurationControllerDelegate
+ onAcceptConfiguration()
+ onSelectedLanguage()
+ onSelectedToolbar(toolbarOption: ToolbarOption)
+ getEditorToolbar(): ToolbarOption
<<Interface>>
ControllerProtocol
+ getViewController(): ViewBase
<<Interface>>
BrowserControllerDelegate
+ onNewRecentFormula(formula: Formula)
+ onGetFormula(formula: Formula)
+ onEditorModeChange(editorMode: String)
+ onTouchStart()
+ onError()
<<Interface>>
ImagePickerControllerDelegate
+ onImage(image: UIImage)
+ needRemove()
MenuController
+ addRecentFormula(formula: Formula)
+ onSelectedLanguage()
ConfigurationController
BrowserController
+ setFormula(formula: Formula)
+ onSelectedToolbar(toolbarOption: ToolbarOption)
+ getEditorToolbar(): ToolbarOption
+ recognize(image: UIImage)
ImagePickerController
HomeController
1

Dentro de la aplicación, los controladores tienen el rol de recibir los cambios del modelo y viceversa. Además, adquieren también la tarea de configuración y coordinación de eventos. Esto quiere decir que son los encargados de llamar a los otros controladores o su propia vista para entregar dichos eventos, con los cuales es posible administrar el ciclo de vida de la aplicación.

Para ello, se hace uso de clases o interfaces apodadas *Delegate*. Dichos elementos son utilizados para desacoplar de los controladores las referencias a otros objetos en los que se "delegan" el tratamiento de sus propios eventos.

Sin embargo, debido a la estructura de la aplicación en la que hay una clase (*HomeController*) que representa la "pantalla principal", ésta sí necesita conocer el estado global de la aplicación y las funciones de los otros controladores, por lo que queda fuertemente acoplada con las demás clases. Pero, al ser la encargada de instanciar los demás controladores y comunicar los cambios que puedan provocar entre ellos, como consecuencia de sus acciones, se ve justificado. Por último, se cree improbable tener cambios en los eventos del sistema, así que también es improbable tener consecuencias debido al nivel de acoplamiento.

Diseño dinámico del patrón MVC

Aquí, se explican los detalles referentes al diseño dinámico de la implementación realizada del patrón MVC. La motivación de este apartado es explicar algunas de las restricciones de desarrollo que se quieren imponer en la creación y uso de los componentes *Vista* y *Controlador*.

Vista. Como se ha hablado durante el diseño estático la *Vista*, se han incluido restricciones para mejorar la reusabilidad y mantenibilidad del código. Entre éstas, la restricción de *Los límites de la vista los define la vista padre* y *Cada vista solamente puede comunicarse con su propio controlador* tienen consecuencias en el "qué se espera del desarrollador a la hora de crear y mantener *MathType for iOS*".

Así pues, el siguiente diagrama muestra una consecuencia directa de estas restricciones en la forma de actuar del sistema. Sin embargo, hay que tener en cuenta que no se incluyen los diagramas de las subvistas, ya que sería un ejemplo análogo y por lo tanto no es relevante para esta demostración. Además, se han obviado los diagramas de llamadas como *addSubView* por tener nombres autoexplicativos.

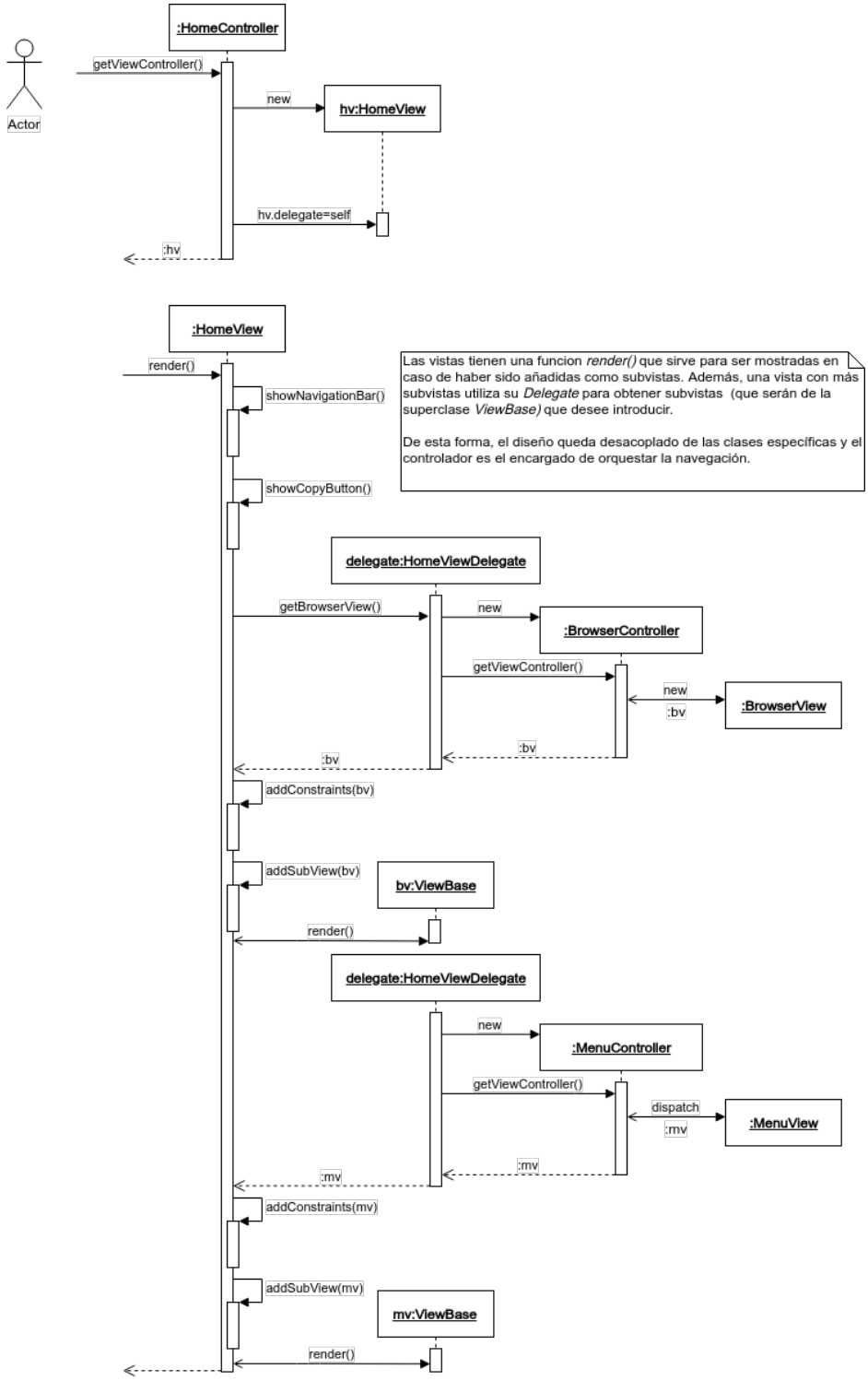


Figura 5.6: Diagrama de secuencia de la vista HomeView.

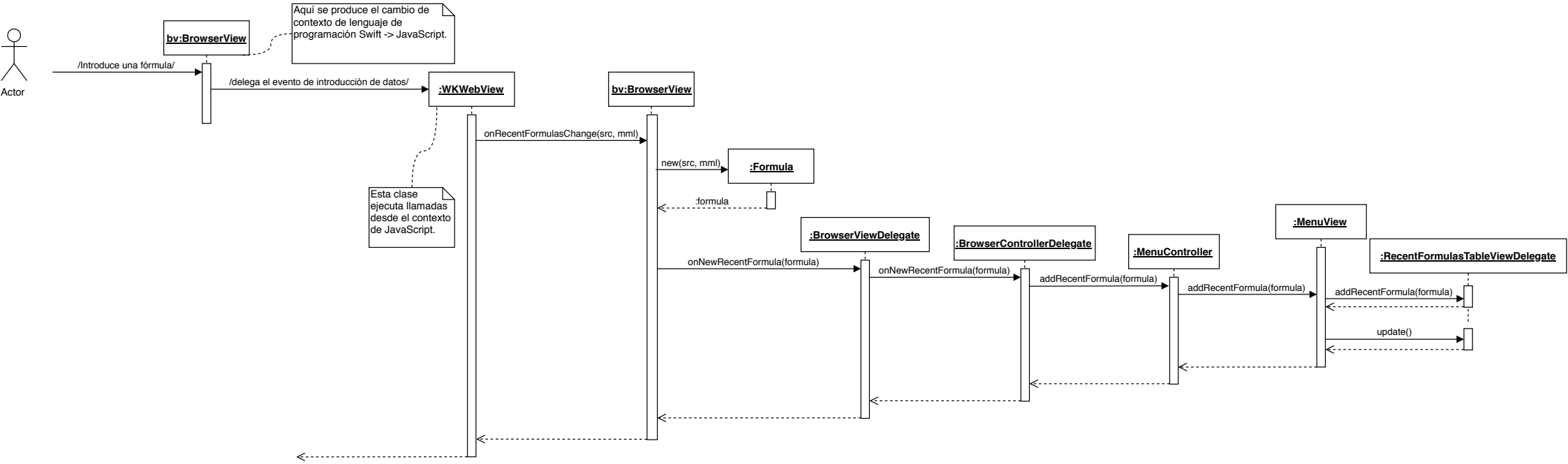
La secuencia muestra como la vista padre *HomeView* utiliza una llamada a la referencia de la interfaz que implementa su controlador. A partir de aquí el controlador se hace cargo de la lógica de saber a qué otros controladores debe llamar y éstos en devolver la vista correcta.

Esta vista, como se puede ver en el diagrama lógico, es de la superclase *ViewBase*. Esto se hace para ocultar el tipo específico que finalmente la vista *HomeView* inserta, desacoplando así las subvistas específicas tanto de los controladores que actúan de intermediario durante la cadena de llamadas como de las vistas padre.

A su vez y como se comentaban en las restricciones que se han impuesto anteriormente, la vista padre en ningún momento modifica su zona de renderizado, sino que solamente tiene permitido modificar las de las subvistas propias (momento en el que llama a la función *addConstraints(...)*) dentro del espacio asignado para sí misma.

Finalmente, también hay que comentar la llamada a *render()*. Esta llamada permite a las vistas tener el poder de decisión de cuándo la subvista va a ser inicializada. Esto se debe a que hay momentos en el que no interesa cargar una vista hasta un cierto momento y es mejor esperar a que el elemento vaya a ser utilizado.

Controlador. Para terminar de explicar el diseño dinámico de *MathType for iOS* a continuación se muestra el comportamiento que existe cuando se da un evento y éste se propaga por las diferentes clases del sistema gracias a los eventos manejados por los controladores. En concreto, el caso de uso en el que hay que actualizar las fórmulas recientes debido a que el usuario ha introducido una fórmula nueva en la vista donde se encuentra el editor.


Actor
/Introduce una fórmula/
bv:BrowserView
Aquí se produce el cambio de contexto de lenguaje de programación Swift -> JavaScript.
/delega el evento de introducción de datos/
:WKWebView
Esta clase ejecuta llamadas desde el contexto de JavaScript.
onRecentFormulasChange(src, mml)
bv:BrowserView
new(src, mml)
:Formula
:formula
onNewRecentFormula(formula)
:BrowserViewDelegate
onNewRecentFormula(formula)
:BrowserControllerDelegate
addRecentFormula(formula)
:MenuController
addRecentFormula(formula)
:MenuView
addRecentFormula(formula)
:RecentFormulasTableViewDelegate
update()

Lo más destacable de esta secuencia es, que debido a que los eventos se provocan en el *WebView*, es éste quien avisa a la vista. Una vez hecho esto la vista delega la responsabilidad de tratar el evento a su *delegado*, que en este caso sería *BrowserController* (que ha implementado la interfaz *BrowserViewDelegate*). Entonces, éste notifica al controlador de la vista padre el evento.

Llegados a este punto, *HomeController* decide que dicho evento requiere ser redirigido al controlador *MenuController*, el cual aplica la decisión que ante dicho evento es necesario actualizar las fórmulas recientes que son mostradas en su vista.

Puente de comunicación Swift-JavaScript

Como se explica en el alcance de *MathType for iOS*, por temas de reusabilidad y mantenibilidad de código se quiere integrar la versión web de su editor en el sistema. La versión web de *MathType* (o *MathType Web*) se ejecuta en el lenguaje *JavaScript*.

El sistema operativo iOS permite añadir una vista llamada *WebView*[6] con la que visualizar contenido web como si de un navegador se tratara. *MathType for iOS* incluye esta vista para así tener la posibilidad de crear un contexto donde ejecutar *MathType Web*.

Este contexto de ejecución es donde se encuentra una instancia de *Editor* y además proporciona un método de comunicación entre *Swift* y *JavaScript* mediante inyección de código.

Para la comunicación se establece la interfaz *JavascriptCommunicationProtocol*, tal y como se muestra en el diagrama lógico, que la vista contenedora del *WebView* tiene que implementar. Dicha interfaz establece un protocolo para gestionar la inyección de código y objetos en el contexto de JavaScript, permitiendo así evaluar código JavaScript desde Swift y crear objetos JavaScript que invoquen código de Swift.

Para acabar, el código JavaScript gestiona su ciclo de vida mediante eventos, que son llamadas a partes específicas del código cuando ocurren situaciones concretas como un cambio en los estilos mostrados o un click sobre un elemento[13]. Dichos eventos son lanzados tanto por los cambios que se producen en el contexto de ejecución de *JavaScript* como en el de *Swift*. Varios ejemplos de eventos son: a nivel de *JavaScript*, la comunicación hacia *Swift* de que una nueva fórmula ha sido creada y por lo tanto debe insertar la que había anteriormente como "fórmula reciente"; y a nivel de *Swift*, la elección por parte del usuario de una toolbar específica para el editor.

El principal motivo que justifica el uso de los eventos como método de gestión del ciclo de vida en *JavaScript* es la naturaleza de esta tecnología, que gira entorno al uso de eventos. Existen eventos de muchos tipos que permiten ejecutar partes de código de manera paralela con relativa facilidad. Por lo tanto, disminuye los errores y aumenta la simplicidad del código, cosa que implica un aumento de la mantenibilidad.

Uso del patrón estrategia en las opciones de configuración

Debido a que las opciones de configuración son presentadas en una tabla donde el usuario puede escoger un valor, existe un comportamiento común. Por un lado siempre hay un texto descriptivo de la opción que se quiere cambiar y por otro lado un elemento de *input* para permitir al usuario insertar datos, que puede ser de diferentes tipos.

Para mejorar la reutilización del código, se ha implementado un patrón estrategia[16] en la clase *ConfigurationOption* devolviendo una clase común al tipo de elemento que se quiere mostrar y que es propia del sistema en *iOS* (*UIViewController*). Éste permite variar el tipo de elemento utilizado para que el usuario inserte información. Por ejemplo, en el caso del idioma el elemento adquiere la forma de una lista donde le es posible seleccionar qué idioma quiere. Mientras que en una opción como la resolución de imagen que se quiere, podría ser una casilla donde insertar un valor numérico entre 0 y 100.

Además, también se ha utilizado la clase *ConfigurationOption* para aplicar el patrón factoría[17]. Permitiendo facilitar la instanciación de los objetos sin la necesidad de exponer la lógica de su creación.

Finalmente, como comentario, hay que destacar el atributo estático *keepPickersViews* de *ConfigurationOption* que se puede ver en el diseño lógico. Este atributo permie solucionar un problema presente en *iOS* referente al uso del patrón factoría. Ya que la creación de instancias haciendo uso de este patrón genera referencias débiles de los objetos creados, por lo que son borrados por el *garbage collector* a pesar de estar siendo usados. Dicha variable mantiene en memoria los objetos.

5.3. Diseño físico

Terminando ya de explicar el diseño de *MathType for iOS*, falta detallar el diseño físico que utiliza.

MathType for iOS es una aplicación que se ejecuta de manera local, sin embargo, realiza llamadas externas a servicios de *Maths for More S.L.*. Esto se debe a que funcionalidades como el reconocimiento de fórmulas o el propio archivo que contiene el editor *MathType Web* que se quiere integrar en este trabajo son dependencias que se obtienen a través de llamadas externas.

Debido a ello y a que solamente se comunica con un servidor, el diseño físico coincide con la arquitectura *cliente-servidor*[50].

Este tipo de arquitectura realiza un procesamiento cooperativo donde cada uno de los componentes pide servicios a otro.

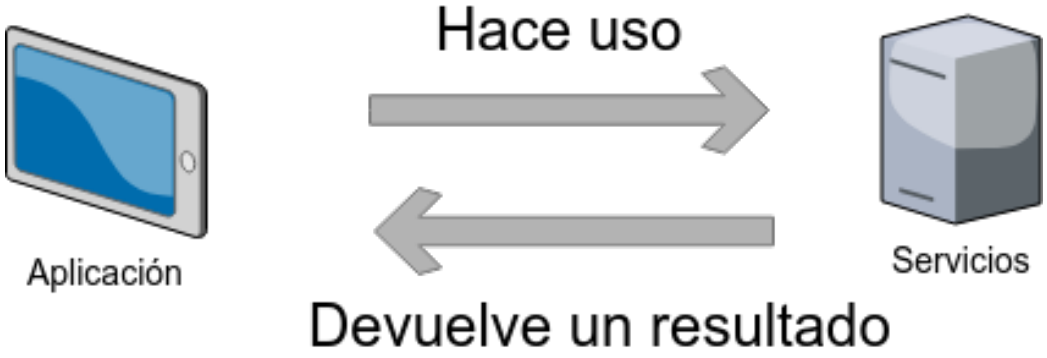


Figura 5.7: Diagrama físico de *MathType for iOS*.

Como se puede ver en la figura anterior, la aplicación realiza peticiones al servidor. De esta forma delega el trabajo más complejo y que requiere de mayor poder de cómputo y recibe un resultado que posteriormente trata.

Las principales ventajas de utilizar este sistema son: delegar un trabajo de cómputo pesado a una máquina con mayor potencia y así acelerar las operaciones y la escalabilidad, ya que se puede aumentar tanto la capacidad del cliente como la del servidor por separado.

Sin embargo, supone a su vez tener desventajas, como la dependencia de un servicio externo por el cual es obligatorio tener una conexión a Internet; también los problemas de latencia de la red, que pueden perjudicar a la velocidad de respuesta; y por último la sobresaturación de los servidores, en caso de que muchos clientes realicen peticiones al mismo tiempo hacia un mismo servidor.

Capítulo 6

Implementación y puesta en producción

Debido a que el diseño puede contener errores o no ser del todo exhaustivo, existe la posibilidad que durante la implementación del software se encuentren más limitaciones de las esperadas y por lo tanto problemas que no se habían tenido en cuenta previamente, por este motivo este capítulo explica los problemas que han ido apareciendo a lo largo del proyecto.

Además, también se explica la automatización implementada para *MathType for iOS*. Ésta, genera los paquetes finales que serán distribuidos a los clientes, verifica su calidad y se alinea la producción con los demás proyectos de la compañía *Maths for More S.L.* haciendo uso de los mismos estándares.

Y por último, es posible acceder al código software de *MathType for iOS* del cual se hace referencia aquí.

6.1. Dificultades encontradas

6.1.1. Cambios en el diseño visual

Durante la etapa de diseño se explica el cómo tiene que ser visualmente *MathType for iOS*. Sin embargo, no se ha tenido totalmente en cuenta la accesibilidad de algunas interacciones ni los colores corporativos entre otras cosas, por lo que ha sufrido una serie de cambios de cara a la implementación.

Los cambios principales que se han realizado después de probar la navegación han sido: añadir una barra de navegación en la parte superior de la aplicación, para así mostrar al usuario que existe una interacción, dado que en dicha barra hay un botón que al ser pulsado despliega el menú; y la incorporación de los colores corporativos.

Por otra parte, también hay cambio de estilos a nivel de *JavaScript*. Los estilos propios que se muestran en el *WebView* permiten que el editor modifique su tamaño para adaptarse a la interfaz y así poder permitir los modos *Split View* y *Slide Over*. Además, también permiten mostrar las fórmulas realizadas en el modo de handwriting ampliadas, para que de esta forma sea más fácil seleccionarlas y arrastrarlas donde sea necesario.

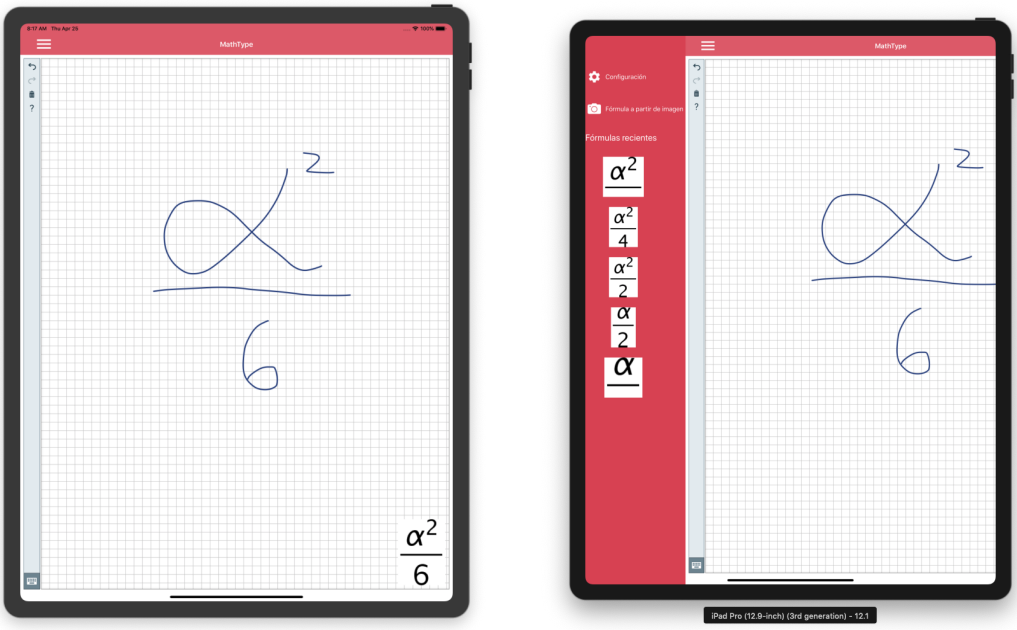


Figura 6.1: MathType for iOS en modo handwriting y a la derecha la misma pantalla con el menú desplegado. En el apéndice B se pueden observar mas imágenes.

6.1.2. Constraints visuales

Uno de los requisitos es que *MathType for iOS* sea compatible con múltiples dispositivos, ya que iOS es ejecutado por varios. Dichos dispositivos presentan diferencias que afectan al desarrollo visual, principalmente el tamaño de pantalla.

Dependiendo del tamaño pueden haber elementos que no son visualizados correctamente. Sin embargo, el análisis no tiene en cuenta la forma correcta de posicionar los elementos de la interfaz para adaptarse al tamaño de pantalla. Esto ha supuesto que durante la implementación se ha tenido que estudiar el cómo hacer esto. Al principio se usaron tamaños estáticos dependiendo del tamaño inicial, provocando que el código empezara a ser poco mantenible

dado que existen diferentes dispositivos y el desarrollo crecía de forma lineal con el número de estos.

Tras aprender más del lenguaje de desarrollo *Swift* se descubrieron nuevas formas de posicionar los elementos de la interfaz para poder refactorizar el código. A partir de entonces se implementan *constraints*.

Las *constraints* son condiciones que se aplican sobre la vista y que son relativas a elementos de la misma. Estas condiciones cumplen operadores relacionales (igualdad, mayor que, menor que... etc) y son gestionadas y actualizadas en todo momento por el sistema. Por lo tanto su uso hace que el código se reduzca y sea más mantenible.

En la siguiente figura se muestran las diferentes constraints en color naranja, que son insertadas en los elementos del menú para hacerle saber al sistema dónde se tienen que ubicar.

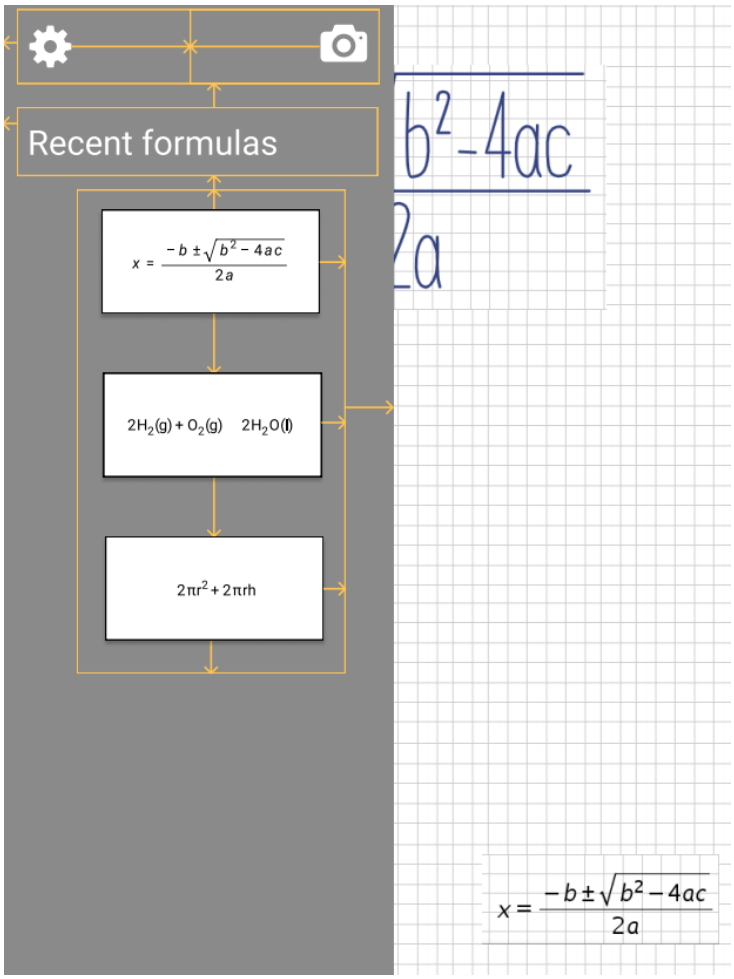


Figura 6.2: Constraints de los elementos del menú

6.1.3. Conexión con los servicios de *MathType*

Para la implementación se utilizó al principio un proyecto realizado en el pasado por el estudiante del TFG, que consistía en un framework mediante el cual poder tener los servicios del editor *MathType* en offline para *iOS*. De esta manera se quería poder eliminar la dependencia de Internet. Sin embargo, en un análisis posterior se vio la necesidad de utilizar los servicios externos de *MathType*, por lo que se prefirió no usar dicho framework y realizar todo online para así que la aplicación tuviera menos peso y siempre estuviera actualizada.

Sin embargo, la conexión con los servicios online plantean dos problemas: existe una seguridad en el editor online que no permite usarlo en un entorno que no sea un servidor web, por lo tanto no es posible cargar un archivo html en local desde el *WebView* y hay que evadir dicha seguridad; por otro lado y debido a que algunos de los servicios que utiliza *MathType for iOS* aún están en fase de desarrollo, hay un problema con la seguridad CORS[53] que también hay que solucionar.

Para el primer problema, se utiliza un servidor que actúa como *bypass* para obtener un archivo html con una dirección aceptada por el editor. Una vez esto, se borra el contenido que hubiera podido quedar antes de ser mostrado en el *WebView* y se carga el editor *MathType Web*. De esta forma, el editor detecta que está siendo ejecutado en un entorno autorizado y permite ser utilizado.

Para el segundo problema se realiza un proceso similar mediante otro *bypass*. Para acceder a los servicios de terceros es necesario que el servidor en cuestión nos retorne en la cabecera un argumento mediante el cual autorizarnos a consumir dicho contenido (si no, saltaría un error de CORS). Por lo que se utiliza un servidor externo al cual realizar la petición. Este servidor a su vez redirige la petición al servidor que queremos acceder, recoge la respuesta, le añade las cabeceras necesarias (access-control-allow-origin) y devuelve la respuesta a *MathType for iOS*.

Finalmente, hay que recordar que estas dos decisiones son consideradas malas prácticas. Sin embargo, debido a que es fácil refactorizar el código y el estado actual de desarrollo de algunos de los servicios no se ha encontrado otra opción mejor.

6.1.4. Importar fórmulas

Como se comentaba en la sección de diseño, existen limitaciones en *iOS* que afectan a la hora de importar las imágenes generadas.

El proceso interno de crear una imagen que ejecuta *MathType* se resume en los siguientes pasos:

1. **Crear la fórmula con las herramientas que proporciona el editor.** Esto lo que hace es mostrar de manera visual al usuario la fórmula, mientras que por debajo se encuentra una representación en formato *MathML* de dicha fórmula.

2. **Creación de la imagen de la fórmula.** Se crea la imagen a partir del MathML anterior.

3. **Inserción de atributos de estilo.** Se pueden añadir atributos css para el tipo de fuente, altura, anchura... etc.

4. **Inserción de atributos de control.** Se añaden atributos que permiten tener control de la fórmula. Por el momento únicamente se utilizan dos: *data-custom-editor*, que permite añadir información sobre la toolbar que se quiere mostrar al usuario durante la reedición de la fórmula; y *data-mathml*, que permite mantener la información del *MathML* que representa la fórmula.

El último paso, concretamente el añadir el atributo *data-mathml*, es el que se buscó implementar en *MathType for iOS*. Sin embargo, *iOS* consta de una limitación en el *WebView* que no se tuvo en cuenta.

El elemento *WebView* tiene implementado un comportamiento predeterminado para los eventos Drag&Drop, pero dicho comportamiento provoca que puedan exportarse las imágenes a otras aplicaciones utilizando la clase *Imagen* del sistema. Esta acción a su vez causa que se pierdan los atributos insertados en la imagen y por lo tanto no es posible saber a qué *MathML* hace referencia dicha imagen.

Debido a esto no es posible importar fórmulas directamente a *MathType for iOS*. Pero pese a ello, se ha querido aportar una forma alternativa.

Para ello se ha utilizado el servicio de reconocimiento de fórmulas de *MathType*. Las imágenes que son llevadas a *MathType for iOS* son capturadas y enviadas a dicho servicio para así, una vez con información de la fórmula que se ha detectado, poder importarse. El uso de este servicio pese a que suele funcionar bien no es del todo correcto, por lo que algunas fórmulas no son

importadas totalmente y contienen fallos. Además, la llamada a un servicio externo supone un aumento del tiempo de respuesta de cara al usuario. Por otra parte, esto permite no sólo editar fórmulas creadas por *MathType*, sino que también fórmulas de imágenes que provengan de otras fuentes.

6.1.5. Imágenes pesadas

En cuanto al uso del sistema externo de reconocimiento de fórmulas, se ha realizado un reescalado de las imágenes que se envían.

Actualmente los distintos dispositivos que ejecutan el sistema *iOS* pueden llegar a realizar fotografías con una calidad de hasta 4k. Esto en cuanto a la visualización es una gran mejora ya que se obtiene más detalles en la imagen, sin embargo a la hora de realizar la llamada al sistema externo ocurren dos problemas: se tarda más en realizar la petición y el buffer de información puede llegar a desbordarse provocando un error en el sistema.

Para evitar este tipo de problemas, las imágenes son reescaladas si superan el tamaño de 400 píxeles de anchura hasta este tamaño. Durante el escalado se rebaja la calidad de imagen y por lo tanto el tamaño de ésta queda reducido, evitando los posibles errores comentados anteriormente.

6.1.6. Referencias débiles

Durante la implementación se ha tenido que tratar numerosos errores como el de las referencias débiles comentados en el diseño de *MathType for iOS*.

Como se ve en el esquema conceptual, hay clases como *ConfigurationOption* que son enumerations que implementan el patrón factoría. Sin embargo, después de un tiempo de debug de la aplicación se ha hecho evidente de que iOS utiliza referencia débiles en las vistas que instancia dicho enumeration. Esto afecta de forma directa al comportamiento de la aplicación ya que cuando una vista que está siendo mostrada es liberada de memoria su comportamiento es imprevisible, aunque en la mayoría de casos tiende a no aparecer o a aparecer con ciertos errores de visualización.

Para evitar los problemas de dicha referencia débil, se ha creado una estructura que guarda en memoria referencias fuertes de las vistas creadas por dicha factoría.

6.1.7. Bugs de iOS

En la implementación de *MathType for iOS* se han encontrado bugs en distintos elementos de la interfaz. Estos han afectado principalmente al diálogo

para elegir entre si se quiere hacer una fotografía o elegir una imagen tomada anteriormente, para reconocer en ella una fórmula.

Parece ser que es algo reportado a *Apple* pero que aún no tiene solución oficial. Dicho diálogo permite ser mostrado con estilos distintos, desde un pop-up normal a una ventana que se extiende desde la parte inferior de la pantalla. Sin embargo esta última provoca un error en las *constraints* presentes. La solución ha sido no utilizar el último estilo comentado y mostrar el pop-up normal.

6.1.8. Cambio de formato de imagen

Durante el diseño no se tuvo en consideración si el sistema era capaz de utilizar diferentes formatos de imagen. Esto ha provocado que a la hora de implementar dicha feature se haya descubierto que *iOS* no permite el uso de su clase *Imagen* en formato SVG como se quería, por lo que esta feature queda descartada.

El formato SVG proporciona una mejor calidad en las fórmulas renderizadas que el formato PNG. Sin embargo, para minimizar el impacto se ha decidido aumentar la calidad de las imágenes PNG de 90dpi a 400dpi, ya que después de varias pruebas se ha visto que el impacto en el tiempo de respuesta de la aplicación no es apreciable por el usuario.

6.2. Automatización de procesos

6.2.1. Herramientas utilizadas

Actualmente, *Maths for More S.L.* se encuentra realizando un esfuerzo para mejorar la mantenibilidad de sus proyectos respecto al proceso de distribución. Dicho esfuerzo se basa en invertir recursos en la automatización de algunas de las etapas, como por ejemplo la generación de los paquetes software finales que serán distribuidos a los clientes y también la verificación de la calidad o *testing*. Además de aprovechar todo ello para configurar un entorno donde poder aplicar el concepto de *integración continua*[30].

Se entiende como *integración* la compilación y ejecución de tests para comprobar el buen funcionamiento del proyecto. Y de este concepto nace la idea de *integración continua* como el acto de hacer integraciones automatizadas de forma periódica, normalmente cuando hay un cambio en el proyecto. De esta manera, es posible detectar posibles fallos rápidamente.

MathType for iOS no es la excepción y se le pide el mismo nivel de compromiso para mejorar la mantenibilidad, intentando utilizar las herramientas que ya se utilizan en los demás proyectos internos.

Por lo tanto, el estudiante del *TFG* tiene que aprender a utilizar dichas herramientas. Cada una tiene una finalidad distinta dependiendo de la etapa que se quiera automatizar y son las siguientes:

- **Git[9].** Existen varios motivos para utilizar esta herramienta. Primero de todo permite llevar un historial de los cambios en el código, por lo que resulta sencillo deshacer cambios y llevar un control de quién ha modificado algo. Por otra parte, es utilizado ampliamente haciendo uso de un servidor interno o externo donde se suben los cambios que puedan haberse hecho de manera local, para así propagarlos a todas las demás personas que trabajen con el mismo repositorio.

- **Apache Ant[3].** Esta es una herramienta de automatización de procesos de compilación. A pesar de haber más y mejores en el mercado se ha escogido debido a que los demás proyectos de *Maths for More S.L.* la utilizan.

- **Appium[28].** Framework open source para la automatización de tests en aplicaciones móviles. Esta es la única herramienta nueva que utiliza este proyecto respecto a los demás de *Maths for More S.L.*. Esto se debe a que es la primera aplicación móvil y no se cuenta con ninguna herramienta previa.

- **Jenkins[29].** Esta herramienta es un servidor que permite ser configurado para implementar el modelo de *integración contínua* explicado anteriormente.

6.2.2. Automatización de la compilación

Una de las partes a automatizar de un proyecto es la compilación. Esto ayuda a generar los paquetes que podrán ser distribuidos a los clientes con facilidad. Además, ayuda al mantenimiento de un proyecto gracias a que el *script* de automatizado hace que quede constancia del cómo hay que hacerse, por lo que en caso de que la persona que sabía compilar el proyecto se marche, éste podrá ser continuado por otra sin que la primera haya tenido que dejar documentación previa.

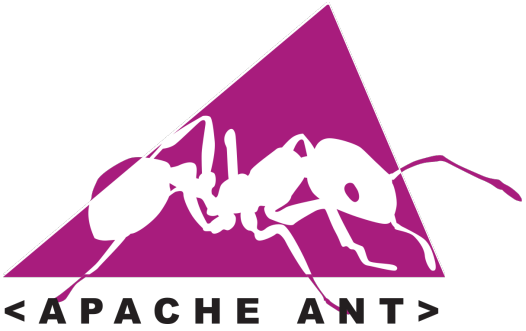


Figura 6.3: Logo de Apache Ant.

Como se comentó anteriormente, *Apache Ant* ayuda a la automatización de procesos de compilación, por lo que será utilizada para este fin. Dicha herramienta tiene multitud de acciones genéricas mediante las cuales puede llevarse a cabo esta tarea: crear directorios, ejecutar scripts por línea de comandos, copiar archivos... etc.

En el caso de este proyecto, la automatización se ha realizado a nivel de proyecto de *XCode*, la interfaz de desarrollo que proporciona *Apple* para programar en *iOS*.

Cuando se dice "nivel de proyecto" se refiere a que no existen recursos comunes a otros proyectos internos de *Maths for More S.L.*. Ya que, en caso de que hubieran recursos comunes, como por ejemplo imágenes corporativas susceptibles de cambios, habría que mantener dichos recursos a parte y que los proyectos que los usaran los copiaran. De esta forma, en caso de cambio de un recurso, bastaría con hacer un cambio en el recurso y todos los proyectos que dependen de él descargarían el nuevo con los cambios.

Entonces, para automatizar el proyecto, primero ha habido un periodo de autoaprendizaje de *Apache Ant* y las herramientas de desarrollo de *XCode*, además de la lectura de documentación para saber qué acciones pueden llevar a cabo. Después de ello, se ha creado en la raíz del proyecto un archivo *build.xml* en el cual, utilizando lenguaje *XML* se pueden definir las tareas que se quieren ejecutar en *Apache Ant*, como el que se puede ver a continuación.

Figura 6.4: Documento *XML* de Apache Ant.

En el caso de este *TFG*, las tareas han consistido en usar el entorno de líneas de comandos de *XCode* y crear las propiedades necesarias, como la localización del paquete resultante, para facilitar el desarrollo durante la automatización de tests.

Una vez terminado, el proyecto está listo para compilarse con una línea de código: *ant build*.

6.2.3. Automatización de los tests

En cuanto a la automatización de tests, va íntimamente ligada a la automatización de la compilación, ya que depende de que exista un paquete previo donde poder ejecutarse.

Figura 6.5: Logo de Appium.

Como se explicaba al principio, se ha escogido *Appium* como herramienta de automatización de tests, ésta requiere de dos servicios propios corriendo en la máquina destinada al *testeo*, pero que se ha obviado su automatización dado que forma parte de la configuración de cada máquina y una vez configurados no hay que volver a hacer nada.

Dicha herramienta aporta un framework implementado en el lenguaje *Java* donde se permiten crear tests de *JUnit*[51]. Además, como se puede ver en la siguiente figura, tiene un *inspector* para mejorar la tarea de *debug* de la aplicación y también la propia creación de tests, ya que permite entre otras cosas crear comandos sobre los elementos de la interfaz.

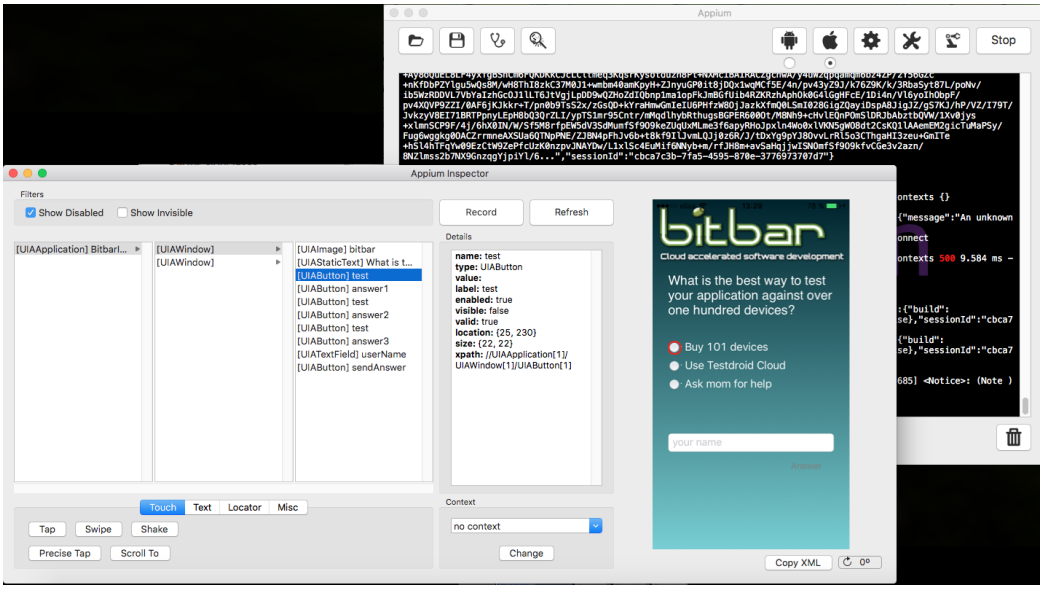


Figura 6.6: Webinspector de Appium.

Por este motivo, el archivo *build.xml* comentado en la sección anterior ha sido ampliado para añadir una nueva tarea que se ejecute con un *ant test*, que pasa los tests. Esta tarea no está incluida en el comando *ant build* comentado anteriormente para que, como desarrollador, sea posible elegir hace lo uno o lo otro.

Llegados aquí, el archivo *build.xml* tiene los dos comandos: *ant build* y *ant test*, que además pueden ejecutarse uno detrás de otro con un *ant build test*.

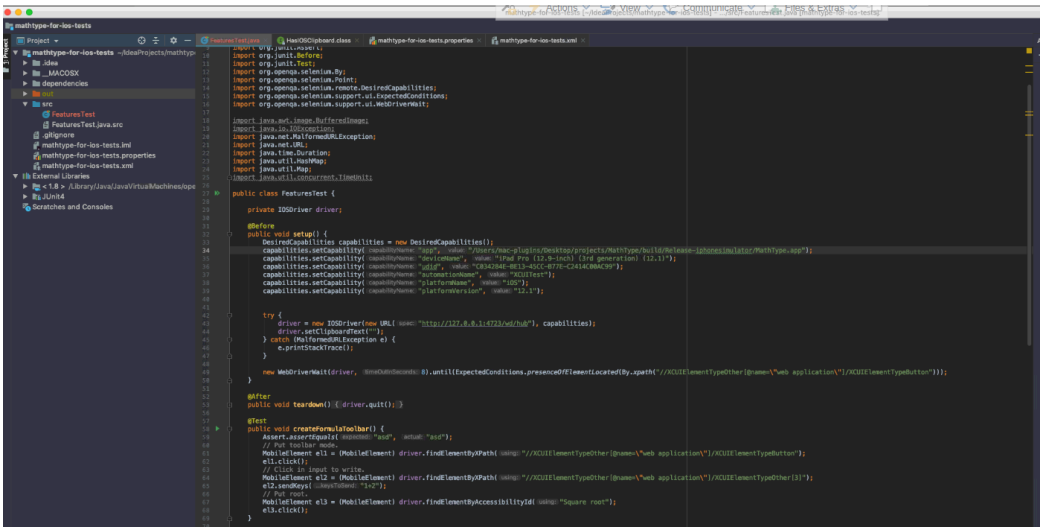


Figura 6.7: Test de JUnit en Appium.

Como comentarios adicionales, cabe destacar que para la creación de las tareas, se ha decidido mantener dos repositorios para el proyecto de *MathType for iOS*: uno contiene el desarrollo de la aplicación y otro el sistema de *testing*. Cuando el desarrollador realiza un *ant test* se descarga todo el sistema de tests y se ejecutan para comprobar la calidad del software. De esta manera, siempre se mantienen actualizados todos los tests.

6.2.4. Integración contínua

Figura 6.8: Logo de Jenkins.

Para terminar, se hablará de la última parte de la automatización, la máquina que ejecutará el proceso de *integración contínua*. Para ello, se utiliza la herramienta *Jenkins*.

Jenkins es una aplicación que se ejecuta en un servidor. Ésta ofrece un servicio de "nodos", que son máquinas (a veces también llamadas *slaves*) que pueden conectarse a dicho servidor para que éste descargue en ellas los repositorios de los proyectos que interesan y se ejecuten las tareas de compilación y testeo. De esta forma, se pueden mantener máquina independientes con características específicas que comprueban el buen estado de todos los proyectos en sus respectivos repositorios de *Git*.

Para hacer esto, se ha creado un nuevo archivo en la raíz del proyecto llamado *Jenkinsfile*. Este archivo guarda la configuración que debe tener respecto al proyecto el servidor Jenkins cuando éste quiera hacer pruebas del repositorio en cuestión.

En este archivo entonces, se ha definido dos cosas principalmente: el nodo (máquina) en el cual tiene que ejecutarse la compilación y testeo; y las fases (o *stages*) que se llevarán a cabo.

En el caso de *MathType for iOS* solamente hay dos fases: *Build* y *Test*. En cada una se ejecutan los comandos análogos de ant definidos previamente. Sin embargo, en un futuro se espera aumentar dichas fases para añadir pasos extra como la automatización que publique directamente el paquete en la *AppStore* de *Apple* cuando los procesos hayan finalizado con éxito.

Una vez terminado todo este trabajo de configuración, el resultado se puede ver en la siguiente figura. Se muestran cada una de las pruebas que se han pasado.

Además, dichas pruebas han sido configuradas a conciencia para que sean ejecutadas cada vez que existe un cambio en el repositorio de *Git* principal. Por lo que con cada cambio de código se podrán tener avisos de si existen o no errores.

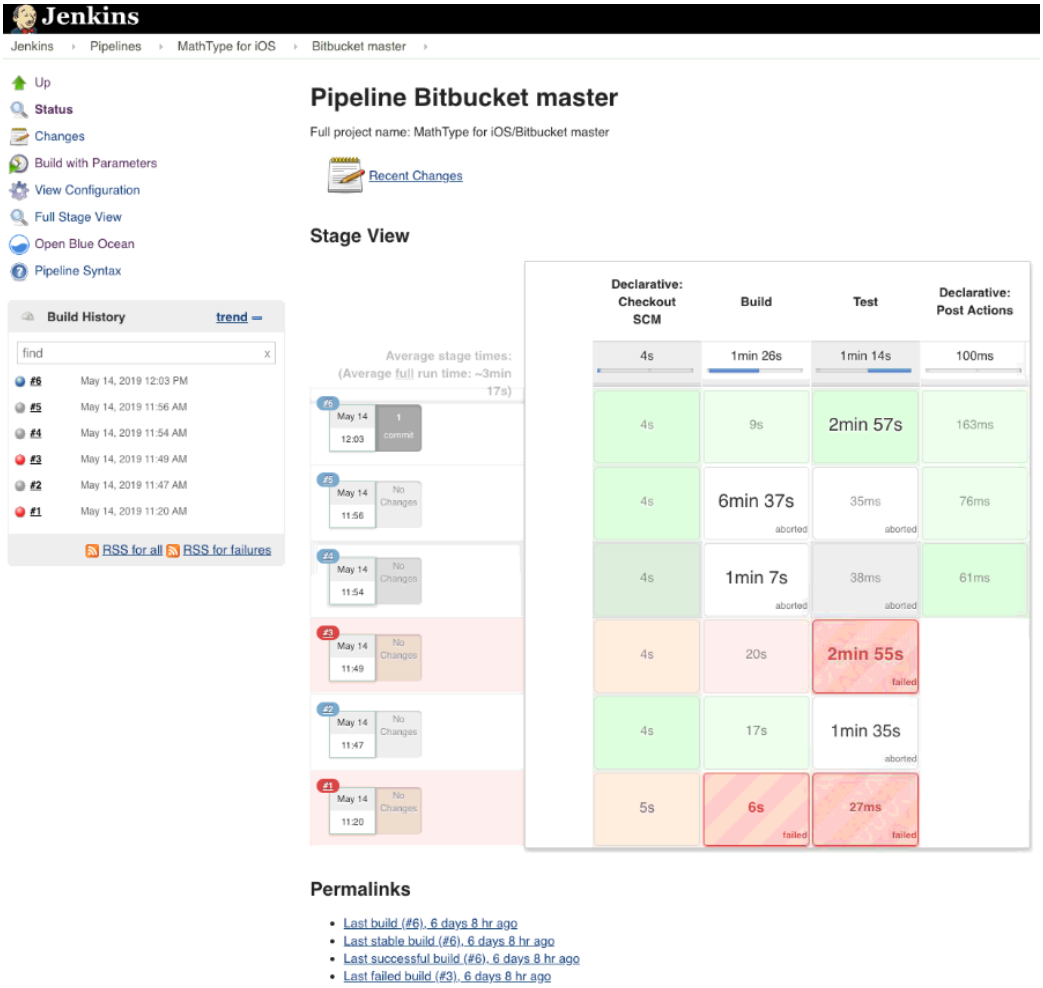


Figura 6.9: Resultados de los tests de *MathType for iOS* en *Jenkins*.

6.2.5. Diagrama de desarrollo

Terminando de explicar, falta detallar el diagrama de desarrollo que representa la puesta en producción.

Cuando se habla de diagrama de desarrollo se hace referencia a dónde se ejecuta cada pieza del conjunto software, ya que puede ser contenido en una o en diferentes máquinas que se comunican entre sí.

En el caso de *MathType for iOS* se utiliza una arquitectura a la que se le ha dado el nombre de *ASA (Aplicación, Servicios y Automatización)*, debido a que el código se ejecuta en las primeras dos partes, mientras que a la tercera se la hace responsable de la automatización del paquete a distribuir y el *testing* del mismo.

En la siguiente figura, se puede observar el diagrama de desarrollo del sistema.

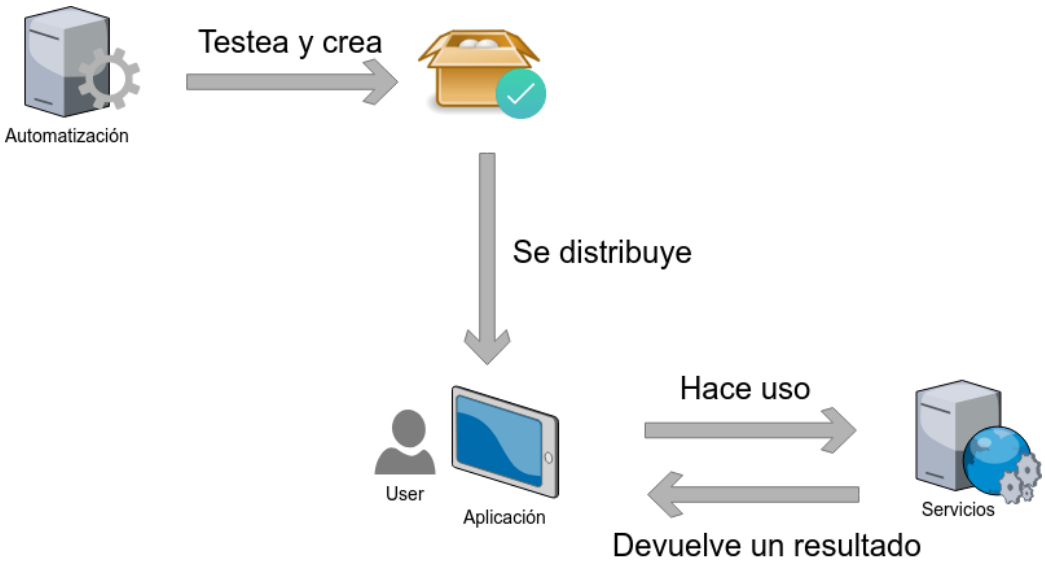


Figura 6.10: Diagrama de desarrollo de *MathType for iOS*.

Aplicación La aplicación es el elemento en el cual se ejecuta el programa. En el caso de *MathType for iOS* se hablaría del dispositivo *iOS* con la aplicación instalada que es abierta por el usuario.

Dicho elemento contiene la aplicación *MathType for iOS* y es el responsable directo de la interacción con el usuario que quiere utilizar el sistema.

Servicios Sin embargo existe parte de la lógica del sistema, como el reconocimiento de una fórmula a partir de una imagen, que no es posible que se ejecute en el propio dispositivo debido a que se escapa del dominio del TFG. Entonces, para suplir esta necesidad se utilizan servicios externos ya creados por *Maths for More S.L.*.

Estos servicios externos son ejecutados en el servidor de la empresa y realizan de forma independiente el trabajo de cómputo del resultado, que será devuelto a la aplicación.

Automatización Finalmente, debido a que una de las metas que se persiguen es la entrega de un software de calidad, debe existir un elemento físico del sistema que se encargue de realizar las tareas de comprobación de *MathType for iOS*.

Para ello, dicho elemento físico genera de manera automática los paquetes que se distribuirán y a su vez genera contextos análogos a los que se tienen en producción. En estos contextos se ejecuta una simulación de un dispositivo con el sistema *iOS* y se pasan los tests explicados anteriormente con la intención de comprobar la ausencia de errores.

Capítulo 7

Validación

Como parte esencial del proyecto existe la validación, fase que permite saber si los tests del paquete generado han pasado y si los requisitos analizados en un principio han cumplido sus criterios de satisfacción con la calidad esperada o mayor.

Para ello, primero de todo se utiliza el entorno de *testing* realizado durante la implementación del proyecto, para observar si los tests que comprueban aspectos específicos de la aplicación pasan correctamente.

En cuanto a los requisitos funcionales, todos han sido cumplidos menos el ya explicado a lo largo del proyecto sobre el cambio de formato de imagen. Sin embargo, esta parte no requiere extenderse más debido a que dichos requisitos plantean funcionalidades que deben ser implementadas y que solamente existen dos respuestas: sí, ha sido implementada; o no, no ha sido implementada. Además, todas ellas son explicadas a lo largo del trabajo de una manera más profunda.

Por otra parte, algunos de los requisitos no funcionales sí que necesitan una explicación más extensa por dos motivos: se han realizado entrevistas en las cuales se ha podido conseguir *feedback* de personas reales y también porque los criterios de satisfacción son criterios subjetivos, como por ejemplo si la aplicación es bonita o no, y cuya respuesta puede ser más elaborada que un sí o un no.

7.1. Proceso de las entrevistas

Para intentar obtener datos representativos y fiables, se ha querido documentar el proceso llevado a cabo durante las entrevistas. De esta forma, es posible detectar en trabajos futuros errores o mejoras que se podrían hacer

respecto a la entrevista o también detectar si realmente eran válidas para el objetivo que se quería.

Primero de todo el objetivo de las entrevistas ha sido el de saber si los requisitos no funcionales han sido cumplidos. Muchos de ellos hacen uso de la valoración subjetiva de diferentes personas que, para este caso, hemos escogido de los departamentos de la empresa *Maths for More S.L.*.

Como primera aproximación, y que se espera mejorar en un futuro, se han escogido a un total de 5 personas donde: dos personas formaban parte del equipo de integración del producto *MathType* en plataformas de terceros, dos del equipo que elabora el editor *MathType* y uno que es el director del proyecto.

El proceso que se ha llevado a cabo en todas las entrevistas ha sido el de seguir los siguientes casos de uso en un dispositivo *iPad Pro 2018 11"*.

1. Crear una fórmula en modo *handwriting*, exportarla a otra aplicación, importarla de nuevo y editarla.
2. Escoger una imagen donde poder reconocer una fórmula.
3. Crear una fórmula en el modo con toolbar de herramientas.

7.2. Resultado de las entrevistas

De las entrevistas se ha obtenido un resultado y a continuación se dividen por párrafos los diferentes requisitos no funcionales y se comenta el comportamiento que han tenido las diferentes personas mientras hacían uso de *MathType for iOS*.

Requisito no funcional 001: Diseño atractivo. Este requisito ha sido el más comentado a lo largo de las entrevistas, algo que se ve justificado debido a que es donde menos experiencia tiene el estudiante. A pesar de que 4 de las 5 personas han dado su visto bueno, el director de proyecto no se ha mostrado del todo conforme. El cómo se muestran las fórmulas recientes en el menú lateral no ha terminado de convencer debido a que si se abre el menú existe un gran hueco vació en la esquina inferior izquierda si no se han creado fórmulas recientemente. Sin embargo, en general el diseño se ve simple, claro y suficiente.

Requisito no funcional 002: Estilo autoritario. Todos los entrevistados se han mostrado conformes con los resultados.

Requisito no funcional 003: Usabilidad. En cuanto a este requisito todos se han mostrado relativamente conformes y sólo han criticado los estilos del propio editor *MathType* y que por ello se escapan del dominio de este proyecto. La principal queja ha sido la toolbar de herramientas parece ser demasiado pequeña y dificulta seleccionar las opciones si se va muy rápido.

Requisito no funcional 004: Lengua. Todos los entrevistados se han mostrado satisfechos con los idiomas que se pueden escoger (inglés y español). Sin embargo, se ha comentado que debido a la muestra de población en la que se realiza la entrevista esto no siempre es así. La muestra utilizan esos dos idiomas en su día a día, sin embargo si se decidiera publicar en el mercado *MathType for iOS* posiblemente haría falta tener más idiomas. Esto, a pesar de ser cierto, no es un problema para este proyecto dado que la arquitectura permite hacer un cambio de idioma de forma sencilla.

Requisito no funcional 005: Aprendizaje. Para terminar, este último requisito ha sido el mejor valorado. Todos los entrevistados han sentido que es muy fácil de aprender a utilizar y se les hace intuitivo, sobretodo por utilizar el modo *handwriting* como método de entrada predeterminado.

Capítulo 8

Planificación temporal

La planificación de un proyecto es parte esencial de la gestión del mismo. Una buena planificación puede ser decisiva para lograr acabar dentro de los plazos necesarios y por lo tanto cumplir los objetivos planteados.

Este proyecto TFG tiene una duración de 540h, siguiendo así la normativa académica aprobada en 2015[14]. De las cuales 75h son de la asignatura de GEP y las 465h restantes quedan para realizar el proyecto de forma totalmente autónoma. El inicio tiene fecha 18/01/2019 y la fecha máxima es la de su defensa, en junio o julio del mismo año, aunque se ha estimado que para el 05/05/2019 ya estará acabado.

La finalidad de este capítulo es la de explicar las diferentes fases que se esperan realizar y las tareas asociadas. Además de aproximar el gasto de recursos en cada tarea y definir la hoja de ruta a seguir.

8.1. Recursos necesarios

En esta sección se muestran los recursos asignados al desarrollo de cada una de las siete fases en las que queda dividido el proyecto y sus tareas asociadas. Cabe destacar que recursos como el espacio de trabajo y mobiliario han sido obviados dado que su uso es continuo a lo largo de todo el proyecto.

8.1.1. Descripción de fases y tareas

Asignatura GEP

La siguiente fase consta de la elaboración de los entregables de la asignatura GEP, que ayudan a contextualizar, planificar y explicar el proyecto. Además se tiene en cuenta las sesiones presenciales y la presentación final que hay que

realizar. El tiempo de esta fase se ha estimado según el global de horas de la asignatura (75h) y una estimación personal después de realizar el primer y segundo entregable. Hay que destacar que esta fase provoca una pausa en el proyecto empezado ya en enero.

Análisis

En esta fase del proyecto se realiza el análisis técnico de la aplicación desde el punto de vista de la ingeniería de software para su posterior implementación. Por ello, las principales tareas que se tienen que realizar son la toma de requisitos, el diseño y especificación de la arquitectura y el análisis del diseño de la interfaz.

Preparación del entorno de trabajo

Otra fase del proyecto es preparar el entorno de trabajo. Esta fase tiene en cuenta la instalación y configuración del entorno de trabajo necesario para empezar a desarrollar tanto en local como en los servidores de integración continua de la empresa *Maths for More S.L.* . Las principales tareas por lo tanto son: la preparación del ordenador de desarrollo y la preparación de los servidores de integración continua.

Pruebas de concepto y automatización

Para el buen desarrollo del proyecto es necesario realizar diferentes pruebas de concepto de las cuales elegir las mejores para implementar en el proyecto final. Todo ello ocupa una fase del proyecto enfocada al aprendizaje, pruebas y automatización. Por lo tanto las tareas relacionadas son: el autoaprendizaje de las diferentes tecnologías a utilizar; las pruebas de concepto para así asentar los conocimientos adquiridos y ver cómo adaptar la tecnología a la especificación del proyecto; y la automatización del compilado y testeo de los proyectos de la tecnología utilizada.

Desarrollo de *MathType for iOS*

Esta fase es la más grande de todas, es la parte principal del proyecto y la que genera el producto *MathType for iOS* que la empresa precisa.

Primero de todo se encuentra la tarea de creación de la aplicación base para iOS. De ella se obtiene la integración del editor MathType Web en iOS.

A raíz de ello, se realizan una serie de tareas que extienden una a una las funcionalidades que se ofrecen, para así cumplir aquellos objetivos planteados en la formulación del problema.

Concretamente serían: el importar y exportar fórmulas mediante el uso de *Drag&Drop*, la reedición de fórmulas creadas, la visualización de fórmulas recientes, el cambio de tipo de imagen, el uso de la cámara para detectar fórmulas y el soporte a distintos idiomas.

Las tareas que extienden las funcionalidades de la aplicación son completamente independientes y pueden realizarse en paralelo o en el orden que se considere oportuno. El orden de realización de éstas es el que se puede observar en la Tabla 8.1. Este orden ha sido escogido según las preferencias de la empresa.

Documentación

Una de las fases finales del proyecto es la documentación. Esta fase es la encargada de generar toda la *documentación de usuario*, para que así las personas que utilizan *MathType for iOS* puedan consultar cómo utilizar el programa en detalle.

Memoria y presentación

En esta fase se terminará la memoria del proyecto que se ha ido haciendo durante la asignatura de GEP y se revisará. Finalmente, se preparará la presentación de la defensa del proyecto.

La memoria del proyecto incluirá la documentación realizada a lo largo de la asignatura de GEP, la fase de análisis y la fase de documentación.

8.1.2. Estimación de horas y recursos

Con la explicación de cada fase realizada anteriormente, la estimación de tiempo y recursos es la siguiente.

Tabla 8.1: Planificación temporal de fases, tareas y recursos.

Fases y tareas	Recursos humanos	Recursos materiales	Recursos software	Horas
Asignatura GEP				**75**
Clase de presentación inicial	· Jefe de proyecto	Ordenador portátil Dell Inspiron 7537 i7 2,66GHz 8GB RAM 500GB SSD	Google Docs, Google Sheets ,Google Slides, TexStudio	2
Entregable 1				25
Entregable 2				12
Entregable 3				12
Entregable 4				20
Ensayo de presentación final				2
Clase de presentación final				2
Análisis				**60**
Toma de requisitos	· Analista software	Ordenador portátil Dell Inspiron 7537 i7 2,66GHz 8GB RAM 500GB SSD	Windows10, VS Code, Draw.io	30
Diseño y especificación software	· Arquitecto Software·			15
Diseño de la interfaz	· Diseñador de Interfaces			15
Preparación del entorno de trabajo				**60**
Instalación y configuración del ordenador de desarrollo	· Desarrollador Software	Ordenador sobremesa Mac Mini i5 3GHz 8GB RAM 256GB SSD	MacOS Mojave 10.14, VS Code, XCode, Apache Ant, Jenkins	30
Instalación y configuración de los servidores de integración continua				30
Pruebas de concepto y automatización				**85**
Autoaprendizaje	· Desarrollador Software	Ordenador sobremesa Mac Mini i5 3GHz 8GB RAM 256GB SSD	MacOS Mojave 10.14, XCode	20
Pruebas de concepto				35
Automatización del build y testeo de aplicaciones				30

Desarrollo de MathType for iOS				**185**
Integrar MathType en iOS	· Desarrollador Software	· Ordenador sobremesa Mac Mini i5 3GHz 8GB RAM 256GB SSD · iPad Pro 2018 11" 64GB	MacOS Mojave 10.14, VS Code, XCode, Apache Ant, Jenkins	20
Importar/Exportar imágenes				15
Reedición de fórmulas				40
Mostrar fórmulas recientes				30
Cambio en el tipo de imagen				20
Uso de la cámara para la detección de fórmulas				30
Soporte para diferentes idiomas	+Traductor			30
Documentación				**25**
Documentación de usuario de *MathType for iOS*	· Escritor técnico · Desarrollador Software	Ordenador portátil Dell Inspiron 7537 i7 2,66GHz 8GB RAM 500GB SSD	VS Code	25
Memoria y presentación				**50**
Memoria del proyecto	· Jefe de proyecto	Ordenador portátil Dell Inspiron 7537 i7 2,66GHz 8GB RAM 500GB SSD	TexStudio, Google Slides	42
Presentación del proyecto				8
TOTAL				**540**

8.2. Diagrama de Gantt

En el siguiente diagrama de Gantt, correspondiente a la planificación del proyecto, se incluyen todas las fases y tareas especificadas anteriormente. Este diagrama muestra los tiempos y también los órdenes de realización de las diferentes tareas durante el proyecto teniendo en cuenta un trabajo semanal de 35h.

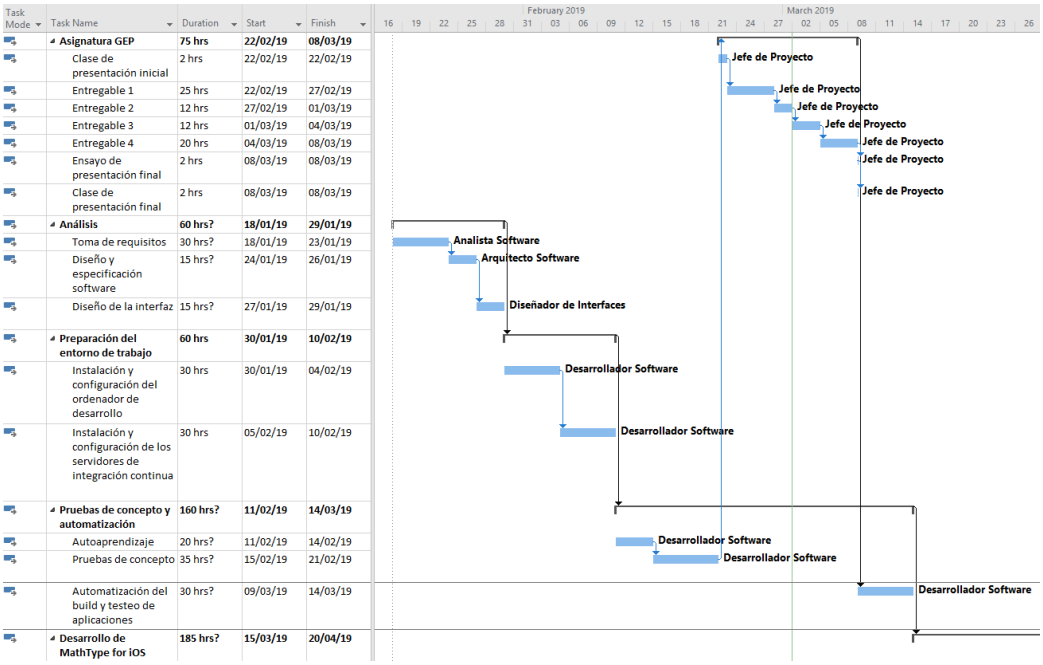


Figura 8.1: Diagrama de Gantt.

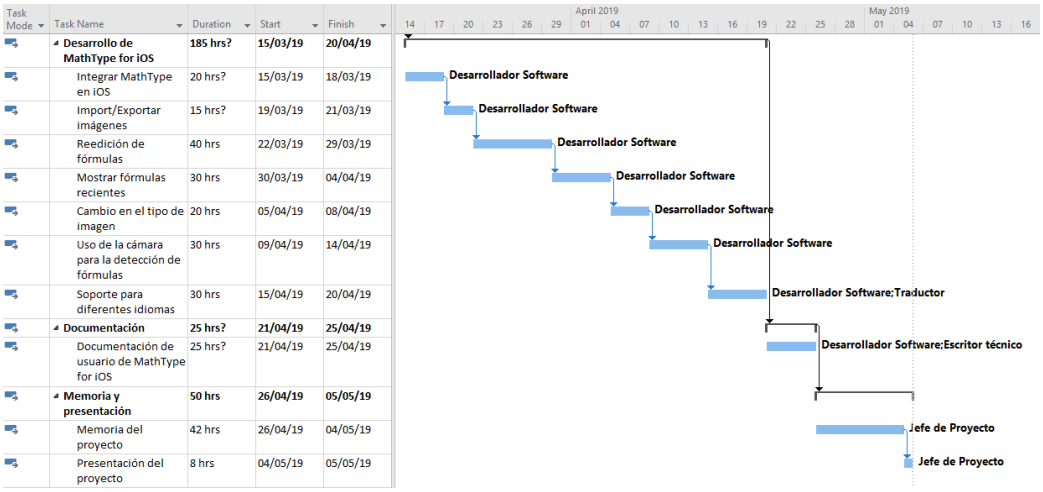


Figura 8.2: Continuación diagrama de Gantt.

8.3. Posibles desviaciones y plan de acción

Como se comentó en otros capítulos, los riesgos de este proyecto se deben a la tecnología. Se hace uso de diferentes tecnologías donde el integrador del

proyecto no es experto. Además existen tareas como la de documentación o diseño que se salen del ámbito de trabajo del estudiante.

Todo esto provoca un riesgo de generar una desviación de tiempo, aumentando así los tiempos de realización de las tareas. Este riesgo se ha estimado con una probabilidad de un 30 % por lo explicado anteriormente y también porque el estudiante tiene experiencia integrando MathType en otras plataformas. Entonces, a pesar de tardar más en tareas como el diseño de la interfaz, también puede ocurrir el efecto contrario en otras como la integración de *MathType* en iOS.

El plan de acción de este proyecto está basado principalmente en el control de situaciones de riesgo. Por ello, durante la realización del proyecto hay un seguimiento constante del trabajo realizado y del tiempo restante.

Para dicho seguimiento se han establecido dos protocolos de acción en caso de detectar una falta de tiempo.

Por una parte, todas las tareas han sido sobreestimadas en un 15 % para tener margen de maniobra en caso de haber estimado el tiempo por debajo de la realidad.

Por otra parte, está planteada también la posibilidad de aumentar el número de horas dedicadas al proyecto. Actualmente se realizan 5h diarias, podría llegar a subirse hasta en 7h diarias.

En caso de que lo anterior fallara, se tomaría como medida de urgencia eliminar tareas de la fase de *Desarrollo de MathType for iOS*. Ya que el soporte de idiomas, el elegir el tipo de imagen y también la sección de "fórmulas recientes" no son necesarios para el correcto funcionamiento de la aplicación. Por lo tanto, se puede prescindir de ellas y que la aplicación siga siendo útil y funcional para el usuario.

Estas decisiones propuestas permitirían la entrega del proyecto en el plazo establecido. Sin embargo, el costo de recursos humanos se incrementaría un 40 % por hora desde el momento en el que se aumentaran las horas diarias. Y por otro lado, la aplicación de la medida de urgencia, pese a funcionar, disminuiría la calidad del producto final.

8.4. Desviaciones durante el proyecto

Durante la implementación del TFG, se han sufrido cambios en la planificación de las tareas debido a que ésta era una mera aproximación y, en general, las tareas han necesitado un mayor tiempo de desarrollo que el estimado en un primer momento.

Dichas desviaciones se han producido sobretodo en las partes donde el estudiante del TFG es menos experto. Un ejemplo de ello son las tareas relacionadas con el diseño de la aplicación y también aquellas que tienen que ver con la documentación de la memoria.

Además, la exigencia en cuanto a la carga de trabajo junto a las responsabilidades diarias del estudiante, han terminado por volverlo ineficiente, por lo que se ha decidido bajar la carga de trabajo a 25h semanales en vez de las 35h.

Esta decisión ha provocado un retraso en la fecha de finalización, pero ha podido ser asumido gracias al margen de tiempo dejado como medida de contingencia. Por este motivo, ninguna tarea ha tenido que ser eliminada para poder alcanzar la deadline propuesta por la facultad en cuanto a la entrega del proyecto.

8.4.1. Cambios respecto al inicio del proyecto

En el trascurso del proyecto, han habido distintas desviaciones temporales. Una de las causas principales ha sido la inversión de tiempo en el autoaprendizaje de las distintas tecnologías. Al tratarse de una integración entre varios sistemas la complejidad de saber el funcionamiento de éstos ha sido mayor de lo esperado.

Primero de todo, se ha tenido que aprender a usar con soltura y con buenas prácticas los lenguajes de programación Swift y JavaScript, aunque debido a que el estudiante tiene ya experiencia en este último ha podido reducir el número de horas. Para ello, se ha leído documentación y se han creado distintas aplicaciones para saber aplicar la teoría aprendida.

Después, se ha tenido que aprender de la misma forma lo relativo a la parte de administración de sistemas para automatizar el proceso de *build* y *testing* utilizando tecnologías como *Ant* o *Jenkins* junto a las instrucciones propias del sistema *Windows* que se utiliza en *Maths for More S.L.* .

Además, durante el desarrollo se han encontrado con limitaciones del sistema que han puesto en duda la viabilidad del proyecto debido a que han afectado directamente a las funcionalidades de reedición de fórmulas y el cambio de tipo de imagen.

En cuanto a la limitación del cambio de tipo de imagen se quería ofrecer la posibilidad de elegir entre PNG o SVG, ya que este último a pesar de tener un mayor peso y calidad no es aceptado por todas las aplicaciones. Sin embargo, *iOS* solamente permite el uso de imágenes PNG para ser exportadas a otras aplicaciones dentro del mismo sistema.

Después de investigar durante un tiempo, no se han encontrado alternativas, por lo que se ha decidido aceptar que no es posible realizar dicha tarea y se ha pasado a minimizar las consecuencias. Para ello, se ha aumentado la calidad de las imágenes creadas por el editor al máximo posible. Dicho máximo ha sido encontrado incrementando la calidad hasta que el usuario nota un aumento del tiempo de respuesta del sistema debido al peso de la imagen.

Además, debido a la rapidez con la que se detectó y se aplicó la solución, se crearon dos nuevas tareas para permitir cambiar la barra de herramientas mostrada por el editor y también para insertar elementos que mejoraran el uso de la interfaz. De esta última se creó un botón, para facilitar el copiado de una fórmula; y el incremento de las fórmulas que aparecen en el editor, para que fuera más fácil agarrar una fórmula creada y utilizar la funcionalidad de Drag&Drop.

Por otra parte el problema de la reedición de fórmulas provoca que, al no poder insertar información extra en la fórmula, *MathType for iOS* no puede saber a qué representación en formato *MathML* hace referencia la imagen en el caso de uso de exportar una imagen y volverla a importar.

Se encontraron dos alternativas para insertar información en las imágenes: aplicar estaganografía en las imágenes exportadas, para después poder ser leídas; o utilizar un nuevo servicio de reconocimiento de fórmulas de *MathType*, que es capaz de reconocer a partir de imágenes de fórmulas el *MathML* que las forma.

Dado que el reconocimiento puede fallar, se investigó la primera opción. Sin embargo, tras ver que las imágenes eran transformadas también por el sistema durante la exportación, se decidió que la segunda opción era más factible.

La justificación de ello es el riesgo que suponía tener que invertir más tiempo sin saber siquiera si podía encontrarse una solución viable. La solución de la esteganografía podía tener tanto peso como un TFG, dado que no es trivial hallar una forma de insertar información en imágenes y que los datos no se vean afectados por transformaciones como la rotación o escalado.

Otro cambio que ha sufrido la planificación ha sido el movimiento de la tarea de automatización del *build* y *testeo* de *MathType for iOS* hacia el final de la fase de desarrollo, debido a los cambios continuos de la interfaz del proyecto durante toda la implementación, ya que cada cambio hubiera provocado que los tests hubieran tenido que ser rehechos.

Y ya para terminar, se ha decidido cambiar una de las tareas finales del proyecto de *Documentación de usuario de MathType for iOS* por la tarea

de *Documentación de desarrollo de MathType for iOS*, ya que se considera más útil explicar de forma breve cómo hay que desarrollar dentro del sistema creado para realizar tareas de mantenimiento sin romper la arquitectura de la aplicación.

Y ahora, se muestra la tabla con la planificación y el diagrama de Gantt actualizado con los cambios comentados.

Tabla 8.2: Estimación de las tareas después de los desvíos.

Fases y tareas	Recursos humanos	Recursos materiales	Recursos software	Horas
Asignatura GEP				**75**
Clase de presentación inicial	· Jefe de proyecto	Ordenador portátil Dell Inspiron 7537 i7 2,66GHz 8GB RAM 500GB SSD	Microsoft Project, Google Docs, Google Sheets ,Google Slides, TexStudio	2
Entregable 1				25
Entregable 2				12
Entregable 3				12
Entregable 4				20
Ensayo de presentación final				2
Clase de presentación final				2
Análisis				**70**
Toma de requisitos	· Analista software	Ordenador portátil Dell Inspiron 7537 i7 2,66GHz 8GB RAM 500GB SSD	Windows10, VS Code, Draw.io	30
Diseño y especificación software	· Arquitecto Software			15
Diseño de la interfaz	· Diseñador de Interfaces			25
Preparación del entorno de trabajo				**50**
Autoaprendizaje				20
Instalación y configuración del ordenador de desarrollo	· Desarrollador Software	Ordenador sobremesa Mac Mini i5 3GHz 8GB RAM 256GB SSD	MacOS Mojave 10.14, VS Code, XCode, Apache Ant, Jenkins	15
Instalación y configuración de los servidores de integración continua				15
Pruebas de concepto y automatización				**70**
Autoaprendizaje	· Desarrollador Software	sobremesa Mac Mini i5 3GHz 8GB RAM	MacOS Mojave 10.14, XCode	30
Pruebas de concepto				40
Desarrollo de MathType for iOS				**200**
Integrar MathType en iOS	· Desarrollador Software	· Ordenador sobremesa Mac Mini i5 3GHz 8GB RAM 256GB SSD · iPad Pro 2018 11" 64GB	MacOS Mojave 10.14, VS Code, XCode, Apache Ant, Jenkins	20
Importar/Exportar imágenes				20
Reedición de fórmulas				40
Mostrar fórmulas recientes				30
~~Cambio en el tipo de imagen~~				15
Elementos de interfaz para mejorar la usabilidad en la exportación de fórmulas				4
Configuración de la toolbar del editor				5
Uso de la cámara para la detección de fórmulas				30
Soporte para diferentes idiomas	+Traductor			6
Automatización del build y testeo de aplicaciones	· Desarrollador Software	· Ordenador sobremesa Mac Mini i5 3GHz 8GB RAM 256GB SSD · iPad Pro 2018 11" 64GB	MacOS Mojave 10.14, VS Code, XCode, Apache Ant, Jenkins	30
Documentación				**25**
Documentación de desarrollo de *MathType for iOS*	· Escritor técnico · Desarrollador Software	Ordenador portátil Dell Inspiron 7537 i7 2,66GHz 8GB RAM 500GB SSD	VS Code	25
Memoria y presentación				**63**
Memoria del proyecto	· Jefe de proyecto	Ordenador portátil Dell Inspiron 7537 i7 2,66GHz 8GB RAM 500GB SSD	TexStudio, Google Slides	55
Presentación del proyecto				8
TOTAL				**553**

Tabla 8.3: Primera parte del diagrama de Gantt.

Task Mode	Task Name	Duration	Start	Finish
	Asignatura GEP	**75 hrs**	**13/03/19**	**02/04/19**
	Clase de presentación inicial	2 hrs	13/03/19	13/03/19
	Entregable 1	25 hrs	13/03/19	20/03/19
	Entregable 2	12 hrs	20/03/19	22/03/19
	Entregable 3	12 hrs	22/03/19	27/03/19
	Entregable 4	20 hrs	27/03/19	02/04/19
	Ensayo de presentación final	2 hrs	02/04/19	02/04/19
	Clase de presentación final	2 hrs	02/04/19	02/04/19
	Análisis	**70 hrs?**	**18/01/19**	**06/02/19**
	Toma de requisitos	30 hrs?	18/01/19	25/01/19
	Diseño y especificación software	15 hrs?	28/01/19	30/01/19
	Diseño de la interfaz	25 hrs?	31/01/19	06/02/19
	Preparación del entorno de trabajo	**50 hrs?**	**07/02/19**	**20/02/19**
	Autoaprendizaje	20 hrs?	07/02/19	12/02/19
	Instalación y configuración del ordenador de desarrollo	15 hrs	13/02/19	15/02/19
	Instalación y configuración de los servidores de integración continua	15 hrs	18/02/19	20/02/19
	Pruebas de concepto y automatización	**70 hrs?**	**21/02/19**	**12/03/19**
	Autoaprendizaje	30 hrs?	21/02/19	28/02/19
	Pruebas de concepto	40 hrs?	01/03/19	12/03/19
	Desarrollo de MathType for iOS	**200 hrs?**	**13/03/19**	**07/05/19**

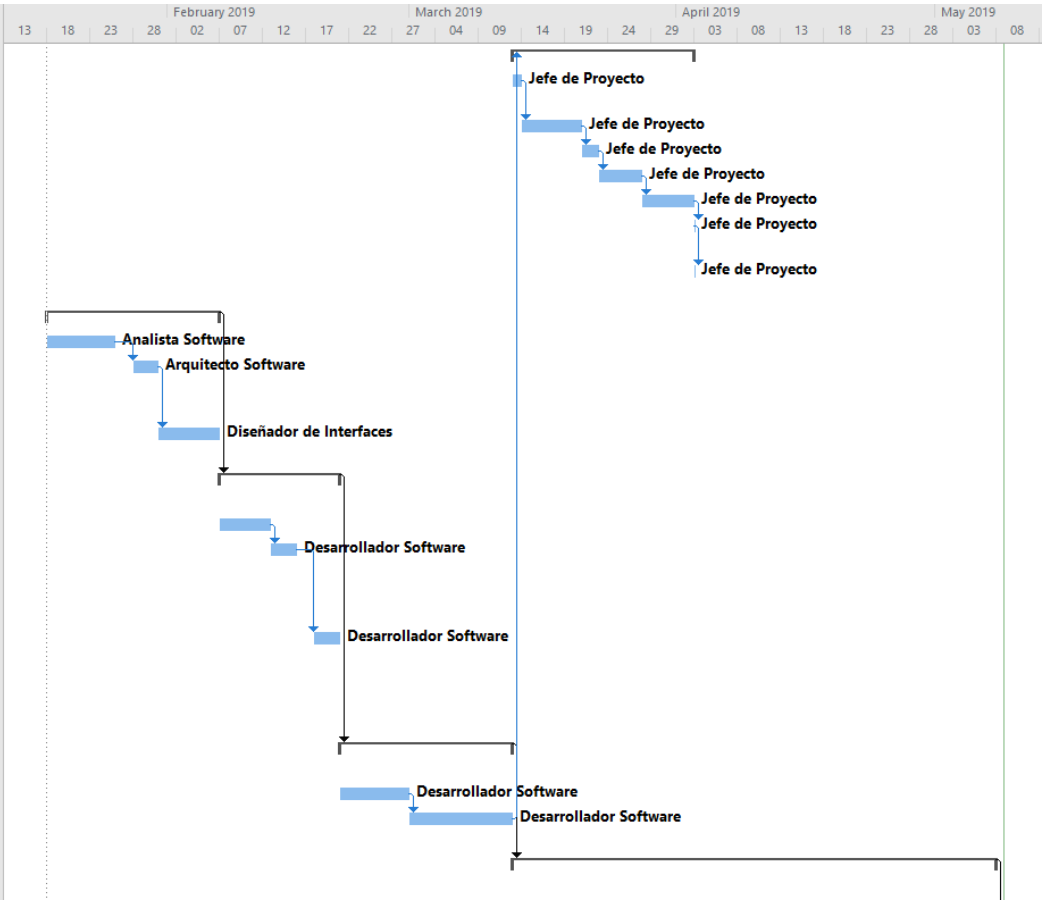
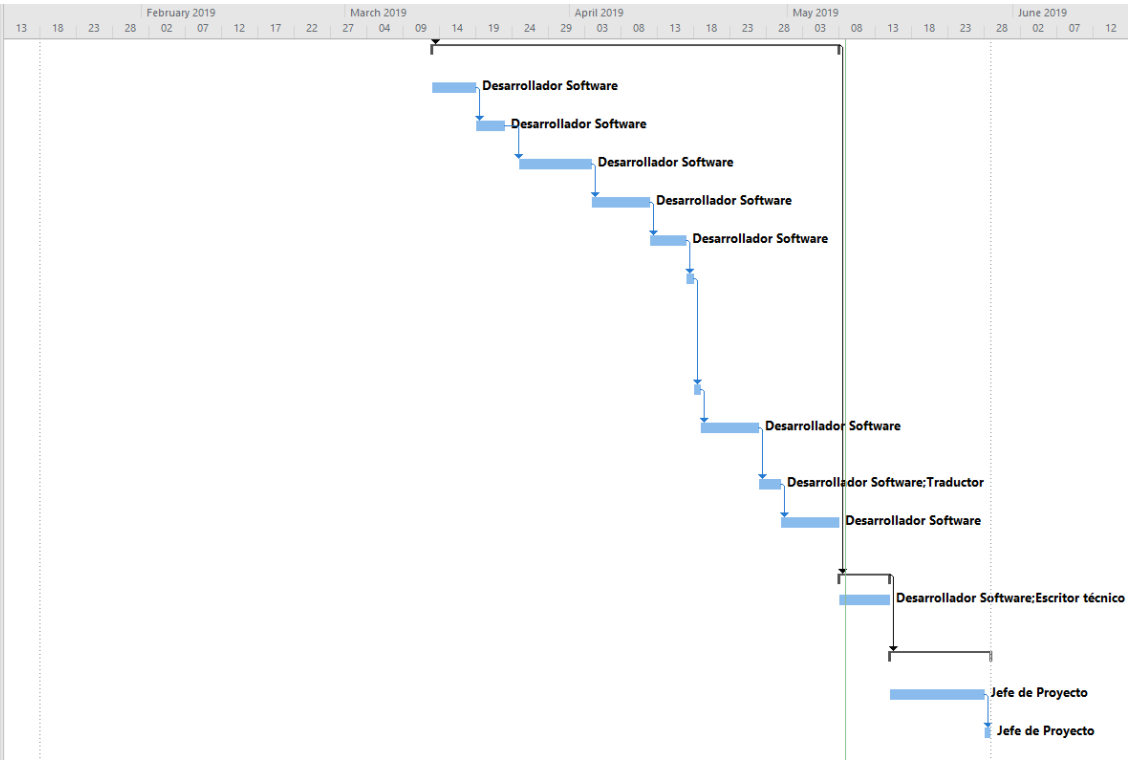


Tabla 8.4: Segunda parte del diagrama de Gantt.

Task Mode	Task Name	Duration	Start	Finish
	Desarrollo de MathType for iOS	**200 hrs?**	**13/03/19**	**07/05/19**
	Integrar MathType en iOS	20 hrs	13/03/19	18/03/19
	Import/Exportar imágenes	20 hrs?	19/03/19	22/03/19
	Reedición de fórmulas	40 hrs	25/03/19	03/04/19
	Mostrar fórmulas recientes	30 hrs	04/04/19	11/04/19
	Cambio en el tipo de imagen	15 hrs	12/04/19	16/04/19
	Elementos de interfaz para mejorar la usabilidad en la exportación de fórmulas	4 hrs?	17/04/19	17/04/19
	Configuración de la toolbar del editor	5 hrs?	17/04/19	18/04/19
	Uso de la cámara para la detección de fórmulas	30 hrs	18/04/19	26/04/19
	Soporte para diferentes idiomas	6 hrs	26/04/19	29/04/19
	Automatización del build y testeo de aplicaciones	30 hrs?	30/04/19	07/05/19
	Documentación	**25 hrs?**	**08/05/19**	**14/05/19**
	Documentación de desarrollo de MathType for iOS	25 hrs?	08/05/19	14/05/19
	Memoria y presentación	**50 hrs**	**15/05/19**	**28/05/19**
	Memoria del proyecto	42 hrs	15/05/19	27/05/19
	Presentación del proyecto	8 hrs	27/05/19	28/05/19



8.4.2. Impacto en los objetivos o desarrollo del proyecto

El impacto de los cambios comentados anteriormente han sido menores gracias a que no han sido problemas complejos o se ha conseguido encontrar una alternativa para minimizar el impacto.

Por una parte, el cambio de la documentación de usuario por la de developers se prevé que aumente el número de horas totales del proyecto.

Por otra parte, la redistribución de las tareas de automatización y *testing*, así como su uso en máquinas configuradas para ello ha provocado que el proyecto no pudiera aplicar buenas prácticas de desarrollo como lo son la integración continua y que los tests se hayan pasado manualmente después de cada tarea terminada. Sin embargo, se cree que ha ahorrado tiempo por no tener que mantener los supuestos tests automáticos que se tendrían debido a los cambios continuos sobre la interfaz.

Finalmente, el impacto de la limitación del sistema que ha provocado que no se pudieran cambiar de formato las imágenes ni añadir información extra ha provocado un ligero retraso debido a la replanificación de tareas para que el proyecto continuara siendo viable.

Capítulo 9

Gestión económica

Una vez realizada la planificación temporal existen datos suficientes como para desarrollar el presupuesto del proyecto. Éste justifica la viabilidad económica de cara a la empresa *Maths for More S.L.* .

9.1. Identificación y estimación de costes

La identificación de costes es el paso previo a la elaboración del presupuesto. Esta sección analiza desde el punto de vista económico los diferentes tipos de recursos del proyecto, definidos previamente durante la planificación.

9.1.1. Costes directos

Recursos humanos. Este proyecto está realizado íntegramente por el estudiante del TFG, asumiendo cada uno de los roles correspondientes durante la realización de cada tarea.

El salario bruto de cada rol ha sido extraído de la *Guía del mercado laboral 2019*[24] de la empresa *Hays*, una de las empresas de recursos humanos más grandes del mundo, y webs reconocidas como *PayScale*[40] e *Indeed*[25]. Los precios se han calculado en base a un trabajo anual de 2008h y un nivel de experiencia equivalente a "junior", es decir, entre 0 y 2 años de experiencia.

Una vez extraído el salario bruto de cada empleado se ha realizado una aproximación del costo que supone para la empresa tener a dicho trabajador, ya que existen más costes como el de la Seguridad Social o formación. La fórmula utilizada es la siguiente:

$$CosteEmpleadoEmpresa = SueldoBrutoEmpleado \cdot 1,35 \tag{9.1}$$

De esta investigación, se ha obtenido la siguiente tabla con los costes por tarea, fase y rol.

Tabla 9.1: Coste de RRHH por rol, fases y tareas.

Fases y tareas	Horas	Jefe de proyecto	Analista Software	Arquitecto Software	Diseñador de Interfaces	Desarrollador Software	Traductor	Escritor técnico	Coste
Coste en €/hora		33,62 €	24,20 €	33,62 €	20,17 €	20,17 €	12,57 €	21,88 €	
Asignatura GEP	**75**	2.521,2 €							2.521,2 €
Clase de presentación inicial	2	67,2 €							67,2 €
Entregable 1	25	840,4 €							840,4 €
Entregable 2	12	403,4 €							403,4 €
Entregable 3	12	403,4 €							403,4 €
Entregable 4	20	672,3 €							672,3 €
Ensayo de presentación final	2	67,2 €							67,2 €
Clase de presentación final	2	67,2 €							67,2 €
Análisis	**60**		726,1 €	504,2 €	302,5 €				1.532,9 €
Toma de requisitos	30		726,1 €						726,1 €
Diseño y especificación software	15			504,2 €					504,2 €
Diseño de la interfaz	15				302,5 €				302,5 €
Preparación del entorno de trabajo	**60**					1.210,2 €			1.210,2 €
Instalación y configuración del ordenador de desarrollo	30					605,1 €			605,1 €
Instalación y configuración de los servidores de integración continua	30					605,1 €			605,1 €
Pruebas de concepto y automatización	**85**					1.714,4 €			1.714,4 €
Autoaprendizaje	20					403,4 €			403,4 €
Pruebas de concepto	35					705,9 €			705,9 €
Automatización del build y testeo de aplicaciones	30					605,1 €			605,1 €
Desarrollo de MathType for iOS	**185**					3.428,8 €	188,5 €		3.617,3 €
Integrar MathType en iOS	20					403,4 €			403,4 €
Importar/Exportar imágenes	15					302,5 €			302,5 €
Reedición de fórmulas	40					806,8 €			806,8 €
Mostrar fórmulas recientes	30					605,1 €			605,1 €
Cambio en el tipo de imagen	20					403,4 €			403,4 €
Uso de la cámara para la detección de fórmulas	30					605,1 €			605,1 €
Soporte para diferentes idiomas	30 [1]					302,5 €	188,5 €		491,1 €
Documentación	**25**					100,8 €		437,5 €	538,4 €
Documentación de usuario de *MathType for iOS*	25 [2]					100,8 €		437,5 €	538,4 €
Memoria y presentación	**50**	1.680,8 €							1.680,8 €
Memoria del proyecto	42	1.411,9 €							1.411,9 €
Presentación del proyecto	8	268,9 €							268,9 €
TOTAL	**540**	4.201,9 €	726,1 €	504,2 €	302,5 €	6.454,2 €	188,5 €	437,5 €	**12.815,1 €**

Recursos hardware y software. Por otro lado, también existe un gasto en los recursos hardware y software desde el inicio del proyecto.

Las amortizaciones se han calculado en base a una vida útil que en ningún caso supera a los años máximos que propone la Agencia Tributaria[1]. Se ha extraído el precio amortizado por hora y se ha calculado el importe del gasto del proyecto teniendo en cuenta que éste utiliza dichos recursos durante 540h.

Los costes se pueden observar en la siguiente tabla. No han sido incluidos recursos gratuitos.

Tabla 9.2: Coste de recursos materiales y software.

Recurso	Tipo	Precio	Vida útil (años)	Coste con amortizaciones
Ordenador portátil Dell Inspiron 7537 i7 2,66GHz 8GB RAM 500GB SSD	Hardware	701,7 €	4	47,2 €
Ordenador sobremesa Mac Mini i5 3GHz 8GB RAM 256GB SSD	Hardware	1.032,2 €	4	69,4 €
iPad Pro 2018 11" 64GB	Hardware	726,4 €	4	48,8 €
Windows10	Software	119,8 €	2	16,1 €
Total		2.580,2 €		**181,5 €**

9.1.2. Costes indirectos

Durante el desarrollo existen costes indirectos como consecuencia de las actividades que se realizan. Dado que el TFG se realiza en una empresa, es el coste de recursos generales (agua, oficina, Internet... etc).

Las amortizaciones se han calculado de la misma forma que en el apartado anterior. Sin embargo, en la columna "Vida útil" hay algunas filas sin datos debido a que son consumibles y han sido calculados de diferentes formas.

Los servicios, la conexión a Internet y el local (200m2 en el centro de Barcelona) han sido calculados a partir de las facturas mensuales de gasto de la empresa y estimando que el proyecto cuesta 4 meses de alquiler, ya que no existe la posibilidad de pagar en unidades menores al mes. El resultado se divide en partes iguales entre los 32 trabajadores de la oficina. En cambio, los snacks han sido calculados en base al gasto individual del estudiante de 6€ cada 40h de trabajo.

Los costes indirectos por lo tanto son los siguientes:

Tabla 9.3: Costes indirectos.

Recurso	Precio	Vida útil	Coste con amortizaciones
Servicios (agua, luz, impuestos de basuras...)	105,38 €	-	105,38 €
Conexión a Internet	22,38 €	-	22,38 €
Silla de oficina	119,99 €	10	3,23 €
Mesa de oficina	179,95 €	10	4,84 €
Snacks	81,00 €	-	80,00 €
Local	325,00 €	-	325,00 €
Total			**540,82 €**

9.1.3. Contingencias

Como todo proyecto, existe un riesgo de necesitar un presupuesto mayor al elaborado debido a los sucesos inesperados. Por ello existe una partida destinada a contener esas posibles desviaciones. Para este proyecto se utilizará un fondo de contingencias del 10 %, una cifra encontrada de manera empírica y que en asignaturas como GPS se nos ha impulsado a utilizar por su buen resultado.

9.1.4. Coste de incidentes

Finalmente se presenta el coste de los incidentes que pueden surgir en el proyecto.

Para ello, se tiene en cuenta el caso pesimista de los problemas presentados en el plan de acción de la planificación, donde se habla del incremento de horas de trabajo. Este aumento de horas probablemente se daría a la mitad del proyecto, durante la fase de *Desarrollo de MathType for iOS*. Debido a eso se ha calculado un coste asumiendo que el aumento de horas empieza en la tarea "Cambio en el tipo de imagen" hasta la finalización del proyecto.

Por otra parte, se ha tenido en cuenta la pérdida del Mac Mini, la herramienta de desarrollo principal. Cuyo coste, gracias a que la entrega podría ser el mismo día y que el estudiante tiene horario flexible, se limitaría a un coste material.

Tabla 9.4: Costes de incidentes.

Causa	Solución	Probabilidad	Coste derivado
Falta de tiempo para terminar el proyecto	Aumentar el número de horas de los roles	30%	446,2 €
Pérdida del ordenador Mac Mini	Comprar un nuevo ordenador	5%	51,6 €
TOTAL			**497,9 €**

9.2. Coste total

De las secciones anteriores, se puede extraer la siguiente tabla con el presupuesto del proyecto.

Tabla 9.5: Coste total.

Concepto	Coste
Costes directos	12.996,6 €
Costes indirectos	540,8 €
Contingencias (10%)	1.353,7 €
Costes de Incidentes	497,9 €
TOTAL SIN IVA	**15.389,0 €**
TOTAL CON IVA	**18.620,7 €**

9.3. Control de gestión

Para el control económico del proyecto se hace uso de indicadores de control. Concretamente de cumplimiento y eficiencia.

El primer indicador es el de finalización de tareas. Dicho indicador tiene que ver con la conclusión de las tareas del proyecto llevando una cuenta de las ya completadas. De esta manera, es posible medir el nivel de realización del TFG.

El segundo indicador permite medir el nivel de ejecución de las tareas según los recursos utilizados. Al finalizar cada tarea se anotan las horas y recursos invertidos reales y, dichos datos, se comparan con las estimaciones realizadas haciendo uso de las siguientes fórmulas presentadas durante la asignatura de GEP[11].

$$\begin{aligned} Desv\acute{i}oEnCoste &= (CE - CR) \cdot CHR \\ Desv\acute{i}oEnConsumo &= (CHE - CHR) \cdot CE \end{aligned} \tag{9.2}$$

Donde: CE=Coste Estimado; CR=Coste Real; CHR=Consumo de Horas Real; CHE=Consumo de Horas Estimado.

Todo ello permite saber el estado del proyecto respecto al consumo de recursos y también conocer las posibles desviaciones, para así calcular si el proyecto puede soportarlas haciendo uso de la partida de contingencias.

9.4. Desviaciones durante el proyecto

Como se ha comentado y justificado en el capítulo de la planificación, durante el proyecto han aparecido un conjunto de desviaciones que ha provocado una replanificación de las tareas.

Esto, ha afectado de forma directa tanto en el tiempo como en el coste del proyecto. Han aumentado en los costes de mano de obra 532,6€ debido al aumento de las horas que se pensaban destinar al proyecto de manera inicial.

De esta manera, ha aumentado la cantidad de horas a pagar al jefe de proyecto y también al desarrollador software.

Para terminar, las siguientes tablas muestran en detalle el aumento del coste total. Como dato a comentar, los costes materiales y los costes indirectos también han aumentado dado que para el cálculo del primero se ha tenido en cuenta las amortizaciones y para el segundo el precio de los servicios aumenta en función de las horas de uso.

Tabla 9.6: Costes de mano de obra recalculados.

Fases y tareas	Horas	Jefe de proyecto	Analista Software	Arquitecto Software	Diseñador de Interfaces	Desarrollador Software	Traductor	Escritor técnico	Coste
Coste en €/hora		33,62 €	24,20 €	33,62 €	20,17 €	20,17 €	12,57 €	21,88 €	
Asignatura GEP	**75**	2.521,2 €							2.521,2 €
Clase de presentación inicial	2	67,2 €							67,2 €
Entregable 1	25	840,4 €							840,4 €
Entregable 2	12	403,4 €							403,4 €
Entregable 3	12	403,4 €							403,4 €
Entregable 4	20	672,3 €							672,3 €
Ensayo de presentación final	2	67,2 €							67,2 €
Clase de presentación final	2	67,2 €							67,2 €
Análisis	**70**		726,1 €	504,2 €	504,2 €				1.734,6 €
Toma de requisitos	30		726,1 €						726,1 €
Diseño y especificación software	15			504,2 €					504,2 €
Diseño de la interfaz	25				504,2 €				504,2 €
Preparación del entorno de trabajo	**50**					1.008,5 €			1.008,5 €
Autoaprendizaje	20					403,4 €			
Instalación y configuración del ordenador de desarrollo	15					302,5 €			302,5 €
Instalación y configuración de los servidores de integración continua	15					302,5 €			302,5 €
Pruebas de concepto y automatización	**70**					1.411,9 €			1.411,9 €
Autoaprendizaje	30					605,1 €			605,1 €
Pruebas de concepto	40					806,8 €			806,8 €
Desarrollo de MathType for iOS	**200**					3.973,4 €	37,7 €		4.011,1 €
Integrar MathType en iOS	20					403,4 €			403,4 €
Importar/Exportar imágenes	20					403,4 €			403,4 €
Reedición de fórmulas	40					806,8 €			806,8 €
Mostrar fórmulas recientes	30					605,1 €			605,1 €
~~Cambio en el tipo de imagen~~	15					302,5 €			302,5 €
Elementos de interfaz para mejorar la usabilidad en la exportación de fórmulas	4					80,7 €			
Configuración de la toolbar del editor	5					100,8 €			
Uso de la cámara para la detección de fórmulas	30					605,1 €			605,1 €
Soporte para diferentes idiomas	6 [1]					60,5 €	37,7 €		98,2 €
Automatización del build y testeo de aplicaciones	30					605,1 €			605,1 €
Documentación	**25**					100,8 €		437,5 €	538,4 €
Documentación de desarrollo de *MathType for iOS*	25 [2]					100,8 €		437,5 €	538,4 €
Memoria y presentación	**63**	2.117,8 €							2.117,8 €
Memoria del proyecto	55	1.848,9 €							1.848,9 €
Presentación del proyecto	8	268,9 €							268,9 €
TOTAL	**553**	4.638,9 €	726,1 €	504,2 €	504,2 €	6.494,5 €	37,7 €	437,5 €	**13.343,3 €**

Tabla 9.7: Costes recalculados de los recursos hardware y software.

Recurso	Tipo	Precio	Vida útil (años)	Coste con amortizaciones
Ordenador portátil Dell Inspiron 7537 i7 2,66GHz 8GB RAM 500GB SSD	Hardware	701,7 €	4	48,3 €
Ordenador sobremesa Mac Mini i5 3GHz 8GB RAM 256GB SSD	Hardware	1.032,2 €	4	71,1 €
iPad Pro 2018 11" 64GB	Hardware	726,4 €	4	50,0 €
Windows10	Software	119,8 €	2	16,5 €
Total		2.580,2 €		**185,9 €**

Tabla 9.8: Costes indirectos recalculados.

Recurso	Precio	Vida útil	Coste con amortizaciones
Servicios (agua, luz, impuestos de basuras...)	105,38 €	-	105,38 €
Conexión a Internet	22,38 €	-	22,38 €
Silla de oficina	119,99 €	10	3,30 €
Mesa de oficina	179,95 €	10	4,96 €
Snacks	82,95 €	-	80,00 €
Local	325,00 €	-	325,00 €

Tabla 9.9: Costes totales recalculados.

Concepto	Coste
Costes directos	13.529,2 €
Costes indirectos	541,0 €
Contingencias (10%)	1.407,0 €
Costes de Incidentes	443,0 €
TOTAL SIN IVA	**15.920,1 €**
TOTAL CON IVA	**19.263,4 €**

Capítulo 10

Sostenibilidad y compromiso social

Para identificar la sostenibilidad del proyecto, se evalúa el impacto de tres aspectos del mismo: económico, ambiental y social. A continuación se explica cada aspecto en el contexto del TFG.

10.1. Dimensión económica

Se ha realizado un análisis de costes preciso para el proyecto teniendo en cuenta todo tipo de costes: directos, indirectos, contingencias y costes debido a incidentes. Además, dicho análisis se ha realizado respecto a las fases y tareas, por lo que su seguimiento y control es más preciso al ser un intervalo de tiempo y conjunto de recursos más acotado. Por esto mismo, se considera que la estimación de costes ha sido razonable.

Por otra parte, este proyecto se ve respaldado con cifras internas de la empresa *Maths for More S.L.* . Dichas cifras prevén que más de 500 licencias del producto serán vendidas a usuarios activos de *MathType for iOS*. Por lo tanto, con un coste por licencia de aproximadamente 50€ se obtiene que la cifra de retorno es de 25.000€ antes de impuestos, superando así el presupuesto del proyecto. También hay que tener en cuenta que la suscripción será pagada más de una vez ya que es anual, por lo que el ingreso aún será mayor.

Sin embargo, dicha previsión se espera de aquí a 1-2 años, por lo que *Maths for More S.L.* ha decidido destinar los recursos de una sola persona para el desarrollo.

En cuanto a otras soluciones alternativas, existen problemas debido a una falta de recursos. Son pequeñas empresas, que se dedican solamente a la

edición de fórmulas matemáticas; o grandes empresas, donde la edición de fórmulas matemáticas es solamente una funcionalidad más de su programa y por lo tanto destinan una cantidad de recursos muy limitada. En ambos casos la estimación de recursos se realiza en función de los recursos humanos necesarios y orden de importancia.

A diferencia de estas, *Maths for More S.L.* y su producto *MathType* se dedica específicamente a la edición de fórmulas matemáticas y tiene los recursos suficientes para desarrollar varios proyectos al mismo tiempo, pudiendo reducir el coste material reutilizando gran parte de los recursos.

También hace falta recordar el bajo riesgo que existe en la dimensión económica debido a que, como ya se ve en la planificación, los costes por incidentes y sus respectivas probabilidades son muy bajos.

Y ya para terminar, se quiere calcular el coste de vida útil del producto asumiendo que para darle mantenimiento se aplica un 10 % de los costes indirectos que se tienen durante todo un año. La justificación de por qué un 10 % es que la automatización de *MathType for iOS* hace que el tiempo de los desarrolladores para su mantenimiento se reduzca considerablemente. Además, también durante el tiempo que se le da mantenimiento se está realizando un esfuerzo en la producción de otros productos de *Maths for More S.L.*. Por lo tanto queda que de los 541,01€ de los costes indirectos cada 4 meses, se gastarían 162,303€ al año.

10.2. Dimensión ambiental

Durante el proyecto, se ha tenido en cuenta la reutilización de recursos. Un ejemplo ha sido el ordenador portátil o el iPad Pro, utilizados en diferentes proyectos. Sin embargo, no se ha podido hacer lo mismo para el caso del Mac Mini y la electricidad consumida.

Se prevé que la electricidad consumida por estos equipos[5][10][39][38] sea de 37,5kW·h de media. Por otro lado, no se tiene en cuenta los gastos generales.

Además, la solución proporcionada mejora entre otras cosas la rapidez respecto a otras soluciones similares. Esto provoca una reducción del consumo, ya que los usuarios utilizan menos tiempo su dispositivo al haber terminado antes el trabajo. Una reducción del tiempo de uso de los usuarios de un 15 % supone una disminución de energía consumida en el iPad Pro de 11,5kW·h a 9,775kW·h (Sin tener en cuenta pérdidas).

Las soluciones alternativas no tienen en cuenta la eficiencia energética y la interfaz que presentan, con colores vivos y más lentas de utilizar, gastan más

energía.

Tabla 10.1: Energía eléctrica consumida.

Recurso	Consumo (Watts)	Horas de uso	Consumo (kW·h)
Ordenador Dell	70,00	210,00	14,70
Ordenador Mac Mini	69,00	330,00	22,77
iPad	11,50	6,00	0,07
TOTAL			**37,54**

Además, dichos costes ambientales tienen un riesgo de aumentar un 15 % debido a que es el porcentaje máximo de desviación del proyecto calculado durante la planificación.

Y ya para acabar, se ha utilizado en el cálculo de la huella ambiental durante la vida útil de la misma forma que en la dimensión económica, el 10 % de gasto realizado durante la realización del proyecto en un año. Por ello, si se consumen 37,54kW·h cada 4 meses, el resultado es que durante la vida útil se gastaría 11,262kW·h por año.

10.3. Dimensión social

No existe ningún editor específico de fórmulas matemáticas en iOS, pero según estadísticas internas existen usuarios de *MathType* que les gustaría poder trabajar en dicha plataforma. Esta solución aporta en el ámbito social una mejora en el trabajo de las personas que necesiten insertar o editar fórmulas matemáticas.

Las alternativas ofrecen una mala experiencia de usuario mientras que ésta tiene una mejor interfaz, provocando que el proceso de insertar o editar una fórmula sea menos tedioso y más rápido. Por lo tanto los usuarios serán más eficientes y disminuirán el tiempo dedicado a un trabajo poco gratificante.

Además, este proyecto aporta experiencia al estudiante. Tiene la posibilidad de aplicar sus conocimientos en un entorno nuevo para él y gracias a que es un proyecto *open source*, puede mostrarlo como ejemplo de su buen trabajo a empresas que piensen contratarle.

Finalmente, los riesgos que pueden suponer un efecto negativo no existen. Únicamente se podría dar el caso en el que la competencia mejorara lo ya creado por este proyecto y por lo tanto que *MathType for iOS* no fuera utilizado.

10.4. Matriz de sostenibilidad

Finalmente, se obtiene la siguiente matriz de sostenibilidad. Ésta muestra los valores en las unidades correspondientes de los apartados PPP y vida útil. Por otra parte, también se le asigna una puntuación de 0 a 10 en el caso de la dimensión social. La justificación de los valores se explican con detalle en los apartados anteriores.

Tabla 10.2: Matriz de sostenibilidad.

	PPP	**Vida útil**	**Riesgos**
Económica	19.263,4€	162,303€/año	30 % costo mano de obra 5 % rotura de material
Ambiental	37,54kW·h	11,262/año	15 %
Social	8	8	Pocos

Capítulo 11

Conclusiones

Como capítulo final, se encuentra la conclusión del proyecto. Dicha conclusión se observa desde diferentes puntos de vista que se irán explicando en las siguientes secciones y que ayudan a tener una visión global de cómo ha ido el proyecto, qué ha fallado, qué se ha aprendido y de cara a un futuro qué se podría hacer.

11.1. Valoración personal

En general, siento que la aplicación ha quedado y no ha quedado bien.

Desde el punto de vista de la empresa y la validación, todos los requisitos han quedado cumplidos y el objetivo del proyecto ha sido alcanzado. Pero por otra parte a nivel personal el resultado visual no ha terminado de convencerme. Simplemente hace falta ver otras aplicaciones como *Facebook*, *Uber* o *Twitter* y es evidente que tienen un mejor diseño debido a que poseen recursos suficientes para tener a diseñadores e ingenieros de UX.

Sin embargo, ha sido un proyecto que a nivel de arquitectura propuesta ha sido muy divertido. He aplicado muchos de los conceptos aprendidos durante la carrera y también de mi experiencia como programador para cumplir los requisitos previamente analizados.

Además, la empresa *Maths for More S.L.* me ha dado la suficiente libertad como para dirigir el proyecto. Eso ha tenido sus cosas buenas y malas, me ha permitido ser creativo con la solución pero por otra parte en cosas como el diseño me ha supuesto una dificultad el no tener ya un estudio previo por su parte que me dijera qué querían.

A su vez, me ha permitido aprender nuevas tecnologías muy demandadas actualmente como las relacionadas con la *integración continua* y las de auto-

matización del *building* y *testing*, enseñándome a hacer el código más mantenible.

Por una parte me ayuda en la detección de errores y me libra de tener que gastar mis recursos en trabajos repetitivos, pero por otra parte también me ayuda a que de cara a un futuro el sistema pueda ser mantenido por otra persona sin tener que crear más documentación extra, ya que la implementación de la automatización se encuentra escrita en un lenguaje comprensible el cual podría ser leído por la nueva persona que me sustituyera.

11.2. Conocimientos aplicados

Durante todo el transcurso del *TFG* se han aplicado conocimientos de muchas de las asignaturas cursadas durante la carrera. Algunas de éstas son:

- **Enginyeria de Requisits (ER).** Esta asignatura ha ayudado al proyecto a hacer la toma y análisis de los requisitos necesarios para cumplir los objetivos propuestos por *Maths for More S.L.*.
- **Gestió de Projectes Software (GPS).** Se han aplicado muchos de los conocimientos de gestión de proyectos y también las reglas de las metodologías ágiles que fueron propuestas a lo largo de la asignatura.
- **Arquitectura del Software (AS).** El conocimiento de esta asignatura ha sido el más utilizado. Gracias a ella han podido usarse en este proyecto patrones de diseño y también a tener en cuenta el nivel de acoplamiento entre clases para cumplir objetivos principales como la facilidad de cambio de las vistas de la aplicación.
- **Introducció a l'Enginyeria del Software (IES).** Esta asignatura supone el paso previo a la aplicación de los conocimientos de la anterior asignatura. Gracias a ella se ha logrado documentar el proyecto desde un punto de vista estático con los diagramas conceptuales y lógicos; y también desde el punto de vista dinámico con los diagramas de secuencia.

11.3. Competencias técnicas

Según la normativa académica, cada proyecto de *TFG* tiene asociado una serie de competencias técnicas que serán evaluadas en la defensa del mismo.

Este proyecto tiene las siguientes competencias:

- **CES1.1.** Desarrollar, mantener y evaluar sistemas software complejos y/o críticos. [Un poco]
- **CES1.2.** Dar una solución a problemas de integración en función de las estrategias, estándares y de las tecnologías disponibles. [En profundidad]
- **CES1.3.** Identificar, evaluar y gestionar los riesgos potenciales asociados a la construcción del software que se puedan presentar. [Bastante]
- **CES1.7.** Controlar la calidad y diseñar pruebas en la producción de software. [Bastante]
- **CES2.1.** Definir y gestionar los requisitos de un sistema software. [Bastante]
- **CES2.2.** Diseñar soluciones apropiadas en uno o más dominios de aplicación, utilizando métodos de ingeniería del software que integren aspectos éticos, sociales, legales y económicos. [Un poco]

En cuanto a la competencia ***CES1.1***, se ve justificada que tenga poco peso dentro del proyecto debido a que a pesar de ser un sistema que tiene que comunicarse con servicios externos, en este caso con algunos de los servicios de reconocimiento de *MathType*, no es un sistema en el que el no funcionamiento de manera temporal suponga una consecuencia crítica.

Por lo que tiene que ver a la competencia ***CES1.2***, sí que tiene un peso muy grande dentro del proyecto. Esto se debe a que se trata de una integración en una tercera plataforma como es iOS. Esto supone que se tengan que comunicar entre sí diferentes tecnologías, teniendo en cuenta que cada una presenta sus propias limitaciones. Además, también existen estándares de uso por parte de todos los proyectos internos de *Maths for More S.L.* que hay que cumplir, como en el caso de la automatización de la compilación y *testing*.

A su vez, el uso de esta gran cantidad de tecnologías justifica la competencia ***CES1.3*** debido a que se ha tenido que realizar un análisis a priori para intentar detectar todas las posibles limitaciones de cada tecnología.

La competencia ***CES1.7*** va íntimamente ligada a la idea de producción de un código de calidad y el uso de buenas prácticas de desarrollo. Durante el proyecto se ha realizado un sistema de *testing* para *MathType for iOS* y su posterior automatización, todo ello para asegurar la calidad del código. Además, para cada tecnología se ha realizado una fase de autoaprendizaje

con la intención de aprender a cómo usarla y también a conocer cuáles son las buenas prácticas, que posteriormente se han seguido durante este desarrollo.

Siguiendo con la competencia ***CES2.1***, se ve justificada en el análisis de requisitos realizado en el capítulo correspondiente.

Para terminar, la competencia ***CES2.2*** se ve justificada gracias al análisis económico, ambiental y social realizado durante la planificación del proyecto y también en la explicación de las desviaciones que ha sufrido el trabajo durante su realización.

11.4. Trabajo futuro

Como trabajo a mejorar de cara al futuro, existen diferentes aspectos de la aplicación que no han terminado de encajar a lo largo del desarrollo, ya sea por limitaciones, el tiempo o simplemente por la inexperiencia del estudiante.

Primero de todo la interfaz gráfica. Durante las entrevistas han habido diversidad de opiniones sobre si era bonita y sencilla de utilizar debido a que para algunos de los entrevistados existen grandes espacios vacíos en el menú lateral. Por lo tanto esto plantea la idea de un cambio de interfaz, cosa que sería sencillo de realizar dado que a arquitectura diseñada lo permite; y también la idea de realizar un mayor número de entrevistas, ya que se han realizado únicamente cinco y aumentar este número permitiría obtener un *feedback* más fiable.

Por otra parte, hay que hablar también de las funcionalidades que han quedado descartadas por limitaciones del sistema. De ellas la más destacable es la inserción de información necesaria en las imágenes generadas para así poder importarlas y obtener las fórmulas en el estándar *MathML* y no un reconocimiento que puede no ser correcto. Para ello, y como se comenta durante el trabajo, podría ser investigar la manera de implementar un algoritmo de esteganografía que resista transformaciones como la rotación o el escalado.

Por último comentar que el proyecto está utilizando servicios aún en desarrollo de *MathType*, por lo que es necesario una vez terminado dicho desarrollo cambiar a los servicios de producción. A su vez, también habría que eliminar el uso del proxy que se utiliza para saltar la seguridad de que el editor *MathType* solamente puede ser utilizado en páginas web en línea.

Apéndice A

Capturas de diseño de MathType for iOS

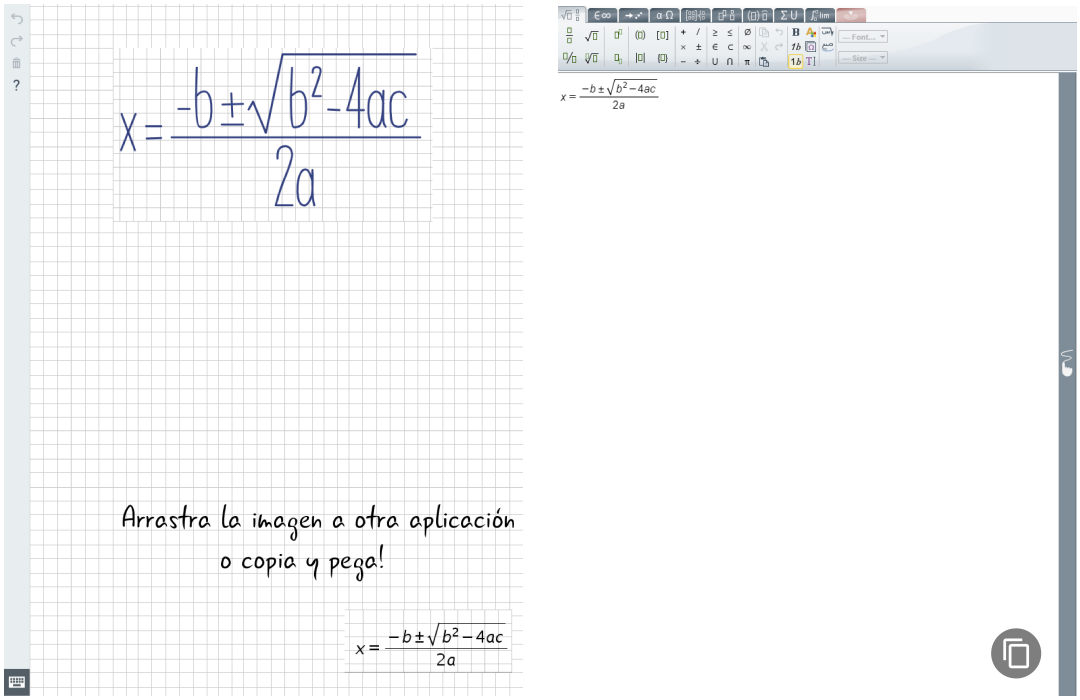


Figura A.1: Pantalla en modo handwriting y en modo toolbar de MathType for iOS.

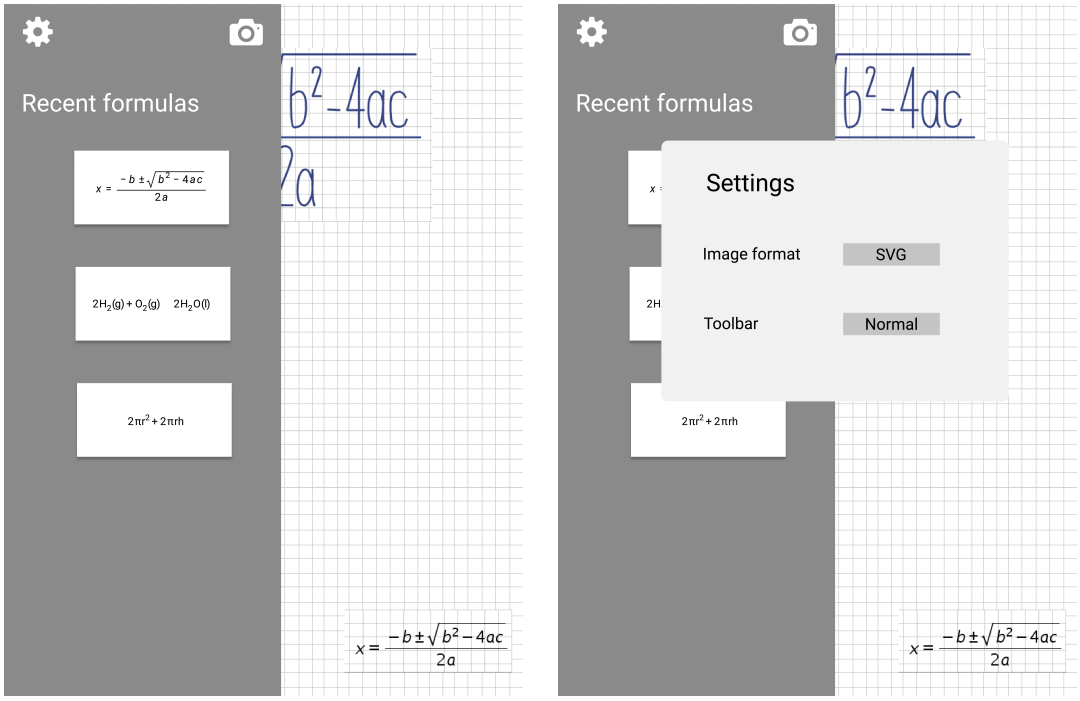


Figura A.2: Menú de MathType for iOS con las fórmulas recientes, el acceso a ajustes y cámara.

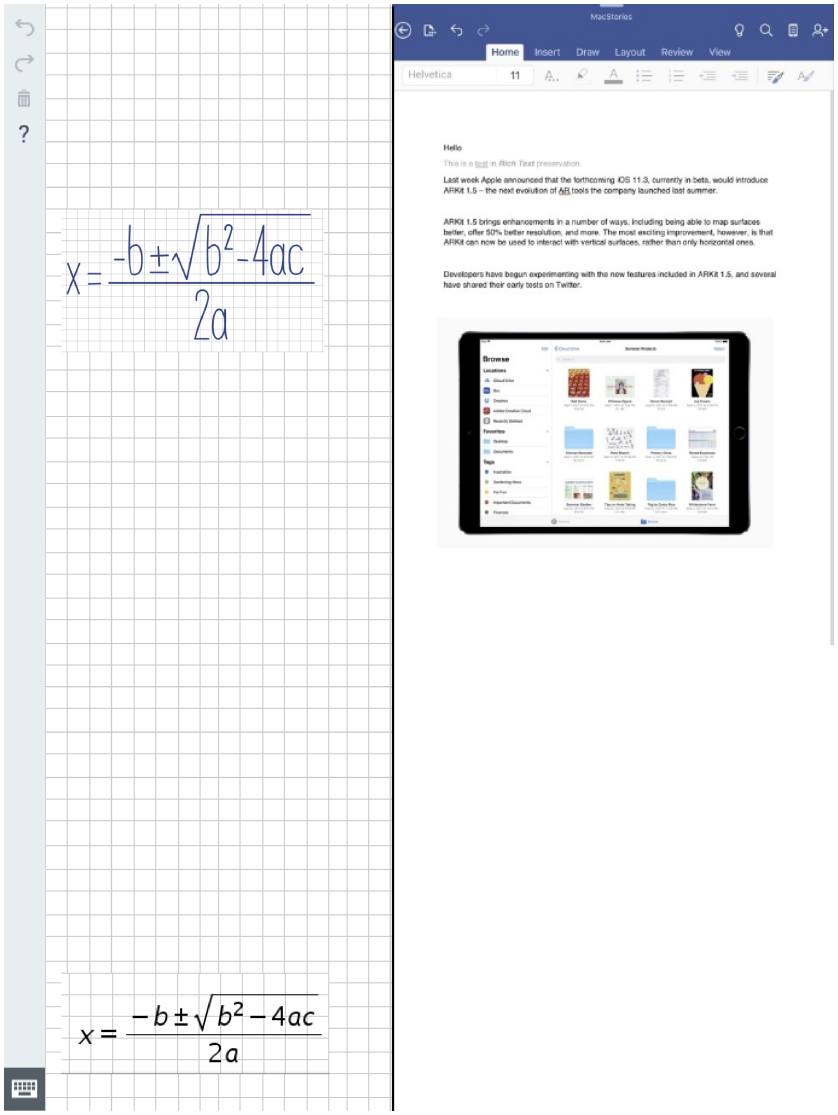


Figura A.3: MathType for iOS en modo Split View.

Apéndice B

Capturas de MathType for iOS

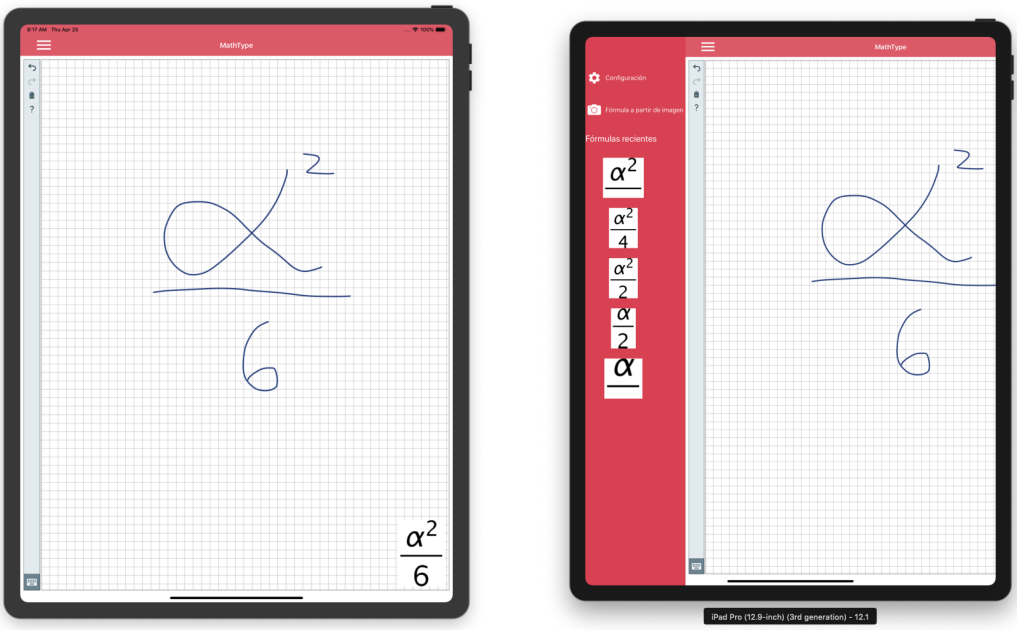


Figura B.1: MathType for iOS en modo handwriting y a la derecha la misma pantalla con el menú desplegado.

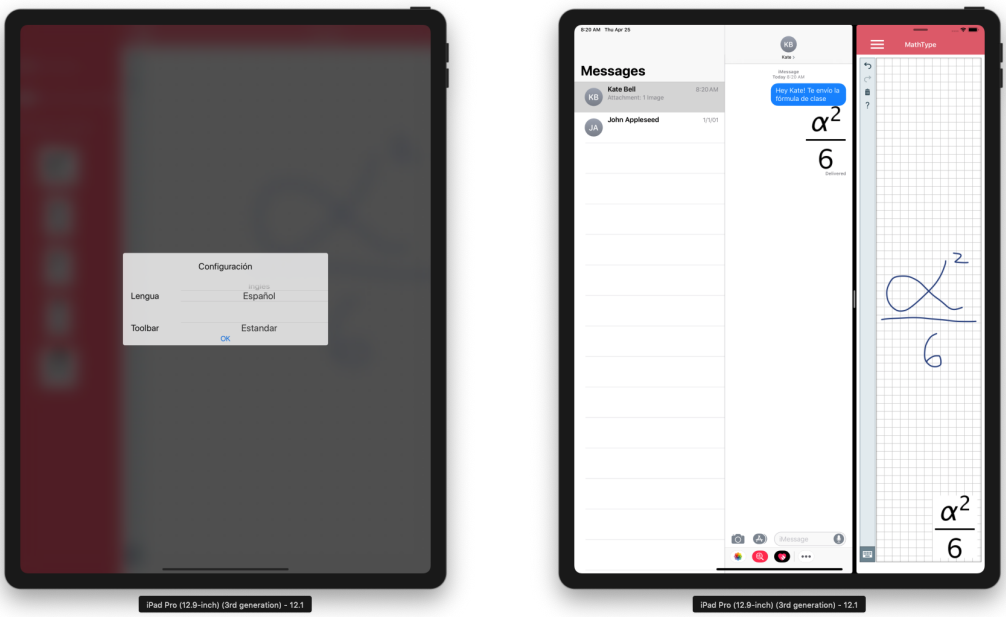


Figura B.2: Menú de configuración y a la derecha usando el modo *Split View*.

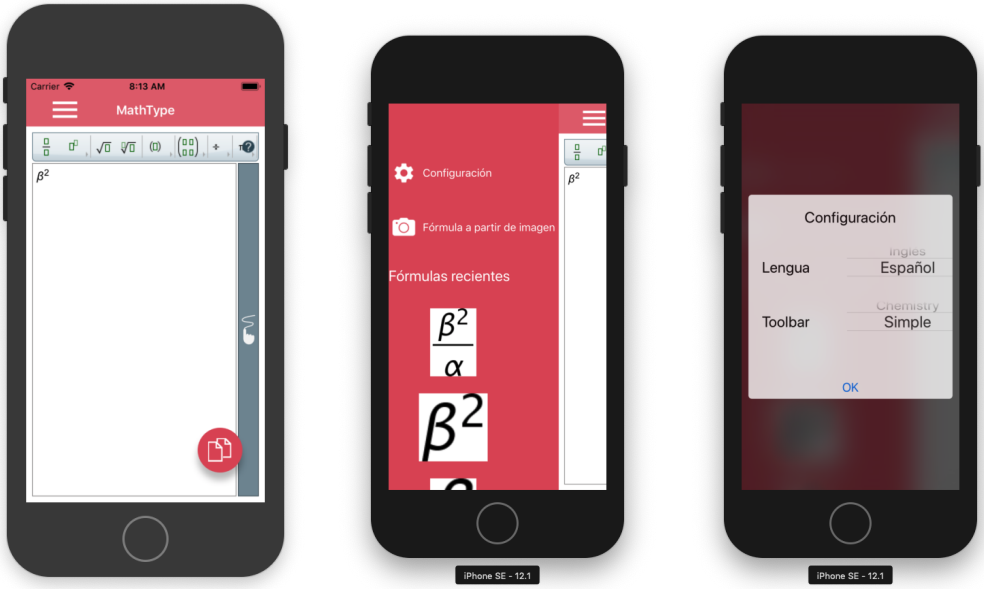


Figura B.3: *MathType for iOS en iPhone SE*.

Apéndice C

Diagrama Lógico de MathType for iOS

Controlador

Vista

Modelo

<<Interface>>
MenuControllerDelegate

+ onOptionSelected(menuOption: MenuOption)
+ onRecentFormulaSelected(formula: Formula)

<<Interface>>
ConfigurationControllerDelegate

+ onAcceptConfiguration()
+ onSelectedLanguage()
+ onSelectedToolbar(toolbarOption: ToolbarOption)
+ getEditorToolbar(): ToolbarOption

<<Interface>>
ControllerProtocol

+ getViewController(): ViewBase

<<Interface>>
BrowserControllerDelegate

+ onNewRecentFormula(formula: Formula)
+ onGetFormula(formula: Formula)
+ onEditorModeChange(editorMode: String)
+ onTouchStart()
+ onError()

<<Interface>>
ImagePickerControllerDelegate

+ onImage(image: UIImage)
+ needRemove()

MenuController

+ addRecentFormula(formula: Formula)
+ onSelectedLanguage()

ConfigurationController

BrowserController

+ setFormula(formula: Formula)
+ onSelectedToolbar(toolbarOption: ToolbarOption)
+ getEditorToolbar(): ToolbarOption
+ recognize(image: UIImage)

ImagePickerController

HomeController

<<Interface>>
MenuViewDelegate

+ onOptionSelected(menuOption: MenuOption)
+ onRecentFormulaSelected(formula: Formula)

<<Interface>>
ConfigurationViewDelegate

+ onAcceptConfiguration()
+ onSelectedLanguage()
+ onSelectedToolbar(toolbarOption: ToolbarOption)
+ getEditorToolbar(): ToolbarOption

<<Interface>>
HomeViewDelegate

+ getBrowserView(): ViewBase
+ getMenuView(): ViewBase
+ getConfigurationView(): ViewBase
+ getImagePickerView(): ViewBase

<<Interface>>
BrowserViewDelegate

+ onNewRecentFormula(formula: Formula)
+ onGetFormula(formula: Formula)
+ onEditorModeChange(editorMode: String)
+ onBeginToLoad()
+ onFinalizeLoad()
+ onTouchStart()
+ onError()

<<Interface>>
ImagePickerViewDelegate

+ onImage(image: UIImage)
+ needRemove()

MenuOptionCell

+ iconImageView: UIImageV
+ descriptionLabel: UILabel

MenuRecentFormulaCell

+ iconImageView: UIImageView

UITableViewCell

UITableView

UILabel

MenuView

+ onSelectedLanguage()
+ addRecentFormula(formula: Formula)

HomeView

+ showLoading()
+ hideLoading()
+ onRecognizeFormulaInImage()
+ didImagePicked()
+ onErrorBrowserView()
+ showFloatButton()
+ hideFloatButton()
+ showConfigurationMenu()
+ hideConfigurationMenu()
+ toogleMenu()
+ closeMenu()

CopyButton

+ pulsate()

UIButton

«**static**»
ToastView

+ short(view: UIView, message: String)
+ long(view: UIView, message: String)

«**extension**»
UIImage

+ resizeWith(percetage: CGFloat): UIImage
+ resizeWith(width: CGFloat): UIimage

Formula

+ src: String
+ mml: String

0..1

«**enumeration**»
ToolbarOption

standard
chemistry
simple

Editor

+ handwritingMode: Bool

+ getFormula(): Formula
+ setFormula(formula: Formula)
+ getEditorMode(): Enum_Toolbar
+ getToolbar(): Toolbar
+ setToolbar(toolbar: Toolbar)

«interface»
UITableViewDragDelegate

RecentFormulasTableViewDelegate

+ delegate: MenuViewDelegate
+ tableview: UITableView

+ numberOfRowsInSection(): Int
+ cellForRowAt(): UITableViewCell
+ didSelectRowAt(index: Int)
+ update()
+ beginningDragSession(): [UIDragItem]

ViewBase

+ render()
+ refreshSize()
+ animateIn()
+ animateOut()

«interface»
NSObject

«interface»
UITableViewDelegate

«interface»
UITableViewDataSource

OptionsTableViewDelegate

+ delegate: MenuViewDelegate

+ numberOfRowsInSection(): Int
+ cellForRowAt(): UITableViewCell
+ didSelectRowAt(index: Int)

«**static**»
Device

+ isPad(): Bool

BrowserView

+ setFormula(formula: Formula)
+ getEditorToolbar(): ToolbarOption

ImagePickerView

«interface»
ImagePickerViewDelegate

«interface»
UIImagePickerControllerDelegate

«interface»
UINavigationControllerDelegate

ConfigurationView

«**enumeration**»
MenuOption

+ description: String
+ image: UIImage

«**enumeration**»
ColorPicker

+ uiColor.blue
+ uiColor.cyan
+ uiColor.red
+ uiColor.orange
+ uiColor.yellow
+ uiColor.green
+ uiColor.purple
+ uiColor.pink
+ uiColor.black
+ uiColor.white
+ uiColor.darkGray
+ uiColor.lightGray
+ uiColor.lightRedWiris
+ uiColor.redWiris
+ uiColor.redOrangeWiris
+ uiColor.lightOrangeWiris
+ uiColor.orangeWiris

WKWebView

<<Interface>>
JavascriptCommunicationProtocol

+ insertJavascriptFile(fileName: String)
+ initEditor()
+ getMathMLFromPng(): String
+ setMathML(mathml: String)
+ getCurrentFormula()
+ setToolbar(toolbarOption: ToolbarOpt
+ recognizeImage(base64: String)

UIImagePickerController

«interface»
UITableViewDelegate

«interface»
UITableViewDataSource

UILabel

UIButton

ConfigurationOptionCell

+ titleText: UILabel
+ chooseElement: UIView

UITableView

UITableViewCell

<<Interface>>
WKNavigationDelegate

+ didFinish()

<<Interface>>
WKScriptMessageHandler

+ didReceive()

<<Interface>>
UIDropInteractionDelegate

+ canHandle(interaction: UIDropInteraction): Bool
+ sessionDidUpdate(session: UIDropSession): UIDropProposal
+ performDrop(session: UIDropSession)

«**extension**»
String

+ localized: String

In order to keep strong references

ConfigurationOption

+ text: String
+ selection: UIViewController
+ «static» keepPickerViews: [UIViewController]

«**static**»
Language

+ setLang(lang: String)
+ getLang(): String

LanguageUIPickerView

ToolbarUIPickerView

<<Interface>>
LanguageUIPickerViewDelegate

+ onSelectLanguage()

<<Interface>>
ToolbarUIPickerViewDelegate

+ currentEditorToolbar: ToolbarOption

+ onSelectedToolbar(toolbarOption: ToolbarOption)

LanguageUIPickerView

ToolbarUIPickerView

+ currentEditorToolbar: ToolbarOption

+ onSelectedToolbar(toolbarOption: ToolbarOption)

<<Interface>>
UIPickerViewDelegate

UIPickerView

<<Interface>>
UIPickerViewDataSource

Glosario

bypass Forma de esquivar un sistema de seguridad informático; también, un enfoque distinto para solucionar un problema informático.

desktop Tipo de aplicación que se instala en el ordenador del usuario y que es ejecutada directamente por el sistema operativo.

LMS Paquete software usado para administrar uno o más cursos para uno o más alumnos. Normalmente un LMS está basado en una tecnología web que permite a los alumnos autenticarse, registrarse en cursos y completar dichos cursos realizando evaluaciones.

MathType MathType es una aplicación software creada por la empresa Design Science que permite la creación de notación matemática tanto en aplicaciones desktop como en aplicaciones web.

MFI Siglas que se refieren a MathType for iOS, software desarrollado en el transcurso de este TFG.

mock Son piezas de código que conforman la especificación de cómo se espera que se reciban llamadas específicas. Su uso es el de verificar el comportamiento de los objetos.

referencias fuertes Son aquellas referencias hacia objetos que provocan que el sistema no tenga permiso para liberar memoria de dichos objetos.

slide over Modo de visualización en el sistema operativo iOS que permite el multitasking superponiendo una ventana de una aplicación sobre otra aplicación y así poder interactuar con ambas a la vez si así se quiere.

split view Modo de visualización en el sistema operativo iOS que permite el multitasking dividiendo la pantalla en dos partes donde ejecutar aplicaciones diferentes pudiendo así interactuar con ambas a la vez si así se quiere.

stub Son piezas de código que proporcionan respuestas predefinidas a ciertas llamadas durante los tests, sin responder a otra cosa para la que no hayan sido programados. Su uso es el de verificar el estado de los objetos.

testing Fase en la cual se realizan pruebas del trabajo realizado con la intención de saber si existe algún tipo de error.

TFG Son las siglas de "Trabajo de Fin de Grado". En el contexto de l'*Universitat Politècnica de Catalunya* el TFG es el trabajo obligatorio que todo estudiante debe realizar en la última parte de la carrera para demostrar los conocimientos adquiridos a lo largo de su estudio en dicha universidad. Este trabajo puede realizarse bajo ciertas modalidades y condiciones.

Bibliografía

[1] Agencia Tributaria. *Tabla de coeficientes de amortización lineal*. [En línea; Acceso: 14-03-2019]. 2015. URL: https://www.agenciatributaria.es/AEAT.internet/Inicio/_Segmentos_/Empresas_y_profesionales/Empresas/Impuesto_sobre_Sociedades/Periodos_impositivos_a_partir_de_1_1_2015/Base_imponible/Amortizacion/Tabla_de_coeficientes_de_amortizacion_lineal_.shtml.

[2] Agilemanifesto.org. *Manifesto for Agile Software Development*. [En línea; Acceso: 18-02-2019]. s.f. URL: https://agilemanifesto.org.

[3] *Ant: The Definitive Guide, 2E (Covers Ant 1.6)*. Shroff Publishers & Distributors, 2005. ISBN: 9788184040807. URL: https://books.google.es/books?id=HzljPgAACAAJ.

[4] Apple Inc. *iWork - Apple*. [En línea; Acceso: 21-02-2019]. 2019. URL: https://www.apple.com/lae/iwork.

[5] Apple Inc. *Mac mini: Información sobre consumo energético y potencia térmica (BTU)*. [En línea; Acceso: 07-03-2019]. 2014. URL: https://support.apple.com/es-es/HT201897.

[6] Apple Inc. *WKWebView*. [En línea; Acceso: 05-05-2019]. 2019. URL: https://developer.apple.com/documentation/webkit/wkwebview.

[7] Ausbrooks, R. and Buswell, S. and Carlisle, D. and Chavchanidze, G. and Dalmas, S. and Devitt, S. and ... *Mathematical Markup Language (MathML) Version 3.0 2nd Edition*. [En línea; Acceso: 23-02-2019]. 2014. URL: https://www.w3.org/TR/MathML3/mathml.pdf.

[8] F. Buschmann y col. En: *PATTERN-ORIENTED SOFTWARE ARCHITECTURE: A SYSTEM OF PATTERNS*. v. 1. Wiley India Pvt. Limited, 2008. Cap. Model-View-Controller, págs. 125-168. ISBN: 9788126516117. URL: https://books.google.es/books?id=3C4DSkw-8AsC.

[9] S. Chacon. *Pro Git*. Books for professionals by professionals. Apress, 2009. ISBN: 9781430218333. URL: https://books.google.es/books?id=qJsXefpx1AUC.

[10] Dell Inc. *Product Safety, EMC and Environmental Datasheet*. [En línea; Acceso: 04-03-2019]. 2013. URL: https://i.dell.com/sites/doccontent/shared-content/solutions/en/Documents/insp_15_7000_7537_p36f001_us.pdf.

[11] Departament d'Organització d'empreses. *Gestión de proyectos - Módulo 2: Aspectos básicos de la Gestión de Proyectos*. [Documento inédito; Acceso: 05-03-2019]. s.f.

[12] Institute Electrical y Electronics Engineers. «Glossary of Software Engineering Terminology, IEEE Standard 610.12». En: (sep. de 1990). DOI: 10.1109/IEEESTD.1990.101064.

[13] Event reference. *Cross-Origin Resource Sharing*. [En línea; Acceso: 16-05-2019]. 2019. URL: https://developer.mozilla.org/en-US/docs/Web/Events.

[14] Facultat d'Informàtica de Barcelona. *Normativa del treball final de grau del Grau en Enginyeria Informàtica de la Facultat d'Informàtica de Barcelona*. [En línea; Acceso: 26-02-2019]. 2015. URL: https://www.fib.upc.edu/sites/fib/files/documents/estudis/normativa-tfg-gei-final.pdf.

[15] D. Flanagan. *Javascript*. Nutshell handbook. O'Reilly Media, Incorporated, 1996, págs. 1-2. ISBN: 9781565922426. URL: https://books.google.es/books?id=dUPGMgEACAAJ.

[16] E. Gamma y col. En: *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional Computing Series. Pearson Education, 1994. Cap. Strategy, págs. 315-324. ISBN: 9780321700698. URL: https://books.google.es/books?id=6oHuKQe3TjQC.

[17] E. Gamma y col. En: *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional Computing Series. Pearson Education, 1994. Cap. Factory Method, págs. 107-116. ISBN: 9780321700698. URL: https://books.google.es/books?id=6oHuKQe3TjQC.

[18] E. Gamma y col. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional Computing Series. Pearson Education, 1994. ISBN: 9780321700698. URL: https://books.google.es/books?id=6oHuKQe3TjQC.

[19] D.R. García, M.Á.S. Urbán y S.S. Alonso. En: *Ingeniería del software : un enfoque desde la guía SWEBOK*. Ibergarceta Publicaciones, S.L., 2011. Cap. Requisitos, pág. 116. ISBN: 9788492812400. URL: https://books.google.es/books?id=6zP%5C_ZwEACAAJ.

[20] D.R. García, M.Á.S. Urbán y S.S. Alonso. En: *Ingeniería del software : un enfoque desde la guía SWEBOK*. Ibergarceta Publicaciones, S.L., 2011. Cap. Requisitos, pág. 121. ISBN: 9788492812400. URL: https://books.google.es/books?id=6zP%5C_ZwEACAAJ.

[21] D.R. García, M.Á.S. Urbán y S.S. Alonso. *Ingeniería del software : un enfoque desde la guía SWEBOK*. Ibergarceta Publicaciones, S.L., 2011. ISBN: 9788492812400. URL: https://books.google.es/books?id=6zP%5C_ZwEACAAJ.

[22] Github contributors. *Swift (programming language)*. [En línea; Acceso: 07-05-2019]. 2019. URL: https://github.com/apple/swift.

[23] Google LLC. *Google Docs: Free Online Document for Personal Use*. [En línea; Acceso: 25-02-2019]. s.f. URL: https://www.google.com/docs/about/.

[24] Hays plc. *Guía del mercado laboral 2019*. [En línea; Acceso: 05-03-2019]. 2018. URL: http://image.email.hays.com/lib/fe4515707564057c751477/m/1/447662db-161d-466c-ab52-29adfde8f11a.pdf.

[25] Indeed. *Sueldos en Traductor/a en Barcelona*. [En línea; Acceso: 05-03-2019]. 2019. URL: https://www.indeed.es/salaries/Traductor/a-Salaries.

[26] Infologic Inc. *MathMagic, the ultimate Equation Editor on the planet!* [En línea; Acceso: 21-02-2019]. 2019. URL: http://www.mathmagic.com.

[27] Richard Jones, Antony Hosking y Eliot Moss. *The Garbage Collection Handbook: The Art of Automatic Memory Management*. Ene. de 2011, págs. 1-16.

[28] JS Foundation. *Appium: Automation for Apps*. [En línea; Acceso: 16-05-2019]. 2019. URL: http://appium.io/.

[29] Kohsuke Kawaguchi. *Jenkins*. [En línea; Acceso: 16-05-2019]. 2019. URL: https://jenkins.io/.

[30] Eero Laukkanen, Juha Itkonen y Casper Lassenius. «Problems, causes and solutions when adopting continuous delivery—A systematic literature review». En: *Information and Software Technology* (2017). ISSN: 0950-5849. DOI: https://doi.org/10.1016/j.infsof.2016.10.001. URL: http://www.sciencedirect.com/science/article/pii/S0950584916302324.

[31] Maths for More S.L. *MathType — WIRIS — math & science*. [En línea; Acceso: 21-02-2019]. 2019. URL: http://www.wiris.com/mathtype.

[32] Microsoft Corporation. *Microsoft Office*. [En línea; Acceso: 25-02-2019]. s.f. URL: https://www.office.com.

[33] Microsoft Corporation. *Microsoft Project*. [En línea; Acceso: 20-02-2019]. 2019. URL: https://products.office.com/en/project/project-and-portfolio-management-software.

[34] Microsoft Corporation. *Microsoft Word - Escribir una ecuación o una fórmula*. [En línea; Acceso: 23-02-2019]. s.f. URL: https://support.office.com/es-es/article/escribir-una-ecuaci%C3%B3n-o-una-f%C3%B3rmula-1d01cabc-ceb1-458d-bc70-7f9737722702.

[35] Microsoft Corporation. *Microsoft Word for iOS*. [En línea; Acceso: 21-02-2019]. 2019. URL: https://itunes.apple.com/us/app/microsoft-word/id586447913.

[36] Frank Mittelbach y col. *The LaTeX Companion 2nd ed*. Ene. de 2004. ISBN: 0201362996 (pbk. alk. paper).

[37] Moodle. *Moodle-Open-source learning platform*. [En línea; Acceso: 25-02-2019]. s.f. URL: https://moodle.org/.

[38] Notebookcheck. *Recensione completa del Tablet Apple iPad Pro*. [En línea; Acceso: 07-03-2019]. 2016. URL: https://www.notebookcheck.it/Recensione-completa-del-Tablet-Apple-iPad-Pro.158113.0.html.

[39] Notebookcheck. *Review Dell Inspiron 15-7537*. [En línea; Acceso: 04-03-2019]. 2018. URL: https://www.notebookcheck.net/Review-Dell-Inspiron-15-7537-Notebook.104976.0.html.

[40] PayScale Inc. *Technical Writer salary in Barcelona*. [En línea; Acceso: 05-03-2019]. 2019. URL: https://www.payscale.com/research/ES/Job=Technical_Writer/Salary/7d396eb3/Barcelona.

[41] James Robertson y Suzanne Robertson. «Volere Requirements Specification Template». En: (ene. de 2000).

[42] Ken Schwaber y Jeff Sutherland. En: *The Scrumm Guide*. [En línea; Acceso: 18-02-2019]. n.p, 2017. Cap. Definition of Scrumm, pág. 3. URL: https://www.leanagiletraining.com/bw/wp-content/uploads/2018/01/Scrum-Guide-US-2017.pdf.

[43] Ken Schwaber y Jeff Sutherland. En: *The Scrumm Guide*. [En línea; Acceso: 18-02-2019]. n.p, 2017. Cap. Scrumm Events, págs. 9-14. URL: https://www.leanagiletraining.com/bw/wp-content/uploads/2018/01/Scrum-Guide-US-2017.pdf.

[44] Ken Schwaber y Jeff Sutherland. En: *The Scrumm Guide*. [En línea; Acceso: 18-02-2019]. n.p, 2017. Cap. Scrumm Artifacts, págs. 14-17. URL: https://www.leanagiletraining.com/bw/wp-content/uploads/2018/01/Scrum-Guide-US-2017.pdf.

[45] Ken Schwaber y Jeff Sutherland. En: *The Scrumm Guide*. [En línea; Acceso: 18-02-2019]. n.p, 2017. Cap. Artifact Transparency, págs. 17-18. URL: https://www.leanagiletraining.com/bw/wp-content/uploads/2018/01/Scrum-Guide-US-2017.pdf.

[46] Ken Schwaber y Jeff Sutherland. *The Scrumm Guide*. [En línea; Acceso: 18-02-2019]. n.p, 2017. URL: https://www.leanagiletraining.com/bw/wp-content/uploads/2018/01/Scrum-Guide-US-2017.pdf.

[47] Ian Sommerville. En: *Software Engineering*. 9th. USA: Addison-Wesley Publishing Company, 2010. Cap. Software processes, págs. 29-32. ISBN: 0137035152, 9780137035151.

[48] Ian Sommerville. En: *Software Engineering*. 9th. USA: Addison-Wesley Publishing Company, 2010. Cap. Agile software development, págs. 56-58. ISBN: 0137035152, 9780137035151.

[49] Ian Sommerville. *Software Engineering*. 9th. USA: Addison-Wesley Publishing Company, 2010. ISBN: 0137035152, 9780137035151.

[50] Shakirat Sulyman. «Client-Server Model». En: *IOSR Journal of Computer Engineering* 16 (ene. de 2014), págs. 57-71. DOI: 10.9790/0661-16195771.

[51] The JUnit Team. *JUnit 5*. [En línea; Acceso: 20-05-2019]. 2019. URL: https://junit.org/junit5/.

[52] Trello.com. *Trello*. [En línea; Acceso: 20-02-2019]. s.f. URL: https://trello.com.

[53] W3C. *Cross-Origin Resource Sharing*. [En línea; Acceso: 07-05-2019]. 2014. URL: https://www.w3.org/TR/cors/.

[54] ZurApps. *MathPad - Equation Writer App*. [En línea; Acceso: 21-02-2019]. 2019. URL: http://zurapps.com/all/index.php/mathpad/mathpad.