

Cómo hacer un motor ECS en C++ desde 0

Grado en Ingeniería Multimedia



Trabajo Fin de Grado

Autor:

Laureano Cantó Berná

Tutor/es:

Francisco José Gallego Durán

Agosto 2022



**Guía para el Desarrollo de los Trabajos Fin de Grado y Fin de Máster.**

Este documento ha sido confeccionado por José Vicente Berná Martínez con el fin de facilitar a los alumnos de grado y máster el desarrollo de sus trabajos finales. Contiene además de los estilos, una descripción de los posibles apartados que pueden ser incluidos en el trabajo. Los alumnos pueden utilizarlo como plantilla, sustituyendo la explicación de cada apartado por sus propios contenidos, conservando formatos y estilos. El alumno puede personalizar el documento. Esta guía cumple con el Libro de Estilos para la presentación de memorias del Trabajo Fin de Grado/Máster de la Escuela Politécnica Superior de la Universidad de Alicante (<https://maktub.eps.ua.es/servicios/gestorContenidos/contenidos/normativaEPS/Pdf/9910.pdf)>

**Versión del documento**

2022.08.30

**Licencia**

Se permite la reproducción, distribución y comunicación pública de la obra, incluso con fines comerciales siempre y cuando reconozca y cite la obra de la forma especificada por el autor o el licenciante.

https://licensebuttons.net/l/by/4.0/88x31.png

# Resumen

# Motivación, justificación y objetivo general

La idea para desarrollar este trabajo viene precedida por un curso académico donde mi principal tarea fue desarrollar por primera vez un motor ECS para el videojuego que realizamos. Durante este tiempo, he estado interesado en el desarrollo de este tipo de patrones para el desarrollo de videojuegos. No fue mi principal idea, ya que durante el curso barajé otras posibilidades tal vez más orientadas a la parte gráfica de los videojuegos, pero al conocer y desarrollar mi ECS, supe que quería hacer algo relacionado con este tema.

Fue entonces, cuando decidí contactar con Francisco José Gallego Durán, mi profesor de cuarto curso encargado de enseñar esta tecnología. Al comentarle todo esto, apareció la idea de hacer un motor ECS y una demo hecha con este motor, pero al final la idea de Fran de hacer una guía sobre cómo hacerlo desde 0 con el lenguaje C++ me gustó mucho y fue la elección final de mi idea.

Es un tema interesante para desarrolladores con cierta habilidad ya en programación, cuya intención sea aprender a desarrollar este tipo de patrón de arquitectura de software en videojuegos, como por ejemplo futuros alumnos de nuestra carrera.

Es un proyecto viable, ya que parto de una experiencia previa, aunque no total en este campo, y me motiva el hecho de poder hacer algo útil para los demás como proyecto de fin de grado.

Por último comentar que no hay a penas contenido acerca de como elaborar desde 0 un ECS completo, por lo tanto será algo innovador donde podré demostrar las habilidades técnicas aprendidas durante mi carrera.

# Agradecimientos

# Citas

# Dedicatoria

# Índice de contenidos

[Resumen 3](#_Toc112858081)

[Motivación, justificación y objetivo general 4](#_Toc112858082)

[Agradecimientos 6](#_Toc112858083)

[Citas 7](#_Toc112858084)

[Dedicatoria 8](#_Toc112858085)

[Índice de contenidos 9](#_Toc112858086)

[Índice de figuras 11](#_Toc112858087)

[Índice de tablas 12](#_Toc112858088)

[1. Introducción 13](#_Toc112858089)

[2. Estudio de viabilidad 15](#_Toc112858090)

[2.1. Análisis DAFO 15](#_Toc112858091)

[2.2. Lean Canvas 17](#_Toc112858092)

[2.3. Análisis de riesgos 19](#_Toc112858093)

[3. Planificación 21](#_Toc112858094)

[4. Estado del arte. 23](#_Toc112858095)

[4.1. Antecedentes. 24](#_Toc112858096)

[4.2. Tecnologías para el desarrollo 25](#_Toc112858097)

[5. Objetivos 26](#_Toc112858098)

[6. Metodología 28](#_Toc112858099)

[7. Análisis y especificación 30](#_Toc112858100)

[8. Diseño 32](#_Toc112858101)

[8.1. Diseño de la persistencia 32](#_Toc112858102)

[8.2. Diseño arquitectura conceptual 34](#_Toc112858103)

[8.3. Diseño API Rest (si hay) 34](#_Toc112858104)

[8.4. Diseño arquitectura tecnológica Front/Back-end 35](#_Toc112858105)

[8.5. Diseño Interacción o Experiencia de Usuario - UX 36](#_Toc112858106)

[8.6. Diseño Interfaces - UI 39](#_Toc112858107)

[8.7. Guías de estilos 42](#_Toc112858108)

[8.8. Diseño de pruebas y validación 43](#_Toc112858109)

[9. Implementación 45](#_Toc112858110)

[10. Pruebas y validación 46](#_Toc112858111)

[11. Resultados 47](#_Toc112858112)

[12. Conclusiones y trabajo futuro 48](#_Toc112858113)

[Referencias 50](#_Toc112858114)

[Apéndice I 52](#_Toc112858115)

# Índice de figuras

[Figura 1. Opción Inicio>Numeración para crear listas numeradas 13](#_Toc112858116)

[Figura 2. Esquema de un análisis DAFO. 16](#_Toc112858117)

[Figura 3. Insertar título desde Referencias. 16](#_Toc112858118)

[Figura 4. Menú tipo de título. 17](#_Toc112858119)

[Figura 5. Cuadro para el análisis Lean Canvan 18](#_Toc112858120)

[Figura 6. Ventana para seleccionar la referencia que se desea insertar. 18](#_Toc112858121)

[Figura 7. Opción actualizar campos. 19](#_Toc112858122)

[Figura 8. Ejemplo de diseño de BD 33](#_Toc112858123)

[Figura 9. Diseño arquitectura conceptual panel IoT. 34](#_Toc112858124)

[Figura 10. Especificación de la Open API de Smart University 35](#_Toc112858125)

[Figura 11. Stack tecnológico del proyecto Smart University 36](#_Toc112858126)

[Figura 12. Ejemplo de la diferencia entre usable, accesible y experiencia de usuario. 38](#_Toc112858127)

[Figura 13. Ejemplo de mapa de interacciones. 39](#_Toc112858128)

[Figura 14. Ejemplo de wireframe con Balsamiq. 40](#_Toc112858129)

[Figura 15. Mockup desarrollado con Justinmind 41](#_Toc112858130)

[Figura 16. Diseño desarrollado con Sketch. 42](#_Toc112858131)

[Figura 17. Paleta de colores Unviersidad de Cádiz 43](#_Toc112858132)

[Figura 18. Ejemplo de ejecución de varias invocaciones a servicio. 44](#_Toc112858133)

[Figura 19. Primer boceto de la aplicación. 52](#_Toc112858134)

# Índice de tablas

[Tabla 1. Planificación temporal TFG 21](#_Toc112858135)

# Introducción

Este es el primer capítulo del trabajo y lo primero que se debe hacer es explicar el contexto donde se ubicará la temática. Un tribunal completamente ajeno a tu trabajo debe comprender el contexto de tu proyecto, la introducción debe explicar el marco informal sobre el que se desarrolla, el mundillo sobre el que versa tu proyecto, quienes están interesados en él, que solución puede implicar, que impacto puede tener lo que tu estas proponiendo.

En este apartado por ejemplo se puede hablar de algún estudio que deje constancia del problema y de su impacto en la sociedad. Entonces se puede comentar brevemente que hay empresas trabajando en la cuestión citando qué impresas y que hacen muy superficialmente. También se puede hablar de noticias recientes que apoyen la temática. Si ha un trabajo previo sobre el que se apoya también se puede comentar.

Uno de los grandes problemas de los alumnos es que no identifican los apartados que debe contener su trabajo fin de grado, a pesar de que lo han estudiado a lo largo de la carrera en las diferentes asignaturas. Es por eso por lo que se ha creado esta guía donde se proporciona una estructura genérica, consejos y recomendaciones.

Otro de los problemas es que acaban gastando mucho tiempo en el formateo correcto del documento porque no suelen invertir tiempo en conocer las herramientas con las que trabajan (Microsoft Word, Open Office…). El uso correcto de los estilos facilita mucho la redacción de los documentos, y más cuando son de gran tamaño.

Todos los títulos de primer nivel deben ir en el estilo “Titulo 1”. Además, a partir del capítulo introducción deben ir numerados utilizando la opción “Inicio>Numeración” como muestra la Figura 1.

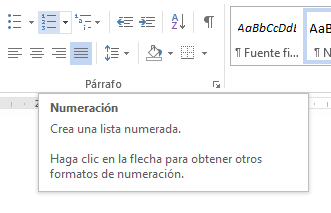


Figura 1. Opción Inicio>Numeración para crear listas numeradas

(Fuente propia)

Los títulos de segundo nivel deben utilizar el estilo “Titulo 2” y además también numerarse, pero utilizando un segundo nivel, esta opción está disponible también dentro de “Numeración” para crear listas numeradas.

Si es necesario se puede utilizar hasta un tercer nivel dentro del texto, aunque en este caso hay que considerar si es mejor dividir el capítulo en varios. Hasta un nivel 3 es aceptable, más de 3 niveles de numeración que originen capítulos del estilo “3.5.4.6.5. Titulo capítulo” no suele ser recomendable y lo que indica es que ese capítulo posiblemente debería haberse desglosado en varios capítulos más.

Cada apartado de nivel 1 debe comenzar en una nueva página, por lo que la final de la sección anterior introducir un “Insertar>Salto de página”. No se deben utilizar *intros* adicionales para separar secciones ya que, si se modifica un texto acortándolo o alargándolo, se descuadrará todo el documento. Tampoco se deben introducir *intros* adicionales entre párrafos, ya que los espacios pueden controlarse simplemente modificando el estilo del texto “Normal” y cambiando el espaciado posterior del párrafo.

El tamaño de la introducción también puede ser muy variable, desde un par de páginas hasta las que sean necesarias, dependerá de la complejidad. El alumnado debe tener en cuenta que se trata de explicar el problema y el contexto del problema.

# Estudio de viabilidad

La sección de estudio es opcional, se incluirá solo si es adecuado al proyecto. En esta sección se puede hacer un estudio de viabilidad. Antes de arrancar con el resto del proyecto se puede analizar un poco si el proyecto en sí mismo, sus pretensiones u objetivos son viables, son pertinentes o son necesarios. También es muy adecuado indicar los riesgos y los planes de contingencia.

Para hacer este estudio de viabilidad se pueden utilizar herramientas como las utilizadas la carrera como el análisis DAFO, el Lean Canvas, y el análisis de riesgos. El alumno o alumna puede utilizar cualquier otra herramienta que le guste más o crea conveniente, incluso valorando opciones y recoger una reflexión inicial sobre cómo debería abordarse este capítulo.

Un estudio de viabilidad es adecuado cuando se está realizando un proyecto disruptivo, un proyecto que trata de generar solución sobre un problema que no la tiene, un proyecto que plantea un software que mejora o cambiar lo existente, etc. Pero si por ejemplo se trata de un encargo (un TFG que ha demandado un profesor, o una colaboración con una empresa), o se trata de un corto de animación, una investigación concreta sobre un algoritmo, etc., es posible que no sea necesario hacer un estudio de viabilidad o utilizar todos los subapartados.

El tamaño total de esta sección dependerá del contenido.

Los subapartados no saltan de página, van seguidos dentro de cada sección por lo que no debe incluirse salto de página.

## Análisis DAFO

El DAFO (de las iniciales de Debilidades, Amenazas, Fortalezas y Oportunidades) es una metodología de estudio de la situación de un proyecto, analizando sus características internas (Debilidades y Fortalezas) y su situación externa (Amenazas y Oportunidades) en una matriz cuadrada como muestra la Figura 2. Esquema de un análisis DAFO.



Figura 2. Esquema de un análisis DAFO.

(Fuente http://egesoftware.blogspot.com.es)

El alumnado debería analizar los cuatro cuadrantes, explicando con texto cada sección y luego componer juna imagen una imagen a modo de resumen. No debe incluirse solo la imagen, sino explicar extensamente cada elemento que vaya a contener cada cuadrante y luego crear el resumen.

Las figuras siempre deben contener un pie de figura explicando su contenido tal como muestra la Figura 2. Además, deben estar siempre referenciadas en el texto tal, es decir, siempre se debe hablar de ellas o nombrarlas en el texto. Las imágenes sirven para acompañar a la explicación, para ilustrar alguna idea o concepto que sin la imagen sería más complicado explicar. No se deben incluir imágenes, figuras, fotos u otros elementos gráficos por el simple hecho de rellenar y hacer “bonito”. Todas las figuras deben indicar la fuente de la que han sido extraídas o bien si son propias. No hacer el reconocimiento de fuente y utilizar la imagen de otra persona apropiándosela como propia es plagio.

Para crear los pies de figura se debe utilizar la opción “Referencias > Insertar título”, como muestra la Figura 3.

Interfaz de usuario gráfica, Aplicación, Word

Descripción generada automáticamente

Figura 3. Insertar título desde Referencias.

(Fuente propia)

En el submenú que aparece dentro de Insertar Titulo, se debe seleccionar el rótulo “Figura”. Si no existe el rótulo Figura se puede crear “Nuevo rótulo”. De esta forma las figuras estarán numeradas de forma automática.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Figura 4. Menú tipo de título.

(Fuente propia)

Después de añadir el título “Figura X. Descripción”, se debe añadir otro texto con la fuente. A este texto de fuente se le aplicará el estilo “Fuente figura-tabla” lo que hace que parezca que forma parte del texto que describe la figura, pero en realidad son dos párrafos separados.

## Lean Canvas

Lean Canvas es otra de las herramientas que pueden ser idóneas para analizar un producto que está siendo creado y que tiene carácter innovador, ya sea porque es una solución nueva a un problema ya existente o bien porque es una mejora sobre otras soluciones que ya existen. La idea de Lean Canvas es analizar el proyecto desde diferentes perspectivas de interés como muestra la Figura 5, no solo desde el desarrollo o los costes.

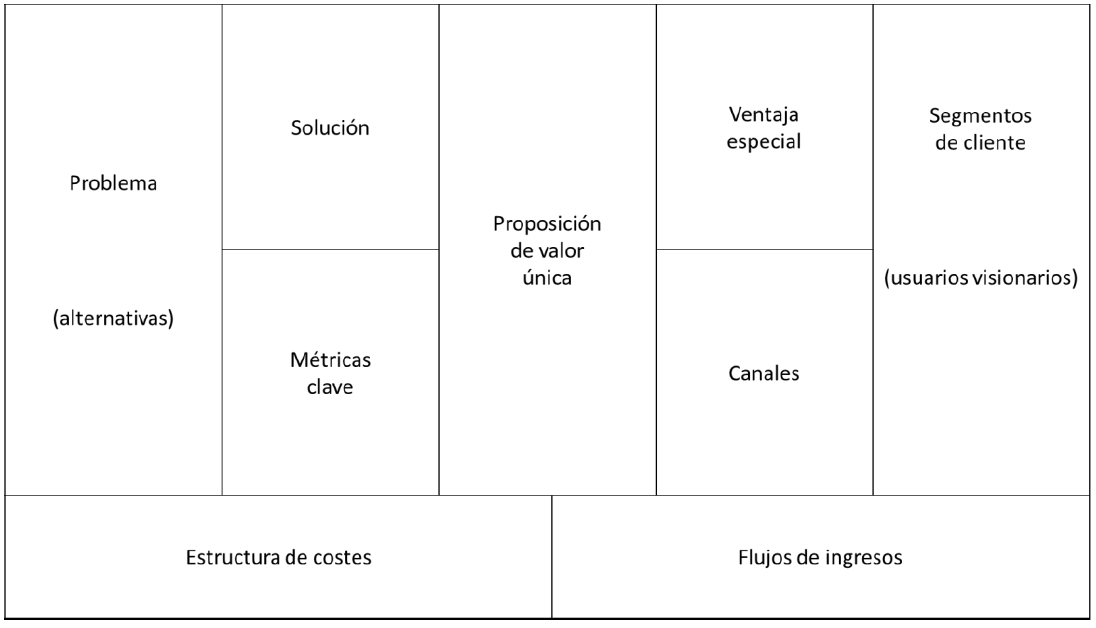


Figura 5. Cuadro para el análisis Lean Canvan

(Fuente propia)

Igual que con el DAFO, el se debe explicar con texto cada una de las secciones, describiendo detenidamente los contenidos, y al finalizar entonces crear el cuadro resumen que recoja una síntesis del Lean Canvas.

Para hacer referencia en el texto a una imagen se debe hacer la opción “Referencias>Referencia cruzada”, aparecerá una ventana como muestra la Figura 6, entonces seleccionar “solo rótulo y número” e insertar.

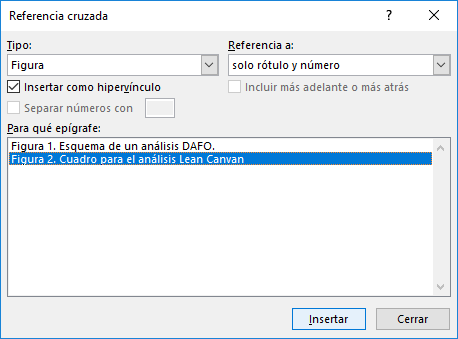


Figura 6. Ventana para seleccionar la referencia que se desea insertar.

(Fuente propia)

De esta manera si más adelante insertamos una nueva imagen, la numeración se mantendrá correlativa y los índices funcionaran adecuadamente, en el texto se referencia a la imagen. Si bien es cierto, que, si insertamos una imagen anterior y cambia la numeración, por ejemplo insertamos una figura 1 que desplaza la numeración de todas las figuras, deberemos buscar en el texto todas las referencias cruzadas, seleccionarlas, pulsar el botón derecho y seleccionar “Actualizar campos” para que se actualice la numeración, como muestra la Figura 7.

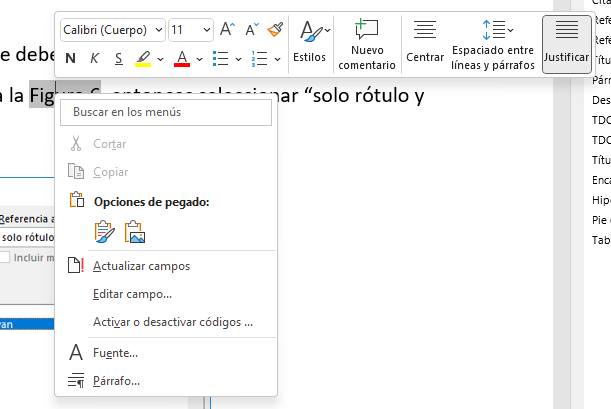


Figura 7. Opción actualizar campos.

(Fuente propia)

## Análisis de riesgos

A lo largo de un proyecto que es de carácter anual (12 créditos equivalen a unas 300 horas), es adecuado que antes de iniciar nada, el alumno pueda tener en consideración riesgos como son el desconocimiento de tecnologías, posibilidad de enfermar levemente durante el curso, sufrir variaciones de carga de trabajo debido a asignaturas, etc. La propuesta de planes de contingencia puede facilitar el éxito del trabajo fin de grado. Para ello lo que se establece es una lista de riesgos, potenciales circunstancias que pueden afectar negativamente al proyecto. Para cada una de ellas se establece una probabilidad de ocurrencia y un nivel de efecto, y posteriormente se generan planes de prevención y planes de contingencias.

Sobre la probabilidad. Lo que se debe hacer es crear una escala de probabilidad, por ejemplo: alta, media y baja. Pero para cada nivel de la escala describir qué significa, por ejemplo, alta implica que va a ocurrir 1 vez al mes, media que ocurrirá una vez al año, y baja que ocurre una vez cada 5 años. Erróneamente a veces se indican porcentajes como: alta es un 30%, media es un 15% y baja un 5%. Pero un porcentaje solo tiene sentido si hay una cantidad base de referencia, por ejemplo 10 veces al año. Entonces es mejor indicar una descripción de que quiere decir cada uno de los ítems de la escala.

Sobre la gravedad igual, se debe proporcionar una escala, por ejemplo: alta, media, baja. Esto indica la gravedad en caso de que se produzca el riesgo. Y se debe describir lo que implica cada nivel, por ejemplo: gravedad alta implica que el proyecto se va a detener durante al menos 2 semanas.

En las escalas se pueden utilizar 3, 4 o 5 niveles, queda a discreción del alumnado.

Sobre los planes para evitar los riesgos se deben considerar dos posibilidades: planes de previsión y planes de contingencia. Los planes de previsión son aquellas cosas que se pueden hacer para evitar que un riesgo ocurra o para que en caso de que ocurra minimizar sus efectos. Son acciones a priori como, por ejemplo: crear un repositorio de apoyo para el código, tener una copia de las versiones del documento, basarse en plantillas o códigos de ejemplo robustos, etc. Los planes de contingencia son las acciones que se han de ejecutar una vez que se materializa el riesgo, por ejemplo: recuperar la última versión del repositorio, incluir una consulta en la web de expertos de XXX…

El análisis de riesgos busca que, como profesionales, antes de comenzar un proyecto, tengamos muy claro qué puede perjudicar al proyecto y establezcamos estrategias para evitarlo. No tiene por que ocurrir ninguna amenaza durante el proyecto y por tanto no entrará en marcha ningún plan. En caso de ocurra debemos documentarlo. Cualquier cosa que pueda afectar al proyecto es una amenaza, ejemplos: perdida de código, rotura de hardware, tener pendiente un examen de una asignatura (porque si suspendemos tenemos que repetir), tener intervenciones pendientes, estar pendiente de cambio de trabajo o posibilidad de comentar…

# Planificación

Una buena planificación desde el principio puede ayudar también en la consecución del proyecto. Además, permitirá una vez terminado, analizar junto a los resultados la adecuación de la planificación al desarrollo completo del proyecto. Esto dará experiencia a los alumnos ya que al planificar se realiza una estimación de costes temporales en función de unas capacidades o habilidades supuestas. Después, al analizar los resultados podrás comparar si subestimaste tus capacidades o no. La planificación temporal además te ayudará a seguir unos pasos en el tiempo, no retrasarte y tampoco invertir demasiado tiempo en aquello que no debe. Por ejemplo, el estado del arte es una sección que puede tomar mucho tiempo si no se acota pues, por lo que es bueno marcarse plazos concretos.

A continuación, se proporciona la Tabla 1 como ejemplo de una planificación para un proyecto de desarrollo de TFG convencional de software. Dependerá de los apartados de tu proyecto y de tus propias habilidades, pero esta distribución temporal permitiría el desarrollo del proyecto en plazos para su entrega en la convocatoria de junio. Se hace una agrupación de varios capítulos, el alumno debería hacer el esfuerzo de deducir cuanto tiempo debería emplear en cada uno de ellos y hacer una planificación detallada.

Tabla 1. Planificación temporal TFG

|  |  |  |
| --- | --- | --- |
| **Contenidos** | **Tiempo total** | **Fecha límite fin** |
| Motivación, introducción, estudio de viabilidad, estado del arte. | 1 mes | 15 noviembre |
| Objetivos  Metodología  Análisis y especificación  Presupuesto, estimaciones, planificación | 1 mes | 20 diciembre |
| Diseño | 2 meses | 25 febrero |
| Implementación | 1 mes | 15 abril |
| Pruebas y validación  Resultados  Conclusiones y trabajo futuro  Referencias, bibliografía y apéndices  Agradecimientos, citas, índices | 1 mes | 1 junio |

Este ejemplo de planificación es para tener una versión de proyecto legada la fecha de 1 de junio y así poder entregar adecuadamente para esa convocatoria. Cada fecha límite marca el plazo en el que debería ser entregada una versión de la memoria con esos capítulos completados al tutor para que este pueda corregirla y proporcionar feedback al alumno.

Las tablas también han de contener un título de la forma “Tabla X…” y han de estar referenciadas en el texto. Para crear el título de tabla también se utiliza “Referencias>Insertar título”, pero en este caso las tablas ubican el título antes de la tabla. El formato de la tabla debe conservarse constante a lo largo de todo el documento y deben ser claras y visibles al imprimirse en blanco y negro, es decir, todas las tablas deben tener el mismo formato y no se debe abusar de los colores si son empleados.

# Estado del arte.

El estado del arte es una de las secciones más importantes del trabajo y debe ser desarrollado antes si quiera de diseñar ningún tipo de solución, ya que permitirá al alumno conocer más información tanto sobre el problema como de las posibles soluciones actuales o en proceso.

La idea del estado del arte es recoger toda aquella información que pueda resultar de interés para el desarrollo del proyecto. Puede por tanto contener varios apartados dentro del propio estado del arte. Si la investigación es muy extensa puede ser recomendable dividir la sección estado del arte en varias secciones como “Estudio de la problemática”, “Antecedentes”, “Tecnologías para el desarrollo”. Todo depende de lo extensa y profunda que sea este apartado.

La sección de estado del arte es un estudio sobre la problemática concreta que aborda el proyecto. Antes de proponer una solución a un problema es necesario conocer en profundidad tanto el problema como su contexto, los actores que intervienen, los recursos disponibles, el impacto de dicho problema. Por ejemplo, si tenemos un TFG que trata sobre cómo hacer un buen marketing digital de una plataforma web concreta, sería bueno estudiar los fundamentos del marketing digital y para ello podríamos recurrir a diversos libros como el de José Sixto García [1], analizando cuales pueden ser dichos factores a favor o en contra. También se podrían buscar ejemplos de otros trabajos relacionados con el tema y analizar cuáles fueron las propuestas y los resultados [2].

Esta sección debería estar plagada de referencias a textos, libros, artículos y otros materiales de relevancia ya que un estado del arte recoge el conocimiento existente, es decir, lo que ya está escrito y asentando en la industria. Es muy importante hacer un buen estado del arte, **siempre**. Antes de comenzar cualquier proyecto el primer paso siempre ha de ser una investigación previa que nos ayude a conocer la problemática en profundidad y a convertirnos un poco en expertos en el ámbito. Un error muy común es tener una idea inicial y cegarse en implementar eso sí o sí, sin investigar primero cual es el trasfondo de la problemática y sobre todo qué se ha intentado hacer antes al respecto. Es muy raro que, en un TFG, un ingeniero o ingeniera novel detecte un problema sobre algo que nunca nadie antes ha reflexionado ni intentado solucionar. Por lo tanto, si somos capaces de detectar cómo esta el tema, nuestro punto de partida para la resolución del proyecto no será cero y evitaremos cometer errores ya superados.

El apartado “Estado del Arte” sirve para recoger todo el *know who* del tema. Pensad que puede hablarse de marketing, pero también podríamos hablar sobre la problemática de detectar si un balón entra o no en una portería mediante el procesamiento de imagen, sobre el desarrollo de actividades de ocio digital por parte de personas con movilidad reducida o cualquier otro tema. Por ello hay que recoger todo el conocimiento que se pueda para ser experto en la materia y así poder después hace una propuesta sensata y útil.

Muchos de los grandes errores que los alumnos comenten en su propuesta de un TFG/TFM se derivan precisamente de tomar decisiones sin conocer suficientemente el problema al que se enfrentan, su contexto, sus necesidades, limitaciones físicas o incluso aspectos legales. Todo ello debe recogerse en esta sección para asegurarnos de que el trabajo será realista.

También es importante recoger propuestas de solución alternativas, describirlas e incluso compararlas, extrayendo sus principales características. Esto puede marcar tendencias del sector que en nuestra propuesta tendremos que implementar. Por ejemplo, es muy común en cualquier aplicación que realiza fotos tener acceso directo a funcionalidades de compartir, esto implica que, aunque inicialmente no esté dentro de nuestro problema igual debemos considerar incluir funcionalidades como estas para respetar la norma de facto del sector.

Hay que tener mucho cuidado con referenciar posts o artículos online de sitios que no tengan reputación. Se pueden utilizar materiales online pero las fuentes deben ser de calidad, es decir, que el autor tiene que tener experiencia demostrada en el tema, ser un medio de comunicación fiable y su contenido ser coherente y creíble. Las fuentes pueden estar en cualquiera de las lenguas oficiales de la Universidad de Alicante (Español, Inglés o Valenciano). Todas las referencias deben estar recogidas en el apartado final de referencias y bibliografía. Para insertarlas en el texto siempre se utilizará “Referencias>Referencia cruzada” para insertarla.

## Antecedentes.

Si el estado del arte trata sobre la problemática del trabajo, los antecedentes son una sección que recoge los trabajos y experiencia previos del alumno con la problemática. Por ejemplo, si el alumno ha realizado unas prácticas en un curso anterior o en una asignatura a partir de las cuales ya se ha avanzado en la solución o tiene una versión previa de la solución que servirá como punto de partida, este es el apartado donde se explicará. Se debe dejar muy claramente explicado cual es el trabajo previo realizado para que el tribunal tenga clara después cual ha sido la aportación del alumno en su trabajo respecto a lo ya existente.

Este apartado de estado del arte y su contenido puede extenderse todas las páginas que sea necesario, no se recomienda ni mínimo ni máximo pues dependerá de la profundidad con la que se trate este capítulo.

## Tecnologías para el desarrollo

Suele ser muy común también analizar las posibles tecnologías para el desarrollo del proyecto. Esta sección es adecuada si de verdad se van a analizar tecnologías y herramientas para decidir cual será utilizada. Esto implica por ejemplo hablar sobre principales características de un lenguaje, sobre la arquitectura de las aplicaciones, tiempos de desarrollo, curva de aprendizaje, tamaño de fuentes y compilados, posibilidad de multiplataforma, ventajas en el despliegue o cualquier otra condición que pueda favorecer o impedir el proyecto. No se trata de nombrar o hablar en general de las tecnologías y nunca se debe copiar por ejemplo la descripción genérica de un software como por ejemplo Photoshop o InDesig, o de una framework como VueJS y Angular. Por ejemplo, si se está desarrollando un TFG en el que se trabaja sobre imágenes y va a ser necesario utilizar software para retoque de imágenes, podemos analizar las principales opciones del mercado como son Photoshop y GIMP, sobre ellos podemos analizar sus funcionalidades, semejanzas y aspectos que los diferencias, precios y licencias, facilidad de uso, plugins, requerimientos hardware, etc. Esto puede ayudarnos a tomar una decisión cuando en el apartado de diseño vayamos a proponer un stack tecnológico, o bien al final de la propia sección de análisis.

No se deben incluir logos de tecnologías para rellenar, ni fragmentos de texto copiados desde webs oficiales o posts que sean meramente divulgativos, pues un TFG es un documento académico de carácter científico, en el que se debe recoger análisis y síntesis.

# Objetivos

En este apartado se debe definir cuál es el objetivo principal del proyecto, u objetivos principales, y los subobjetivos de cada uno de estos. Los objetivos definen cuales son las metas de nuestro proyecto. No son funcionalidades. A partir de estos objetivos se dispondrán después recursos, diseños y tomas de decisión. La consecución de los objetivos supondrá la superación de las dificultades o problemas inicialmente planteados.

Los objetivos definen la pretensión del proyecto (no la funcionalidad), y es interesante que cumplan una serie de características para asegurarnos de que nos aportan valor. Una estratégica interesante es hacer que cumplan el principio SMART propuesto por Doran [Error: no se encontró el origen de la referencia], según la cual cada objetivo debe ser:

* Specific (Específico): debemos especificar y concretar al máximo cada uno de nuestros objetivos, con el mayor nivel de detalle posible. Si no sabes si un objetivo tiene suficiente detalle, hazte esta pregunta: con los datos que aportas, ¿podría saber otra persona ajena al proyecto qué vas a hacer y cómo?
* Measurable (Medible): en ocasiones resulta difícil pensar en un objetivo que podamos medir, pero tiene que quedar muy claro que nuestras metas deben serlo. Solo de esa manera podremos cuantificar si alcanzamos los resultados esperados en función de los indicadores que establezcamos para cada uno de ellos. Recuerda siempre que tus objetivos sean cuantitativos; si no se puede medir, será difícil saber si has alcanzado el éxito.
* Attainable (Alcanzable): está claro que nuestro objetivo ha de ser alcanzable en la situación en la que nos encontremos. Fijar objetivos que son imposibles de conseguir no tiene ningún sentido, y además puede mermar nuestra moral. Piensa en tus objetivos como en retos ambiciosos, pero posibles. Además, siempre debe existir la posibilidad de que puedan ser reajustados si nuestro entorno sufre cambios. Intenta evitar objetivos “globales”, es decir, de los que están destinados a alcanzar “toda la población mundial”.
* Realistic (Realista): el objetivo debe ser realista y estar orientado a resultados objetivos y concretos. Como ya hemos mencionado, que estén dentro de nuestras posibilidades, por lo que hemos de tener en cuenta factores ajenos como nuestro entorno o los recursos de los que disponemos. Aquí los análisis DAFO o de riesgos anteriores pueden ser interesantes,
* Timely (Tiempo): todo objetivo tiene que tener un horizonte temporal. Por ejemplo: si quieres conseguir 10.000 visitas a tu sitio web... quieres lograrlo ¿en un mes? ¿en un año? Hay mucha diferencia, por eso hemos de definir muy bien el horizonte de tiempo para cada uno de nuestros objetivos. Si es un objetivo a largo plazo, es recomendable hacer un esquema o línea temporal donde podamos marcar los distintos hitos o etapas que nos llevarán a conseguir nuestra meta final.

No olvides que si uno de tus objetivos está ligado a la monetización de tu solución (ya sean soluciones de pago, freemium, licencia, venta del producto, etc), debe estar contemplado como un objetivo. Si no es un objetivo después no deberían aparecer funcionalidades como por ejemplo el pago a través de una pasarela de la licencia.

Es recomendable que tu proyecto contemple uno o como máximo dos objetivos principales, y a partir de estos se marquen subobjetivos. Veamos ejemplos:

* Sí tu TFG es desarrollar un corto de animación, el objetivo principal sería el desarrollo de un cortó de animación de temática tal, y algunos de los subobjetivos podrían ser investigar sobre cortos anteriores relacionados (imagina que el corto va a tratar sobre el bulín), conceptualizar y caracterizar los personajes o elementos, modelarlos y texturizarlos, difundir el corto en algún medio o certamen, etc.
* Si tu TFG es generar una aplicación tipo recordatorio de tus medicamentos, este sería el objetivo principal, y como subobjetivos se pueden establecer el obtener las funcionalidades más comunes de otras aplicaciones existentes, diseñar la experiencia de usuario para un tipo de usuario concreto, diseñar el apartado gráfico, desplegar sobre infraestructuras, subir a play store, etc.

Tus objetivos y subobjetivos marcarán tu trabajo. Si estableces que quieres por ejemplo enviar tu corto a un certamen que se celebra en marzo, deberás planificar y desarrollar tu TFG para que esté listo en esas fechas. Si tu subobjetivo es desplegar en play store, entonces tendrás acciones encaminadas a ello que formarán parte de tu TFG.

Este capítulo se extiende habitualmente 1 o 2 páginas. Pueden ser más.

# Metodología

Describir los procedimientos mediante los cuales se va a desarrollar el trabajo, esta sección describe “el cómo” se va a trabajar. Dependiendo del tipo de proyecto se podrán utilizar unas metodologías u otras o incluso combinación de varias. Metodologías ágiles tipo Scrum o Lean pueden ayudar a asimilar cambios y dinámicas en el transcurso del trabajo mientras que otras más tradicionales como la Cascada puede ayudar a eliminar la incertidumbre desde el principio generando mayor confianza en el proyecto. El alumno debe describir cómo será su procedimiento. Utilizar una metodología ágil no quiere decir no generar documentación ni que no haya fases de análisis, especificación o diseño. Todas estas fases en mayor o menor medida se dan a lo largo de un proyecto, la metodología solo indica cuándo se desarrollarán y la profundidad y completitud de estas fases. Por ejemplo, en una metodología Scrum se describen historias a partir de las cuales de definen funcionalidades que son diseñadas y planificadas en mediante esprints. En una metodología en cascada todo el análisis, especificación y diseño se realizan en bloque y por completo al inicio del proyecto para luego pasar a desarrollar la solución. Si se trata de una investigación donde se ha de generar un nuevo algoritmo se puede utilizar el método hipotético-deductivo para describir soluciones y luego demostrarlas, o si se basa por ejemplo en la imitación de un organismo celular se podría utilizar el método inductivo.

La metodología o metodologías por emplear las define en alumno, según le sea más cómodo. Se trata de describir cómo va a trabajar para que no sea una improvisación continua y se ordene la secuencia de pasos que va a realizar.

También es una buena práctica definir e indicar que herramienta o herramientas serán utilizadas para gestionar el proyecto, contabilizar las tareas realizadas y llevar un control de aquellas pendientes. Estas herramientas deberán permitir desarrollar el trabajo según la metodología escogida. Trello, Microsoft Project, JIRA, Toggle, son solo algunas de estas herramientas, justifica además tu elección.

Este apartado puede extenderse 1 o 2 página, dependiendo de la justificación o desarrollo que se haga incluso más.

Las siguientes secciones dependerán del tipo de proyecto, algunas posibles estructuras serían:

* Desarrollo software: análisis y especificación; diseño y dentro de esta sección todos los apartados de diseño o bien hacer un apartado distinto para cada uno; implementación donde se trataría de documentar los sprints o fases del desarrollo; pruebas y validación, donde se documentan las pruebas que se han realizado para asegurar que el código está libre de errores y se valida que la solución de verdad resuelve el problema; despliegue y puesta en producción; etc.
* Creación de un corto: preproducción con definición de idea, storyboard, guion y arte; producción describiendo las herramientas, sonido, modelado, texturizado, riggin, escenarios, animación, iluminación, cámaras y render entre otros; posproducción con todas las acciones que se realizan para rematar el trabajo.

Se va a extender como ejemplo algunas secciones de un posible desarrollo software, pero se pueden buscar múltiples ejemplos en el repositorio de la UA de Ingeniería Multimedia (<https://rua.ua.es/dspace/handle/10045/36932>), solo es cuestión de buscar TFGs de la temática similar a la nuestra.

# Análisis y especificación

En esta sección nos centramos en el problema concreto que queremos solución y cuál va a ser el alcance la solución a desarrollar. Dicho alcance depende de muchos factores como por ejemplo los usuarios que lo utilizarán, la población a la que está orientada, la forma en la se distribuirá en función de sus necesidades, etc. En general y para no dejarnos elementos sin definir es bueno utilizar alguna herramienta de análisis y especificación, como las estudiadas durante la carrera de IEEE 830. Seguir esta plantilla facilita una definición sistemática de aquellos aspectos que son necesarios tener en cuenta a la hora de diseñar nuestra solución software. Ahora sí que se definirán aspectos relativos a requerimientos de rendimiento, de capacidad, funcionales, de interfaz con sistemas de terceros, limitaciones. Puede utilizarse como guía el artículo [5].

Como resultado principal de esta sección se ha obtener una lista de requerimientos funcionales y no funcionales deseados, que pueden especificarse en como un conjunto de tablas. Estos requerimientos deben estar identificados con un identificador único de forma que después, en la fase de diseño, se pueda indicar claramente como están siendo resueltos. Los requerimientos funcionales pueden ser identificados con los identificadores RF1, RF2, etc., mientras que los no funcionales pueden ser identificados con RNF1, RNF2, etc. No pueden incluirse requerimientos que no estén debidamente justificados y relacionados con el problema, de igual forma que luego el diseño posterior responderá exclusivamente a las necesidades de requerimientos establecidos. Aquí es donde se deben tener en cuenta los objetivos y subobjetivos. Si por ejemplo se ha marcado que un subobjetivo es explotar la solución con opciones fremium, entonces serán necesarios requerimientos funcionales del estilo: poder realizar suscripción, bloquear funcionalidades que se consideran de las de pago, cancelar suscripción, etc.

Algunos requerimientos deberían estar de serie en cualquier sistema como son:

* El control de errores: aportar mecanismos que permitan informar tanto al usuario como a los administradores de los errores que se hayan producido, a cualquier nivel, ya sean de acceso a BD, de acceso a disco, de acceso a algún servicio de tercero, errores de login, etc.
* Auditoría: deberían aportarse mecanismos suficientes para proveer de mecanismos de auditoría a los administradores y así poder observar comportamientos sospechosos de los usuarios, de agentes externos, mal funcionamiento del propio sistema. Estos mecanismos son históricos y logs a diferentes niveles como sistemas y aplicación.
* Alineación con la legalidad: dependiendo de la información a almacenar es necesario cumplir con los requerimientos legales marcados por la legislación a la que se verá sometida. Se necesario detectar estos requerimientos y dejarlos especificados claramente ya que pueden condicionar nuestra infraestructura, desarrollo, funcionalidades, etc.
* Requerimientos no funcionales: los relacionados con la mantenibilidad del sistema como son su flexibilidad, su capacidad, su modularidad. Definirla en función de las necesidades reales. Si un software está destinado a ser utilizado por los miembros de una PYME no tiene los mismos que si es un servicio destinado a todos los jóvenes españoles de entre 14 y 25 años, y por tanto estos aspectos no funcionales tampoco son lo mismo.

Esta sección contendrá todas las páginas que sea necesaria para dejar bien especificado los requerimientos del proyecto. Se deben especificar todos los requerimientos funcionales y no funcionales con independencia de que vayan a dar tiempo de ser implementados. Después durante el diseño o implementación se puede crear un plan de priorización donde vayamos a dar más peso a unos requerimientos y por tanto algunos no sea posible implementarlos.

En esta sección se deberían describir a los usuarios que utilizarán el sistema para luego definir los requerimientos que cada uno tiene. Habitualmente se habla de usuarios administradores (quien instala y mantiene la solución) y usuarios clientes (quien paga por utilizar la aplicación) Puede haber multitud de usuarios. Es importante cuando se habla de requerimientos funcionales, indicar a que usuario afecta.

# Diseño

El capítulo de diseño es con diferencia el más importante del TFG. Podría decirse que es el corazón de dicho trabajo. En este punto se ha de diseñar la solución de forma que dé respuesta a todos y cada una de los requerimientos funcionales y no funcionales anteriormente establecidos. Por tanto, en esta sección se hará constante referencia a los identificadores de los requerimientos que se están abordando. Hay que tener en cuenta que un requerimiento puede necesitar de diferentes elementos en un software para ser resuelto. Por ejemplo, si uno de los requerimientos es el control de acceso posiblemente requerirá de dotar de identificación mediante usuario/contraseña al sistema, y por tanto necesitaremos un sistema de gestión de usuarios y permisos de usuario compuesto por una interfaz gráfica, posiblemente algún tipo de API, web service u otro acceso a datos, algún sistema de persistencia (base de datos, motor de base de datos, etc.), control de errores, cifrado de las comunicaciones a nivel de infraestructura y cifrado de datos a nivel de BD, etc. Se deberán aportar diseños para cada uno de estos aspectos.

El diseño de soluciones es muy complejo y en él intervienen aspectos a distintos niveles o vistas de un proyecto. Una buena práctica es agrupar el diseño en apartados en función de la materia o el aspecto que se está definiendo, concentrando así todo lo relativo a ese contexto en un único apartado. Una aplicación será la suma en realidad de todos esos diseños. El resto de apartados pueden ser subapartados dentro de diseño, o ser apartados de primer nivel, esto es a discreción del alumnado. Además estos apartados son una sugerencia, pueden ser estos, pueden eliminarse aquellos que no procedan o añadirse algunos si se considera importante.

## Diseño de la persistencia

La persistencia es el aspecto relativo a todo lo concerniente al almacenamiento de datos de nuestra propuesta. Hay que entender que esto debería incluir los siguientes aspectos:

* Almacenamiento de datos: analizar y detectar todos los campos y datos a almacenar, sus tipos, sus tamaños, sus relaciones. Diseñar las colecciones o tablas en las que deban ser almacenados. Definir el modelo de base de datos para ellos: SQL, noSQL, documental, estática, dinámica, orientadas a objetos, etc. Leer el artículo sobre “Bases de Datos” [6] para tener en mente que hay muchas características que definen las bases de datos y que es necesario definir una correctamente en función de los datos que manejamos. Además, indicar la necesidad de cifrado o no de los datos y el tipo, según la legislación o según los requerimientos.
* Almacenamiento de datos de administración o metadatos: definición de los datos que serán almacenados referentes a la administración del sistema como los logs, la trazabilidad de acciones, los errores, la auditoría. Definir igual que antes estructuras, modelos de bases de datos, motores, definición de archivos y sus estructuras, etc.
* Seguridad e integridad: definir los aspectos relativos a las políticas de seguridad para acceso al servicio de base de datos, la infraestructura donde será instalada, cómo se establecerán los mecanismos de copias de seguridad o incluso de detección de errores en la integridad, si se proveerá de herramientas que generen alertas o avisos sobre estas situaciones, etc.

Todos estos aspectos pueden ser desarrollados, aunque cómo mínimo debería obtenerse un modelo de diseño de la base de datos donde se refleje la información que se maneja, algo como lo mostrado en la Figura 8.

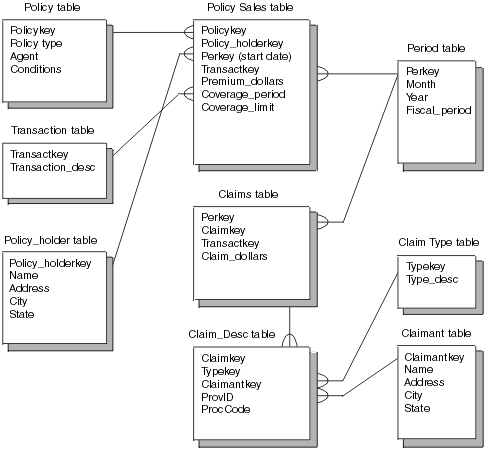


Figura 8. Ejemplo de diseño de BD

(Fuente [https://www.ibm.com](https://www.ibm.com/))

En esta sección se trata de reflexionar sobre las necesidades en el almacenamiento de datos y dejarlas claramente indicadas para que durante la implementación no se deban tomar decisiones sobre la marcha. Un sistema implica una gran cantidad de datos. Por un lado existirán los datos relativos al propio servicio (imagina una aplicación para compartir archivos), pero además de los datos para desarrollar el servicio pueden existir una gran cantidad de metadatos que serán utilizados por administrador para mantener el servicio (por ejemplo un registro de todas las acciones de los usuarios, cuando hicieron login, cuando salieron, geoip desde la que se accede, etc). Todo aquello que se quiera almacenar tendrá unas necesidades, restricciones, condiciones. En esta sección es donde hay que reflexionar y especificar todo esto.

## Diseño arquitectura conceptual

El diseño de la arquitectura conceptual permite definir a grandes rasgos los módulos y bloques funcionales que contendrá la solución. Es una descripción de cómo se plantea desarrollar la solución y condicionará las tecnologías que posteriormente se utilizaran. Este diseño puede hacerse en varias fases donde cada uno de ellos profundiza en un bloque. Podemos observar un ejemplo en los siguientes diagramas extraídos del TFG de Alejandro Torres, Figura 9

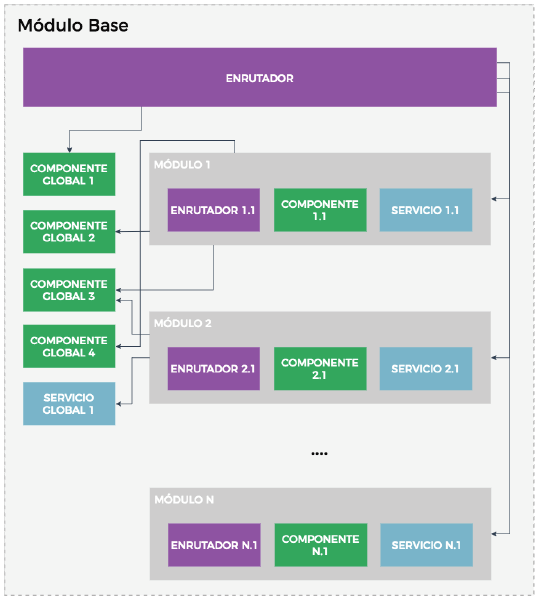
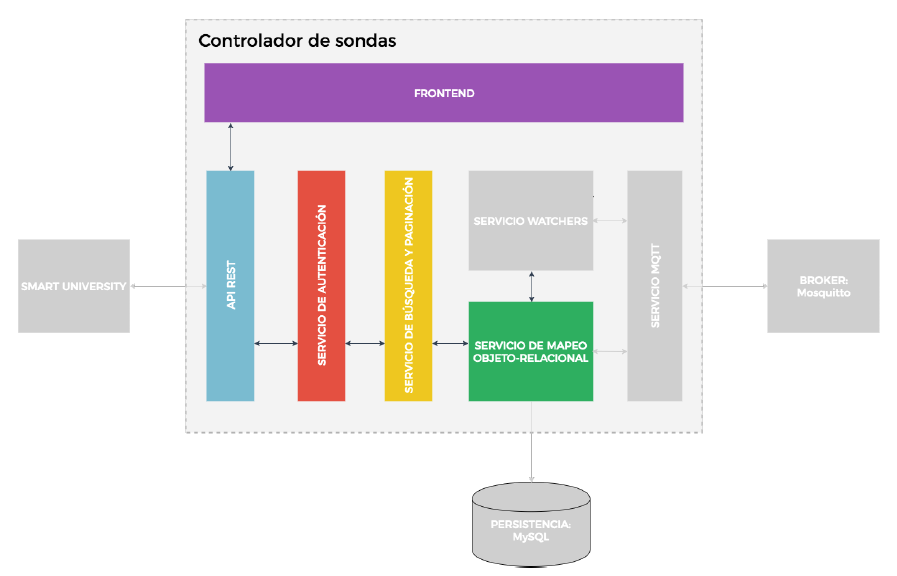


Figura 9. Diseño arquitectura conceptual panel IoT.

(Fuente TFG Alejandro Torres Mateu)

Estos diagramas ayudaran en el desarrollo a estructurar archivos, directorios, módulos, agrupar funcionalidades y distribuir el código. También puede identificar servicios de terceros que serán utilizados (por ejemplo login con una red social). Esta arquitectura puede ser muy sencilla y tal vez no sea necesario este apartado.

## Diseño API Rest (si hay)

Dependiendo del tipo de solución a abordar y siguiendo esquemas conceptuales actuales muy extendidos, es posible que el producto se base en una API Rest. Es necesario diseñar estar API al margen de qué tecnologías serán utilizadas, identificando las entidades que la formarán, las operaciones a realizar, por lo menos las básicas o llamadas CRUD y las rutas. Muchas veces no comprender que es una API Rest y cómo utiliza los métodos HTTP para hacer su trabajo genera una mala definición de esta API que, aunque funcional, conceptualmente es errónea. Es recomendable algún material de referencia como puede ser el libro de Enrique Amodeo [8], con él será muy fácil diseñar buenas API Rest que luego será muy fácil implementar. Esta guía además explica aspectos que pueden comprometer nuestro funcionamiento como la concurrencia o la utilización de recursos multimedia.

Cómo mínimo deberías obtener tras este apartado un listado con el routing de nuestra API Rest, la definición de los parámetros y los datos/formatos que devuelve nuestra API. Algo parecido a lo que por ejemplo nos ofrece la OPEN API Doc de Smart University [9].

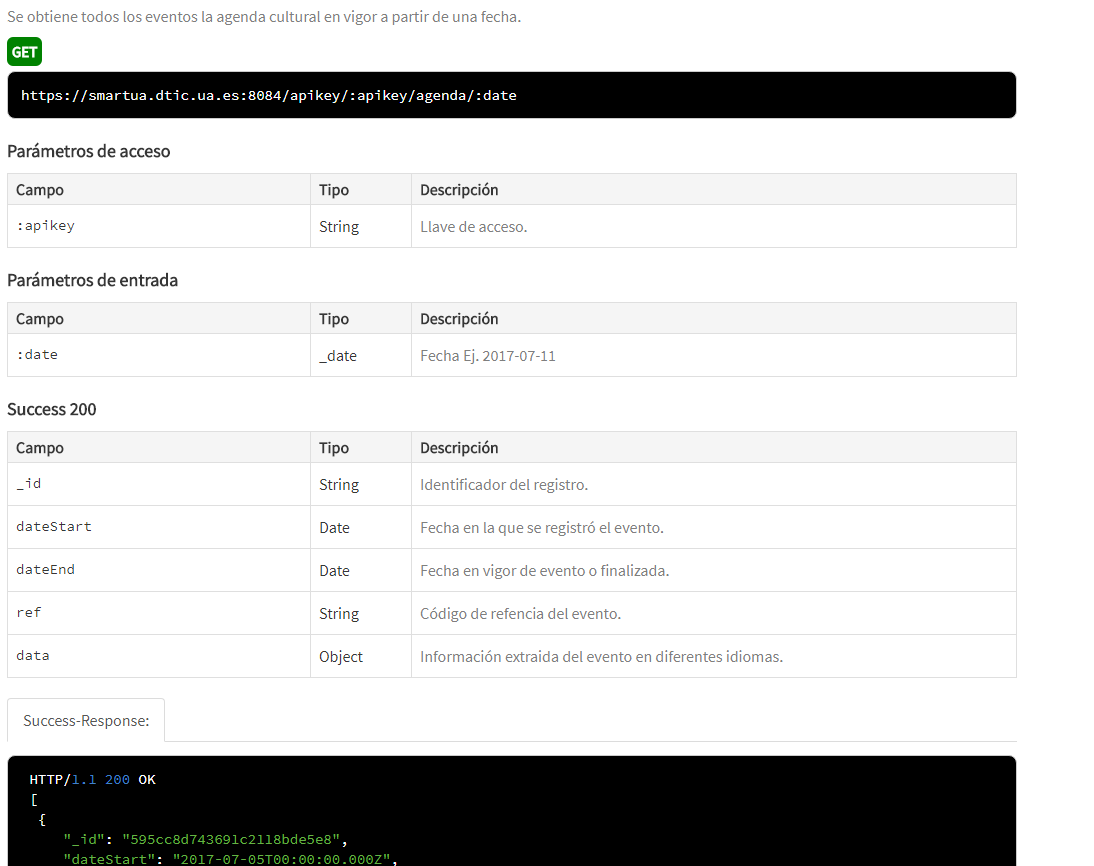


Figura 10. Especificación de la Open API de Smart University

(Fuente Smart University www.smart.ua.es)

## Diseño arquitectura tecnológica Front/Back-end

Este apartado ya incluye una propuesta de las tecnologías que serán utilizadas para el desarrollo del proyecto. Estas tecnologías también deberán escogerse en función de los requerimientos tanto funcionales como no funcionales e incluso del posible análisis que se haya podido realizar durante el estado del arte. Si la solución deberá estar destinada por ejemplo a conformarse como un servicio web, las tecnologías serán web, si debe ser una aplicación móvil, las tecnologías estarán relacionadas con esto. La definición de tecnologías puede llegar incluso a sistemas como: servicios de identificación de terceros, herramientas de copias de seguridad, Sistemas Operativos, sistemas de lanzamiento de tareas en segundo plano, etc. Aquí se especificará e incluso se discutirán alternativas para presentar las decididas finalmente. La Figura 11 muestra por ejemplo el stack tecnológico seleccionado para implementar la arquitectura del proyecto Smart University en función de su arquitectura conceptual de 3 niveles. El alumno decidirá el suyo en función de su propio diseño conceptual.

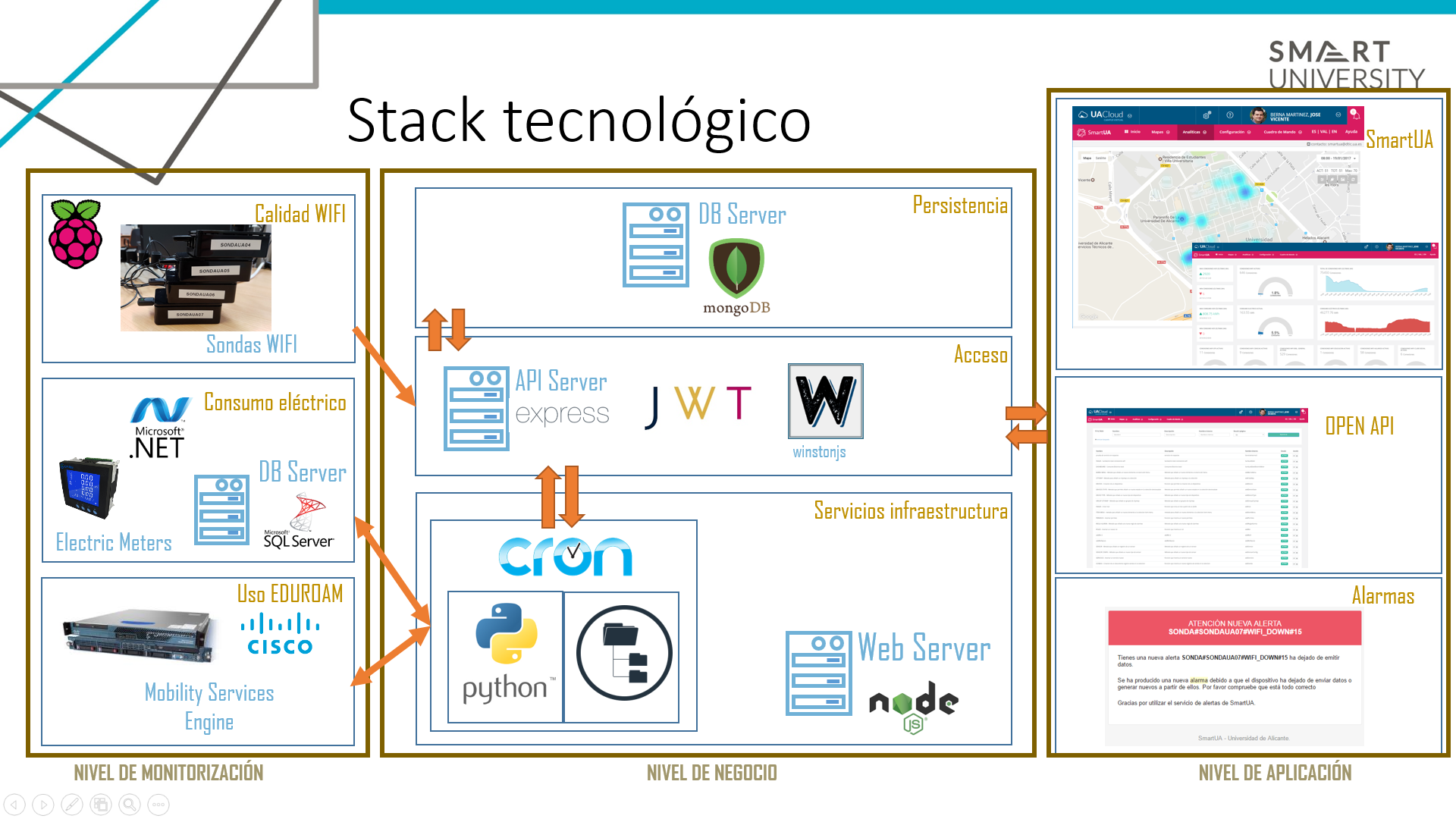


Figura 11. Stack tecnológico del proyecto Smart University

(Fuente Smart University www.smart.ua.es)

## Diseño Interacción o Experiencia de Usuario - UX

La experiencia de usuario es un componente muy importante de las actuales aplicaciones y suele ser además un factor de fracaso o éxito. La experiencia de usuario es un concepto todavía controvertido y polémico que muchas veces se asume desde los tradicionales de accesibilidad y usabilidad pero que en realidad actualmente es mucho más que estos conceptos. Sirva el siguiente ejemplo de la Figura 12 como definición clarificadora de la diferencia entre los conceptos de accesibilidad, usabilidad y experiencia de usuario. La función de una máquina de resonancia magnética es la generación de una imagen que proporciona información sobre la composición y estructura del cuerpo humano sin invadirlo literalmente. Si bien la funcionalidad está clara, para el paciente su experiencia de usuario se resume en que ha de colocarse en el interior de la máquina y verse sometido durante un periodo de tiempo variable a un ruido estridente que lo puede poner nervioso, estresar e incluso inducir al pánico. La evolución de esta máquina en el tiempo ha mejorado en cuestión de la calidad del resultado que se obtiene de la prueba, pero su funcionamiento prácticamente es el mismo. Como podemos ver en la Figura 12-1, las versiones básicas de la máquina pensaban poco en el paciente y solo se centraban en la funcionalidad. El paciente debía subirse a una camilla alta, utilizando escalones o adaptadores, recostarse en la camilla y sufrir el proceso. Además, el profesional que realizaba la prueba no podía estar con él para no interferir durante el proceso electromagnético lo cual le dejaba solo en la habitación y con el peligro de que si durante la prueba se movía, tenían que repetirla por completo. Uno de los pasos fue mejorar la usabilidad de la máquina como vemos en la Figura 12-2, diseñando habitaciones donde el profesional pudiese controlar en todo momento al paciente y a la vez el paciente de alguna forma se pueda ver acompañado y guiado en el proceso. Pero aun así la experiencia de usuario mejora poco, aunque sí la usabilidad de la máquina por parte de los profesionales. Sin embargo, en la Figura 12-3 ya vemos que se introduce un elemento destinado exclusivamente al paciente, la utilización de camillas elevadoras de forma que el paciente ahora puede hacerse la prueba más cómodamente, esto mejora el acceso (accesibilidad) a la realización de la prueba (pero no su funcionalidad), mejora algo la experiencia de usuario, pero en realidad es funcionalidad mejorada para el usuario. Por último, en la Figura 12-4 vemos una modificación que no tiene nada que ver absolutamente con el aspecto funcional, ni del usuario ni del paciente, pero que está completamente destinado a la experiencia de usuario. La habitación y la propia máquina han sido decoradas con distintos motivos de forma que un niño (pacientes habituales también) no tendrá miedo e incluso puede disfrutar de la experiencia (quien sabe si tal vez le queden ganas incluso de repetir).



Figura 12. Ejemplo de la diferencia entre usable, accesible y experiencia de usuario.

(Fuente propia)

La consideración de aspectos destinados a la UX normalmente no interfiere en la usabilidad ni accesibilidad (otras veces sí), pero sí que interfieren en la huella que generan en el usuario y por tanto en su predisposición a volver a nuestro servicio.

Hay muchas técnicas para estudiarla. Uno de los primeros aspectos a comprender es precisamente que es lo que se espera que un usuario haga en nuestro servicio, entender cómo esperamos que interaccione con él y para ello nos puede ayudar los User Journey Map, aquí este artículo [10] os puede ayudar a entender para qué sirven.

También puede ser muy útil saber más sobre UX, sus aspectos claves y ciertas cosas que se pueden tener en cuenta en general y que te ayudaran a que tus usuarios quieran utilizar más tus aplicaciones. En el libro de Yusef Hassan [11] podrás encontrar muchas claves. La idea es tenerlas claras para cuando diseñes tus interfaces.

Es importante dentro del desarrollo UX hacer el diseño del flujo de navegación, es decir, comprender cuales son los pasos que deberá seguir el usuario en nuestra aplicación para lograr llevar a cabo una acción. Este diseño del flujo no presta atención al diseño gráfico, sino a las acciones. Pensar el viaje que el usuario va a tener en nuestra aplicación para que el usuario no tenga puntos muertos. Por ejemplo, cuando un usuario está completando un formulario y pulsa el botón enviar, qué es lo que verá después o si existen errores cómo se le va a mostrar esta información. Estos mapas de navegación pueden ser diagramas como el de la Figura 13, un diagrama en el que se muestra el flujo del usuario entre las distintas interfaces. Este mapa ayuda a visualizar que al usuario no se le somete a excesivos pasos intermedios para alcanzar funcionalidades y que todos los puntos de retorno son correctos.

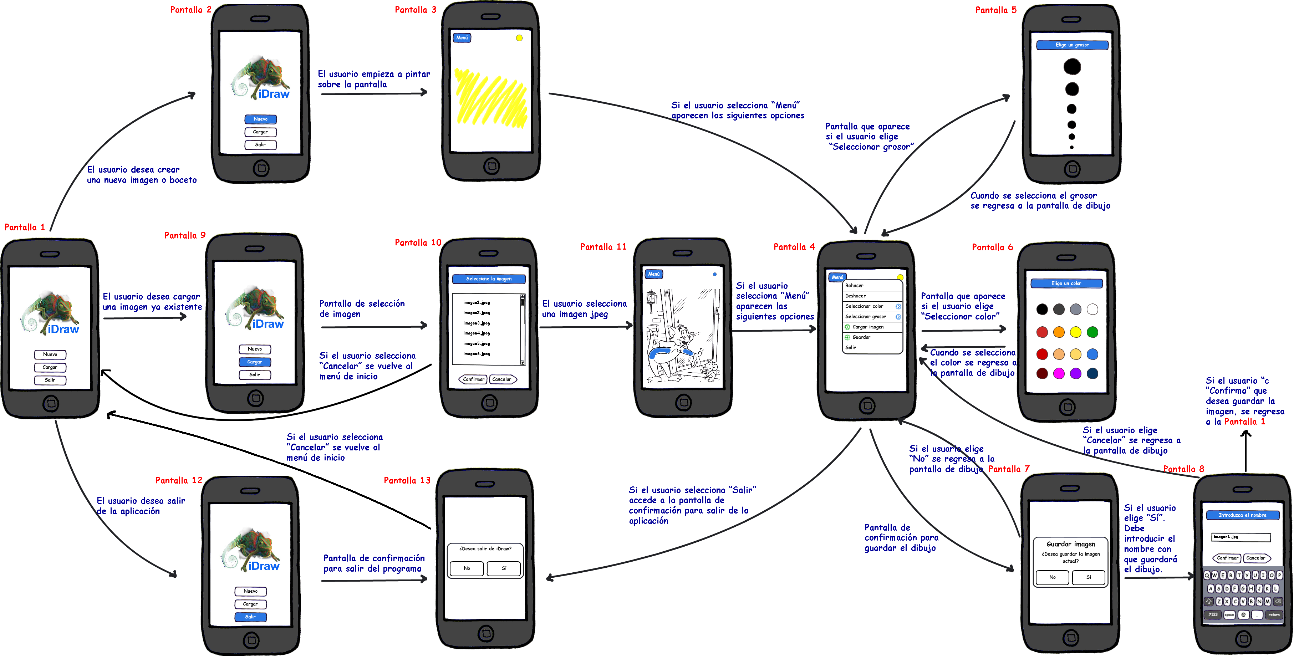


Figura 13. Ejemplo de mapa de interacciones.

(Fuente blog wordpress https://josemigueldih.wordpress.com)

## Diseño Interfaces - UI

El diseño de interfaces es la parte donde los RF y RNF se materializan, es decir, es la parte de la ingeniería donde se desarrollan los mecanismos físicos mediante los cuales los distintos usuarios podrán interaccionar con nuestro sistema. Pueden ser interfaces web, aplicaciones de escritorio, aplicaciones móviles o incluso líneas de comandos. Este diseño debería implementar los requerimientos funcionales y no funcionales que no han sido satisfechos por otras partes del diseño (por ejemplo el cifrado en BD para satisfacer requerimientos de seguridad).

Es importante tener en cuenta las distintas prácticas de usabilidad, accesibilidad y UX tanto estudiadas como vistas en apartados anteriores. La definición de las interfaces permite la toma de decisiones en etapas de diseño y no en desarrollo, lo cual agilizará esta última fase. El diseño de interfaces puede hacerse con distintos grados de fidelidad lo cual facilitará más o menos el desarrollo. Hay que tener en cuento que cuanto mayor exactitud tenga el diseño de interfaces ahora, menos decisiones y desarrollos hay que hacer después. Podemos hablar de tres niveles de fidelidad.

* Wireframe

Es la primera representación de nuestra propuesta de solución, normalmente utiliza cajas y símbolos entendibles y está orientado a reflejar de una forma poco refinada las funcionalidades del sistema y dónde encajan más o menos los elementos que lo forman. La Figura 14 es un ejemplo. Los wireframes son indispensables en las primeras fases del diseño. Aquí se incluyen ya los aspectos de usabilidad y accesibilidad que están más relacionados con las funcionalidades. Tiene como ventaja que es muy rápido su desarrollo ya que no se tienen en cuenta por menores de diseño y está muy orientado a expresar de qué forma se implementen los RF y RNF. El inconveniente es que dista mucho de lo que será el producto final, al cliente le puede inducir a error y hay muchas interacciones que pueden no percibirse y quedar olvidadas. Una herramienta es Balsamiq [12].

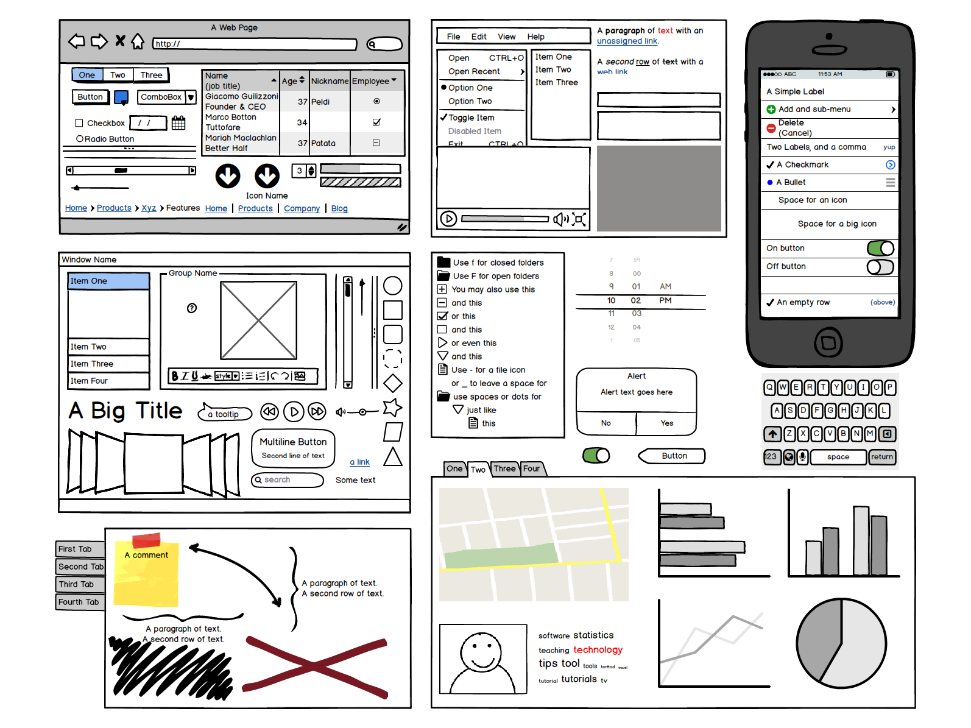


Figura 14. Ejemplo de wireframe con Balsamiq.

(Fuente Balsamiq www.balsamiq.com)

* Mockup

Un mockup es una representación visual y estática de un diseño como muestra la Figura 15. Es una vista exacta de cómo será la interfaz que percibirá el usuario y por tanto indica no solo la posición de os elementos sino su estado final una vez implementados. Además en este diseño ya se contemplan temas relacionados con la UX ya que colores, tipos y fuentes, tamaños, fondos, aspecto general, percepción, etc., ya están definidos. Tiene como principal ventaja que ya es muy cercano al desarrollo y por tanto muchos elementos pueden ser utilizados para agilizarlo, además de que muchas decisiones como la disposición de los elementos en un formulario, el orden de aparición de ciertos pasos, etc., ya están tomadas. El inconveniente es el tiempo necesario para su desarrollo ya que incluye aspectos puramente de diseño que ralentizan por tanto su creación. Se pueden utilizar herramientas cómo Canva [13] o Justinmind [14].

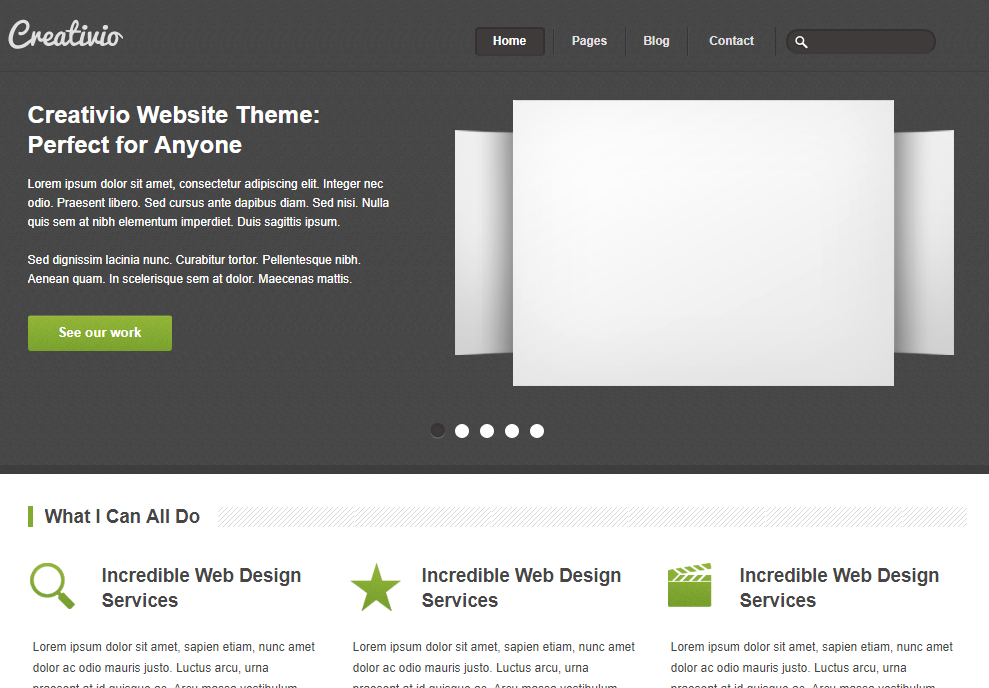


Figura 15. Mockup desarrollado con Justinmind

(Fuente Justinmind www.justinmind.com)

* Prototipo

Un prototipo es también una representación de la versión final de un diseño que, además, simula la interactividad. Un prototipo permite entender mejor la interacción del usuario con la herramienta y es un buen principio para mostrar a un cliente ya que le permitirá a él interactuar con dicho producto sin que se haya sido necesario implementar código. La ventaja es evidente, tener una simulación del producto final que permitirá medir no solo el UX sino también validar la interacción como muestra la Figura 16. La gran desventaja es que requiere de invertir mucho más tiempo en diseño y puede ser incluso prohibitivo en plazos. Es aconsejable solo en los casos en los que se requiere de una aprobación del cliente. Herramientas como Justinmind [14] o Sketch [15].

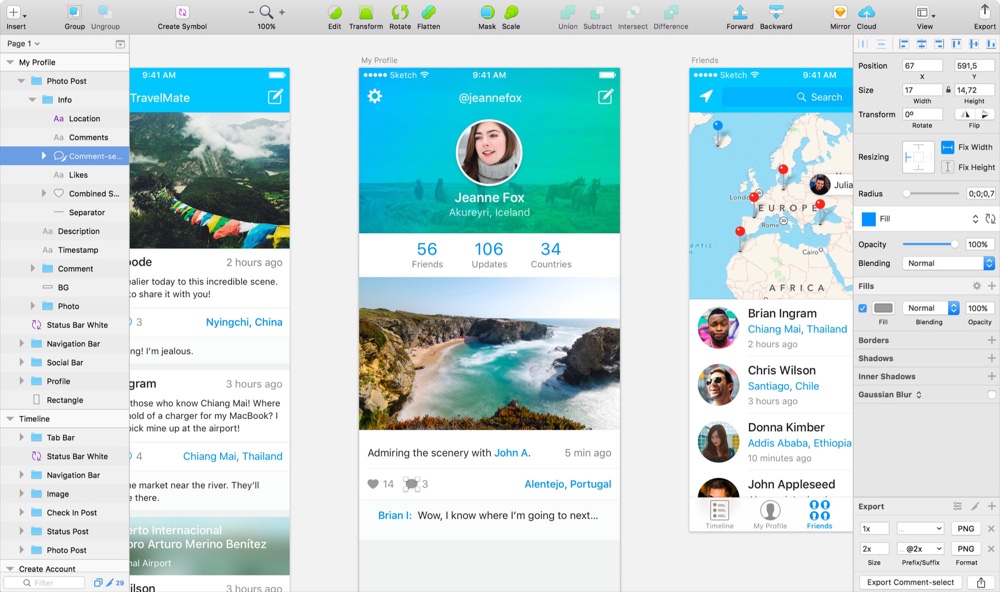


Figura 16. Diseño desarrollado con Sketch.

(Fuente Sketch www.sketchapp.com)

Las herramientas como Sketch o Justinmind producen mapas de navegación donde además se puede comprobar que coincidan con la UX diseñada.

## Guías de estilos

Si tu diseño es de alta fidelidad la guía de estilos estará implícita en el propio diseño. Aun así, es bueno que la aísles y la describas. La guía de estilos define fuentes, tipos, tamaños, colores y ubicaciones en general. No es que exista una única forma de realizar una guía de estilos, pero puede ser muy útil buscar una de una gran corporación y pensar en detallar los mismos elementos. Por ejemplo, te puede servir de la de la Universidad de Cádiz [16] donde se definen desde la ubicación de los elementos, las paletas de colores (Figura 17) y tipografías. Si tienes todos los elementos definidos e incluso plantillas, el desarrollo será muchísimo más rápido y tu aplicación final mucho más coherente. Si esto ya lo has hecho en tu diseño de interfaces, asegúrate de que es coherente y constante.

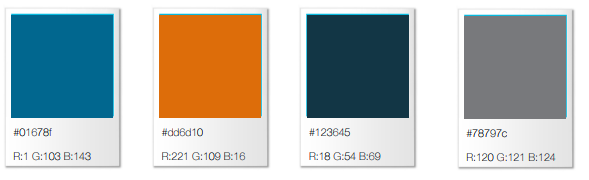


Figura 17. Paleta de colores Unviersidad de Cádiz

(Fuente Guía de Estilos Universidad de Cádiz)

## Diseño de pruebas y validación

También es interesante que definas otros elementos relacionados con la comprobación de tu sistema a diversos niveles. Tipos de pruebas y validaciones hay decenas y podría escribirse un TFG solo con el diseño de estas, pero dado que en el caso del desarrollo de un TFG el objetivo principal no es el diseño de pruebas, debemos definir al menos algo que nos permita concluir que hemos alcanzado los objetivos.

Por un lado, es muy recomendable, ya que vas a implementar posiblemente un API o servicios de acceso a datos, que definas y automatices alguna prueba que te permita comprobar que tras un cambio estos servicios siguen funcionando. Esto puede ser tan complejo o simple como quieras pero, por ejemplo, en el caso de una API puedes utilizar herramientas como Postman [17]. Cada vez que desarrolles una llamada nueva de tu API puedes definirte una llamada que genere una respuesta correcta y una que genere una respuesta incorrecta, con Postman puedes crear baterías de pruebas que vas incrementado junto a tu desarrollo. Una vez que realizas una modificación puedes fácilmente ejecutar una secuencia de llamadas y analizar que todas han devuelto el resultado esperado. Puedes ver un ejemplo en la Figura 18.

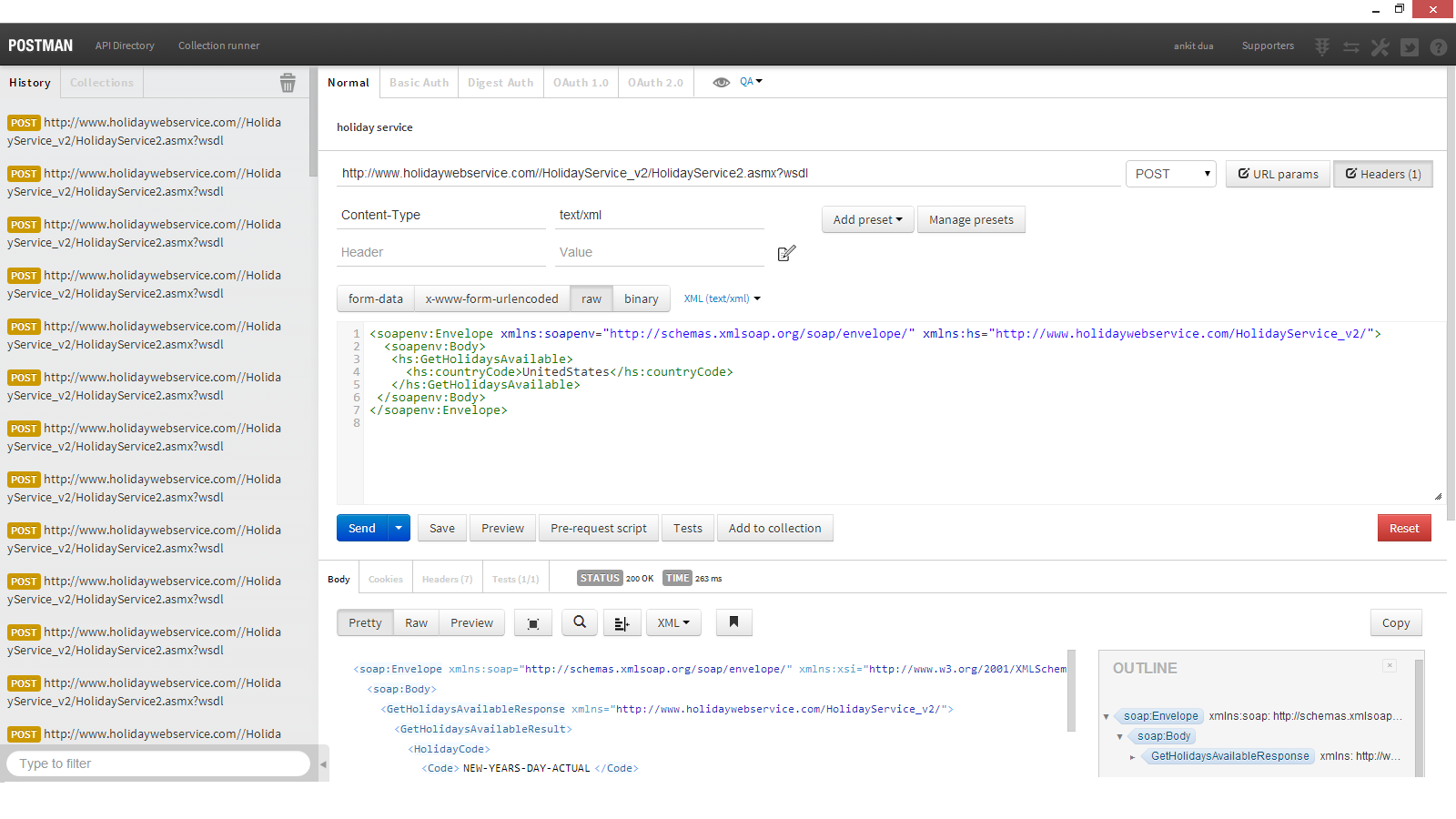


Figura 18. Ejemplo de ejecución de varias invocaciones a servicio.

(Fuente Postman http://blog.getpostman.com)

También es adecuado definir indicadores y los valores esperados para estos indicadores. Por ejemplo, si tienes un RNF sobre la capacidad de tu servicio y lo has diseñado pensando en que el número de conexiones va a estar en torno a 100 instancias por segundo, puedes definir un indicador *NumConextiones*, y un rango: 0-50 bajo, 50-80 medio, 80-100 bueno, 100< muy bueno. Cuando hagas pruebas de estrés de conexiones en función del valor obtenido sabrás si has alcanzado con éxito tu RNF.

Otra buena práctica es crearse una lista de comprobación con los RF y RNF y asegurarse de que han sido implementados según su definición ya que muchas veces y debido a que las funcionalidades están dispersas a lo largo de diseños de distinto tipo, pueden perderse o quedar definidos a medias.

Para validar la solución, es decir, comprobar que realmente lo implementado es lo que se buscaba solucionar, puedes definir varios casos de uso, incluyendo la definición de los datos o acciones que el usuario debe realizar junto a los resultados esperados. Estos caso se convertirán en listas de comprobación cuando hayas terminado tu desarrollo, y deberás asegurarte de que se pueden llevar a cabo y producen los resultados esperados realmente.

## Otros apartados

Puedes considerar añadir otros apartados que puedan ser de tu interés durante el desarrollo, por ejemplo, si inspeccionas alguna API de terceros y te interesa tener documentadas algunas llamadas o incluso hacer algunas pruebas. Recuerda que el apartado de diseño dependerá del tipo de proyecto que estés realizando.

Si vas a desarrollar código también puedes considerar la aplicación de buenas prácticas, para lo cual puedes utilizar alguna guía de buenas prácticas que documentes en un apartado, o bien establecer tus propias reglas para ayudarte a mantener un código limpio, estructurado y ordenado.

# Implementación

En este apartado se documenta el proceso de desarrollo pero no se escribe el código. Es muy importante tener en cuenta que debe adecuarse a la metodología propuesta y exhibir evidencias de dicha metodología. Si se indicó por ejemplo una metodología iterativa, deberían indicarla las iteraciones realizadas y las funcionalidades que se incluyeron en cada una. Si la metodología es Scrum podrían mostrarse los sprints.

También se puede documentar aspectos como la preparación de los entornos en caso de que se hayan preparado entornos de desarrollo, integración, preproducción y producción, herramientas para el desarrollo y soporte (como por ejemplo gestión de versiones con Git).

Se indica la estructura del proyecto, la distribución de archivos, archivos de configuración a nivel de sistemas (por ejemplo, si es necesario modificar la configuración de nuestro servidor web Apache) y de aplicación (si nuestro aplicación por ejemplo tiene un idioma por defecto y se configura en alguna variable).

Se analizan los problemas ocurridos durante el desarrollo, las decisiones tomadas, los ajustes necesarios y su justificación (por ejemplo, descarte de funcionalidades por razones temporales).

Si se han desarrollado algún algoritmo o solución muy a destacar se puede mostrar el código comentado, pero se debe ser muy crítico con esto y es preferible mostrar pseudocódigo o algoritmos en vez de código puro. Se puede utilizar la sección de Anexos añadida al final para mostrar código, documentos de seguimiento, documentos manuales que han sido utilizados u otros elementos de interés.

Exhibir el producto final, sus interfaces destacables para comprobar que el resultado es similar al diseñado.

Habitualmente este apartado consistirá en varios subapartados, uno para cada esprint o fase del desarrollo. Se debería indicar claramente al inicio de cada subapartado el objetivo concreto de esa sección y luego dentro desarrollar los aspectos necesarios. Se suele mostrar cómo se hace una cierta acción por primera vez, por ejemplo, como se crea un componente en Angular, la técnica utilizada para el texturizado, el filtro aplicado en el vídeo. De esta forma cuando se vuelva a utilizar esto mismo simplemente se dice que se ha hecho igual que las veces anteriores. La idea es no pegar todo el código, pero sí ir dejando huellas de tu código o tu trabajo. Tal vez se aproxima a un “cuaderno de bitácora”. Ten en cuenta que aquí debería mostrarse como avanza el desarrollo. Por ejemplo, es habitual desarrollar primero las funcionalidades y en una fase final realizar la maquetación, por lo que se verían en las primeras capturas de pantalla las interfaces planas, sin colores y sin estilos gráficos, y en la fase de maquetación se verían como quedan definitivamente.

Es muy importante que desarrolles este capítulo al mismo tiempo que desarrollas porque si no, no podrás mostrar los pasos intermedios. Comienza siempre por una primera fase de preparación de entornos de trabajo, donde documentes perfectamente como has preparado tu entorno de trabajo, es decir, qué herramientas has instalado, que configuraciones has seguido, que versiones del software, si has utilizado un repositorio cómo lo has enlazado, etc.

Por último, muy importante, si has generado código, no olvides dejar un enlace claro y directo al repositorio donde el tribunal podrá ver tu proyecto. Y por supuesto, que este código quede en una rama final principal, o bien la *master* o bien una *release*. Deja un código limpio, comentado, coherente y estructurado. Y por supuesto, que sea público y accesible.

# Pruebas y validación

Una vez la proyecto ha sido finalizada deben incluirse las pruebas y validaciones diseñadas en el apartado anterior. La idea es demostrar que no solo se ha generado la herramienta esperada, sino que es fiable y robusta. Validar finalmente que resuelve el problema analizado y propuesto de una forma aceptable.

Es posible que muchas pruebas unitarias se hayan desarrollado durante la codificación, pero si por ejemplo se posee una batería de llamadas a la API en Postman, puede ser un buen momento comprobar que realmente todas pasan la ejecución.

Una de las pruebas que son muy deseables son las pruebas de campo con usuarios reales. Cada vez es más habitual que cuando termina el desarrollo, la herramienta, el videojuego, el corto de animación o la creación que se haya llevado a cabo, sea sometida al juicio de un conjunto de usuarios. Este proceso no es baladí, pues debe seleccionarse al conjunto de usuarios de prueba adecuado (si es un videojuego por ejemplo debe seleccionarse a personas que suelan jugar). Después diseñar qué es lo que queremos que el usuario haga con nuestro producto. Y finalmente facilitarles algún tipo de encuesta o test que recoja los resultados y nos permita analizar si el producto cumple con las expectativas.

Si se han definido indicadores es un buen momento para demostrar que las métricas son adecuadas, por ejemplo comprobar que el rendimiento cumple con unos mínimos o que algunas cifras son alcanzadas.

Si hay problemas en las pruebas se deben documentar y analizar a que son debidas proponiendo las posibles soluciones, estas pueden ser propuestas como trabajo futuro.

# Resultados

Analizar el producto final y mostrar los resultados. Si está en producción o ha sido probada por usuarios tester o reales indicar las valoraciones, el número de usuarios, si tiene contenidos, si está generando monetización o los indicadores de tráfico, que comentarios o valoraciones ha recibido nuestros resultados.

En este punto se trata de analizar cómo lo que se ha hecho ha mejorado el problema. Es decir, las interfaces y producto se han mostrado durante el diseño, aunque en esta sección se puede hacer un resumen de este. Pero estaría muy bien por ejemplo si hemos hecho una herramienta web, mostrar cómo de eficiente es el proceso que resuelve ahora. Si se trata de un corto con algún objetivo, pues analizar que impacto ha tenido, si hay noticias que hablan de él, etc. Si por ejemplo es un videojuego y se ha presentado en alguna gamejam, pues hablar de si ha ganado algún premio o a obtenido alguna mención. Si es una aplicación en producción y tiene por ejemplo algún plan de pagos o monetiza a través de publicidad, sería muy adecuado hablar de si ya ha generado ingresos.

Realizar también un análisis interno sobre los costes temporales planificados y los reales, analizar las diferencias entre ellos y justificarlas. Analizar también otros ajustes como los funcionales, si se deben a un exceso de funcionalidades, a otros problemas como posibles errores que producen retrasos o problemas con el hardware. Si has tomado tiempos a través de herramientas como Toggl se pueden mostrar gráficas y resúmenes de los tiempos generados.

También se puede hacer un recorrido por algunas asignaturas de la carrera enlazando los contenidos vistos con los desarrollados en el trabajo. De esta forma se pone en valor la puesta en práctica de las competencias desarrolladas durante la carrera. Es una buena reflexión que puede ayudar a hacer un análisis final sobre aquellas asignaturas más significativas para nuestro proyecto.

Esta sección no es para repetir cosas y dichas antes durante el desarrollo, sino para analizar qué resultado a producido al final nuestro trabajo. Recuerda que el TFG tiene una motivación, un problema de trasfondo y objetivo. ¿De verdad se ha logrado algo sobre este objetivo inicial?

# Conclusiones y trabajo futuro

Apartado de carácter obligatorio que todo trabajo debe contener. En este apartado se recogerán las conclusiones extraídas de nuestro trabajo. Cosas como recalcar el alcance de los objetivos, si se han logrado por completo o no, sobre los resultados obtenidos que repercusiones pueden tener en el entorno o comunidad destino, sobre la dificultad global del proyecto y tus propias impresiones personales.

Además, todo trabajo siempre está inacabado y por ello debemos finalizar las conclusiones con unas líneas de trabajo futuro indicando por ejemplo mejoras que deseamos realizar en posteriores versiones, una línea de trabajo para su puesta en producción, funcionalidades que no han podido ser implementadas e incluso si hay un plan de comercialización a la vista. También se pueden analizar costes futuros para un despliegue o necesidades de desarrollo más allá del alcance del TFG.

Sirva como ejemplo las propias conclusiones de la generación de este documento:

Tras varios años de experiencia donde se aprecia que los estudiantes tienen un gran problema en la estructuración de su TFG, tanto a nivel de contenidos como en planificación, y también en proporcionar un formato adecuado, se ha decidido solucionar el problema generando una guía que sirva tanto de plantilla y modelo de documento como de ejemplo y descripción de los posibles contenidos. Esta guía no es exhaustiva ni universal, pues está orientada a proyectos de tipo desarrollo software y pincela algunos otros tipos de proyectos, de ahí sus secciones tan orientadas a este proceso. Pero sin embargo puede servir como base a cualquier proyecto y TFG/TFM.

Se desprende de esta guía que el trabajo del ingeniero es precisamente el arte y técnica de aplicar los conocimientos científicos a la invención, diseño, perfeccionamiento y manejo de nuevos procedimientos en la industria y otros campos de aplicación científicos. En resumen, la creación e innovación. Por ello ha de poner su empeño no tanto en el desarrollo de la solución final sino en el proceso que sigue a lo largo de todo el proyecto y que le lleva hacia el producto final.

Este es el resultado de varios años de trabajo en la dirección de TFG/TFM y después de haber elaborado diversos recursos, talleres y seminarios, y creo que será un buen principio donde los alumnos podrán superar fácilmente la primera dificultad, el del documento vacío, y ponerse a trabajar siguiendo indicaciones claras y una hoja de ruta en forma de plantilla.

Como trabajo futuro inmediato queda por supuesto evaluar formal o informalmente el resultado de esta guía, recogiendo impresiones de mis propios alumnos de TFG/TFM junto con sus tasas de éxito, la calidad de los trabajos tras la evaluación del tribunal e incluso el impacto en otros TFG/TFM.

Como trabajo a largo plazo queda la mejora continua de dicha guía a partir de los comentarios de alumnos, profesores, compañeros y tribunales junto con los propios resultados de la evaluación.

# Referencias

La sección de referencias debe incluir las obras y materiales consultados y empleados en la elaboración de la memoria. Las referencias pueden indexarse en una lista numerada siguiendo el orden de aparición con lo que serán referenciadas utilizando el número de la entrada entre corchetes, o en orden alfabético con lo que serán referenciadas utilizando nombre y fecha. Para más detalles se puede consultar la Guía de Estilos de la EPS [3]. La lista siguiente compone la lista de referencias utilizadas en las explicaciones de la guía. El número de referencias totales de un TFG no está fijado, pero es un indicativo de la investigación previa realizada. El formato de las referencias será APA (suele ser el más habitual) o Harvard. El formato APA sigue una convención en el mundo académico ampliamente utilizada. Ya deberías haber utilizado este formato de citas pero si no es el caso ten en cuenta que la forma de tus competencias. Puedes encontrar una guía completa de como utilizar el formato APA en [18] y en [19] tienes una herramienta que te permite generar las citas a referencias bibliográficas de forma sencilla. Cuando se tratan de libros o artículos, también puedes utilizar Google Scholar para buscarlos y utilizar la opción “citar” que aparece bajo cada enlace para obtener la cita APA.

1. Aramendia, G. Z. (2020). Fundamentos de marketing. Editorial Elearning, SL.
2. Molina, M. M. (2015). Cómo triunfar en las redes sociales: Consejos prácticos y técnicas para conseguir todo lo que te propongas en Internet y sacarle más partido a tus redes sociales: Facebook, Twitter, Instagram, LinkedIn. Grupo Planeta Spain.
3. UA. (2014) Libro de Estilos para la redacción de trabajos TFG/TFM de la EPS. Disponible en: <https://maktub.eps.ua.es/servicios/gestorContenidos/contenidos/normativaEPS/Pdf/9910.pdf>
4. Doran, G. T. (1981). There’sa SMART way to write management’s goals and objectives. Management review, 70(11), 35-36.
5. *Especificación de Requisitos según el estándar de IEEE 830*. Disponible en: <https://www.fdi.ucm.es/profesor/gmendez/docs/is0809/ieee830.pdf>
6. *Bases de Datos*. Wikipedia. Disponible en: <https://es.wikipedia.org/wiki/Base_de_datos>
7. Torres Mateu, A. (2017). Panel frontal para el controlador IoT de Smart University. Disponible en: <http://hdl.handle.net/10045/69787>
8. Amodeo, E. (2013). Principios de Diseño de API's REST. Canada: Leanpub.
9. Documentación de la API de Smart University de la Universidad de Alicante. Disponible en: <https://smartua.dtic.ua.es:80/public/apidoc/>
10. *El Mapa de Experiencia del Cliente o Customer Journey Map*. Disponible en: <http://innokabi.com/claves-para-emocionar-a-tu-cliente-customer-journey-map/>
11. Montero, Y. H. (2015). Experiencia de usuario: principios y métodos. Experiencia de usuario: principios y métodos, 9. Disponible en: <http://yusef.es/Experiencia_de_Usuario.pdf>
12. Balsamiq. Herramienta para crear wireframes. Disponible en: <https://balsamiq.com/>
13. Canva. Herramienta para crear mockups. Disponible en [https://www.canva.com](https://www.canva.com/)
14. Justinmind. Herramienta para crear mockups y diseños interactivos. Disponible en <https://www.justinmind.com/>
15. Sketch. Herramiento para crear mockups y diseños interactivos para Mac. <https://www.sketchapp.com/>
16. Guía de Estilo Web de la Universidad de Cádiz. Disponible en: <http://www.uca.es/recursos/doc/Unidades/Gab_Com_Mark/465200059_19420109123.pdf>
17. Postman. Herramienta para análisis de API REST. En forma de plugin para Chrome y aplicación. Disponible en: <https://www.getpostman.com/>
18. Universidad de Alicante. (2013). Estilo APA. Biblioteca Universitaria. <https://web.ua.es/es/eurle/documentos/trabajo-de-fin-de-grado/estilo-apa.pdf>
19. Generador APA. (2022, 23 mayo). Formato APA con el Generador APA de Scribbr. <https://www.scribbr.es/detector-de-plagio/generador-apa/>

# Apéndice I

Se pueden añadir tantos apéndices como sean necesarios. Los apéndices recogen todo aquello que no es recomendable que esté en mitad del texto como por ejemplo código sobre el que se habla específicamente pero que es demasiado largo como para insertarlo en mitad de la memoria, documentos escaneados de originales y que han servido para extraer el análisis de datos, bocetos realizados a mano y que luego fueron digitalizados en las herramientas de diseño. Todo esto si se quiere dejar constancia puede ser insertado en estos apéndices.

Hay que tener en cuenta que todo apéndice tiene que estar referenciado siempre en la memoria, es decir, que en algún momento en el texto tiene que hacerse mención del apéndice y su contenido.

Además, el apéndice debe contener no solo imágenes sino también el texto suficiente para explicar su contenido pero sin repetir texto de la memoria. Por ejemplo, supongamos que se ha hecho un primer wireframe a papel que después ha sido digitalizado. En la sección de diseño correspondiente indicaríamos que:

*“… en el Apéndice I hemos dejado el primero boceto manual que aquí está digitalizado y que incluye los elementos a, b, c…”*

Mientras que en el Apéndice I indicaríamos lo siguiente, junto a la imagen bocetada:

*“A continuación se muestran los primeros bocetos hechos a mano alzada y que dieron paso a los posteriores diseños de la aplicación. Se incluyen aquí, Figura 19, para que pueda apreciarse como las primeras ideas son perfeccionadas posteriormente en los procesos digitales”.*



Figura 19. Primer boceto de la aplicación.

(Fuente SmartBlog [http://www.smartblog.es](http://www.smartblog.es/))

Los apéndices no son secciones de relleno donde volcar contenidos con el objetivo de alargar la memoria. Es una sección complementaria de contenidos de cualquier tipo que deben ser valiosos para el trabajo.