

Guía para el desarrollo de TFG

Grado en Ingeniería Multimedia



Trabajo Fin de Grado

Autor:

Pablo García Carbonell

Tutor/es:

Elena Lloret

agosto 2022



# Resumen

Incluir en la memoria un resumen de máximo 500 p0alabras. Este resumen será el mismo que se enviará junto a la documentación en la entrega del TFG. Este resumen normalmente se creará una vez finalizado el trabajo y con la visión del trabajo completo. El resumen debe describir de forma concreta y directa el trabajo realizado durante el proyecto, por eso debe ser redactado una vez que se ha finalizado el proyecto.

También es recomendable que sea traducido a las tres lenguas oficiales de la Universidad de Alicante: español, inglés y valenciano.

# Motivación, justificación y objetivo general

Viajar es una de las mayores aficiones a lo largo del mundo, a todo el mundo le gusta visitar y conocer nuevos destinos y culturas. Sin embargo, antes de realizar cualquier viaje es necesario investigar sobre el lugar que vamos a visitar, viendo cuáles son las zonas de interés de la ciudad para cada usuario y eso no siempre es divertido. Perdemos mucho tiempo viendo qué puede ser lo más interesante para cada uno, leyendo comentarios y opiniones en páginas como *Tripadvisor[[1]](#footnote-1)* o *Trivago[[2]](#footnote-2)*.

Por ello, comencé a investigar un poco sobre el tema de *apps* y *webs* turísticas, llegando a la conclusión de que existen dos opciones a la hora de realizar el estudio previo al viaje: por una parte, contratar una agencia de viajes y que ellos se encarguen de organizarlo por ti; y por otra parte buscar tú por tu propia cuenta esas zonas que te puedan llamar la atención del lugar que vas a visitar. Cada opción tiene sus ventajas e inconvenientes: por ejemplo, al contratar una agencia no pierdes tiempo buscando e investigando, pero tienes que pagar a esta para que haga el trabajo por ti; y la otra forma es gratuita, pero pierdes tiempo indagando sobre el viaje que vas a hacer.

Esta decisión que tiene que hacer el usuario antes de viajar para planificar fue la principal razón que me motivó a elegir este trabajo de final de grado (TFG), queriendo realizar una combinación que recoja las ventajas de cada opción y elimine los inconvenientes, es decir, realizar un servicio gratuito que realice el trabajo de búsqueda por ti. Además, a nivel personal, viajar es uno de mis principales *hobbies*.

Por otra parte, el afán por aprender nuevas tecnologías es otro añadido por el que escogí realizar este trabajo de final de grado. En el caso concreto del elegido, deberé utilizar técnicas de procesamiento de lenguaje natural (PLN), que se suele implementar utilizando el lenguaje de programación Python, el cual no he estudiado en profundidad a lo largo del grado, y por tanto me va a permitir adquirir nuevos conocimientos y destrezas con este lenguaje.

Por tanto, utilizando técnicas de procesamiento de lenguaje natural, el objetivo de este TFG es proponer y desarrollar una aplicación web gratuita que ejerza una búsqueda de lugares de interés adaptado a las características del usuario (edad, objetivo por el que viaja), reduciendo así de manera considerable el tiempo que pasaría buscando e investigando sobre la ciudad destino.

# Agradecimientos

Esta sección se utiliza para escribir unos agradecimientos breves a todas aquellas personas, organizaciones, empresas, etc., que han contribuido a que el alumno o alumna haya alcanzado el final de su grado o que han colaborado en su trabajo: padres, familiares, pareja, tutor, amigos... Es voluntaria y completamente personal.

Este apartado no suele contener 1 página.

# Citas

Incluir una o varias citas que demuestren el carácter de tu trabajo, de tu personalidad y actitud en la vida.

*Cuando tenía 17 años, leí una cita que decía algo como: “Si vives cada día como si fuera el último, algún día sin duda estarás en lo cierto”. Causó una honda impresión en mí, y desde entonces, por los últimos 33 años, me he mirado en el espejo cada mañana y me pregunto: “Si hoy fuera el último día de mi vida, ¿querría hacer lo que voy a hacer hoy?” Y si la respuesta es no por demasiados días seguidos, sé que necesito cambiar algo.*

*Steve Jobs*

*Si algo se vuelve demasiado complicado, se atasca o no te convence: reinicia.*

*José Vicente Berná*

# Dedicatoria

También es posible añadir una sección dedicatoria, donde se deja constancia de a quién va dedicado el trabajo. Suele ser una simple frase o texto como el siguiente. El estilo es libre.

*A mi padre*

# Índice de contenidos

[Resumen 1](#_Toc132276994)

[Motivación, justificación y objetivo general 2](#_Toc132276995)

[Agradecimientos 4](#_Toc132276996)

[Citas 5](#_Toc132276997)

[Dedicatoria 6](#_Toc132276998)

[Índice de contenidos 7](#_Toc132276999)

[Índice de figuras 9](#_Toc132277000)

[Índice de tablas 10](#_Toc132277001)

[1. Introducción 11](#_Toc132277002)

[2. Estudio de viabilidad 13](#_Toc132277003)

[2.1. Análisis DAFO 13](#_Toc132277004)

[2.2. Análisis de riesgos 15](#_Toc132277005)

[3. Planificación 18](#_Toc132277006)

[4. Estado del arte. 19](#_Toc132277007)

[4.1 Procesamiento del Lenguaje Natural (PLN) 19](#_Toc132277008)

[4.1.1 Modelos para el PLN 19](#_Toc132277009)

[4.1.2 Niveles/Componentes del PLN 20](#_Toc132277010)

[4.1.3 Aplicaciones del PLN 21](#_Toc132277011)

[4.2 Estudio de la competencia 21](#_Toc132277012)

[4.2.1. Tripadvisor 21](#_Toc132277013)

[4.2.2 Triposo 23](#_Toc132277014)

[4.2.3 FourSquare 23](#_Toc132277015)

[4.2.4 Agencias de viaje 25](#_Toc132277016)

[4.2.5 ChatGPT 25](#_Toc132277017)

[4.2.6 Comparación de la competencia 28](#_Toc132277018)

[5. Objetivos 29](#_Toc132277019)

[6. Metodología 31](#_Toc132277020)

[7. Análisis y especificación 33](#_Toc132277021)

[8. Diseño 35](#_Toc132277022)

[8.1. Diseño de la persistencia 35](#_Toc132277023)

[8.2. Diseño arquitectura conceptual 36](#_Toc132277024)

[8.3. Diseño API Rest (si hay) 37](#_Toc132277025)

[8.4. Diseño arquitectura tecnológica Front/Back-end 38](#_Toc132277026)

[8.5. Diseño Interacción o Experiencia de Usuario - UX 39](#_Toc132277027)

[8.6. Diseño Interfaces - UI 42](#_Toc132277028)

[8.7. Guías de estilos 45](#_Toc132277029)

[8.8. Diseño de pruebas y validación 46](#_Toc132277030)

[8.9. Otros apartados 48](#_Toc132277031)

[9. Implementación 49](#_Toc132277032)

[10. Pruebas y validación 51](#_Toc132277033)

[11. Resultados 52](#_Toc132277034)

[12. Conclusiones y trabajo futuro 53](#_Toc132277035)

[Referencias 55](#_Toc132277036)

[Apéndice I 56](#_Toc132277037)

# Índice de figuras

[Figura 1. Ingreso por año en España por el turismo. 11](file:///C:\Users\pablo\Desktop\TFG\Memoria%20TFG%20Pablo%20García%20Carbonell%20.docx#_Toc132477802)

[Figura 2. Esquema de un análisis DAFO 13](file:///C:\Users\pablo\Desktop\TFG\Memoria%20TFG%20Pablo%20García%20Carbonell%20.docx#_Toc132477803)

[Figura 3. Página de inicio de Tripadvisor. 22](file:///C:\Users\pablo\Desktop\TFG\Memoria%20TFG%20Pablo%20García%20Carbonell%20.docx#_Toc132477804)

[Figura 4. Resultado de búsqueda de Madrid en Tripadvisor 22](file:///C:\Users\pablo\Desktop\TFG\Memoria%20TFG%20Pablo%20García%20Carbonell%20.docx#_Toc132477805)

[Figura 5. Página principal de FourSquare 24](file:///C:\Users\pablo\Desktop\TFG\Memoria%20TFG%20Pablo%20García%20Carbonell%20.docx#_Toc132477806)

[Figura 6. Resultado búsqueda FourSquare 24](file:///C:\Users\pablo\Desktop\TFG\Memoria%20TFG%20Pablo%20García%20Carbonell%20.docx#_Toc132477807)

[Figura 7 Respuesta “lugares de interés para pareja de 50 años en Madrid” por el ChatGPT. 26](file:///C:\Users\pablo\Desktop\TFG\Memoria%20TFG%20Pablo%20García%20Carbonell%20.docx#_Toc132477808)

[Figura 8. Respuesta “lugares de interés para jóvenes de 20 años en Madrid” por el ChatGPT. 26](file:///C:\Users\pablo\Desktop\TFG\Memoria%20TFG%20Pablo%20García%20Carbonell%20.docx#_Toc132477809)

[Figura 9. Resultado a comida para jóvenes de 20 años por ChatGPT. 27](file:///C:\Users\pablo\Desktop\TFG\Memoria%20TFG%20Pablo%20García%20Carbonell%20.docx#_Toc132477810)

[Figura 10. Metodología ágil Scrum. 31](file:///C:\Users\pablo\Desktop\TFG\Memoria%20TFG%20Pablo%20García%20Carbonell%20.docx#_Toc132477811)

[Figura 11. Lienzo Kanban. 32](file:///C:\Users\pablo\Desktop\TFG\Memoria%20TFG%20Pablo%20García%20Carbonell%20.docx#_Toc132477812)

[Figura 12. Kanban del proyecto desde Trello. 33](file:///C:\Users\pablo\Desktop\TFG\Memoria%20TFG%20Pablo%20García%20Carbonell%20.docx#_Toc132477813)

[Figura 13. Diseño arquitectura conceptual panel IoT. 38](#_Toc132477814)

[Figura 14. Especificación de la Open API de Smart University 39](#_Toc132477815)

[Figura 15. Stack tecnológico del proyecto Smart University 40](#_Toc132477816)

[Figura 16. Ejemplo de la diferencia entre usable, accesible y experiencia de usuario. 41](#_Toc132477817)

[Figura 17. Ejemplo de mapa de interacciones. 42](#_Toc132477818)

[Figura 18. Ejemplo de wireframe con Balsamiq. 44](#_Toc132477819)

[Figura 19. Mockup desarrollado con Justinmind 45](#_Toc132477820)

[Figura 20. Diseño desarrollado con Sketch. 46](#_Toc132477821)

[Figura 21. Paleta de colores Unviersidad de Cádiz 47](#_Toc132477822)

[Figura 22. Ejemplo de ejecución de varias invocaciones a servicio. 48](#_Toc132477823)

[Figura 23. Primer boceto de la aplicación. 57](#_Toc132477824)

# Índice de tablas

[Tabla 1. Análisis DAFO 14](file:///C:\Users\pablo\Desktop\TFG\Memoria%20TFG%20Pablo%20García%20Carbonell%20.docx#_Toc132277058)

[*Tabla 2. Análisis de riesgos* 16](file:///C:\Users\pablo\Desktop\TFG\Memoria%20TFG%20Pablo%20García%20Carbonell%20.docx#_Toc132277059)

[Tabla 3. Análisis de la competencia 28](#_Toc132277060)

# Introducción

Viajar es un estilo de vida, conocer otras culturas, otros comportamientos. El ser humano es un ser social y le gusta viajar [1]. Por otra parte, existen otros tipos de viaje, como puede ser viajar por trabajo o viajar para visitar a algún familiar o conocido. Sin embargo, también existen otros tipos de viaje, los que se realizan por diversión, como podrían ser viajes como forma de bienestar, viajes como capricho, viajes en familia o con amigos. Pero todos estos últimos tienen algo en común: viajar por turismo y ocio.

Los últimos años los viajes han sufrido un descenso considerable por una razón obvia, la pandemia causada por el COVID-19 [2], dando como ejemplo España, que se ve cómo ha perdido hasta 50 mil millones de euros en turismo a causa del coronavirus, como se puede ver en la Figura 1.

Gráfico, Gráfico de barras

Descripción generada automáticamente

Figura . Ingreso por año en España por el turismo.

(en asterisco (\*) se marcan los años con COVID)

<https://es.statista.com/estadisticas/673456/ingresos-anuales-por-turismo-extranjero-espana/>

Sin embargo, actualmente se podría decir que se ha superado la pandemia y convivimos con dicho virus sin problema. Por tanto, hoy en día la mayoría de gente aprovecha cualquier hueco para viajar, normalmente siendo escapadas de un par de días, fin de semana la mayoría de las veces [3].

Los amantes de los viajes, al querer visitar un país y otro sin parar, intentan planificar su viaje rápidamente, habiendo dos formas principales: la contratación de agencias de viaje como Viajes el Corte Inglés[[3]](#footnote-3) o *webs* como *Tripadvisor[[4]](#footnote-4)*, cada una con sus ventajas e inconvenientes. Al contratar una agencia no se pierde tiempo investigando, pero es necesario pagar a esta. De la otra forma, se realiza gratuitamente, pero es necesario realizar la búsqueda de forma personal y no por algún agente externo, perdiendo así bastante tiempo.

Entonces, ¿por qué no crear un servicio que recoja estas ventajas, pero suprima los inconvenientes? Es decir, una página web que realice la búsqueda de los lugares de interés por el usuario y de forma gratuita. Este es el principal objetivo de TouristApp, el trabajo de fin de grado que se recogerá en este documento.

Es posible que exista algún servicio parecido, pero el resultado suele ser muy general. Sin embargo, con esta propuesta se busca un resultado más concreto y orientado al usuario. Por ejemplo, en el caso de una escapada a Barcelona con el objetivo de ocio, no es lo mismo que realice el viaje un matrimonio de 50 años que un grupo de adolescentes.

Cogiendo como idea las 5 W del periodismo [4], la búsqueda de los lugares de interés se realizará mediante cuatro preguntas clave: dónde vas a viajar, cuándo es el viaje (fecha de inicio y fecha de fin), quién va a realizar el viaje (que cantidad de gente y rango de edad) y para qué viaja (con que objetivo). Respondiendo estas 4 preguntas se podrá dar un resultado muy concreto y preciso para el usuario, resolviendo así el problema comentado anteriormente de tener un resultado muy general.

Por tanto, con TouristApp se busca realizar un algoritmo de búsqueda de rutas turísticas adaptado a las condiciones, gustos y características de cada usuario basado en herramientas de procesamiento de lenguaje natural (PLN).

# Estudio de viabilidad

Antes de ponernos manos a la obra con el proyecto se realizará un previo estudio de viabilidad con el fin de tener una mejor perspectiva del proyecto a desarrollar, estudiando y analizando todas sus características y factores externos. Este estudio de factores internos y externos del producto se realizará mediante un análisis DAFO.

Una vez realizado este punto, se realizará un análisis de riesgos para poder ver los percances que pueden surgir a lo largo del desarrollo del TFG, y con ello preparar planes para poder evitar o solucionar todos los problemas que puedan aparecer.

## Análisis DAFO

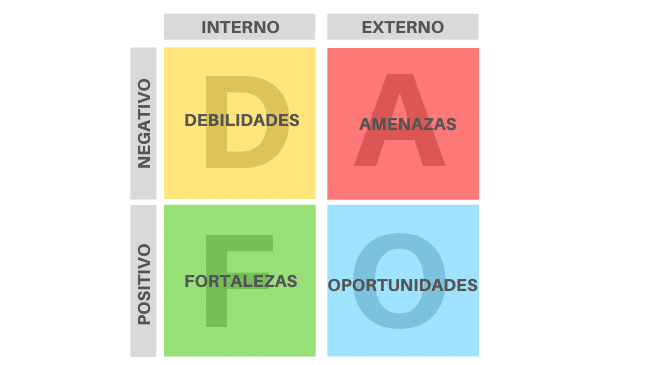
El análisis DAFO consiste en un esquema de trabajo utilizado para el estudio de un proyecto. Este esquema se divide en dos partes, factores externos e internos, las cuales se dividen en otras dos partes nuevas. La parte de factores externos se divide en **amenazas** y **oportunidades**, y la parte de factores internos en **debilidades** y **fortalezas**. De estas cuatro iniciales es de donde proviene el nombre este esquema: debilidades, amenazas, fortalezas y oportunidades, DAFO.

Figura . Esquema de un análisis DAFO

*Fuente*: <https://dircomfidencial.com/diccionario/analisis-dafo-20161113-1643/>

Como se observa en el esquema y se ha comentado anteriormente, el análisis DAFO se divide en dos secciones, las cuales se dividen en otras dos subsecciones. Los agentes internos, los cuales se pueden cambiar por el propio desarrollador del proyecto; y los agentes externos, los cuales afectan al proyecto, pero no dependen de él mismo, es decir, no son causados por el desarrollador. Sin embargo, aunque estos agentes externos no dependen del productor del proyecto, puede intentar cambiarlos evitando las amenazas o buscando las oportunidades.

En la Tabla 1 se muestra el análisis DAFO del proyecto para poder saber cuáles son los factores que mejorar y sobresalir entre las diferentes alternativas existentes en el mercado.

Tabla . Análisis DAFO

|  |  |  |
| --- | --- | --- |
| Factores externos | AMENAZAS  - Competencia con otras *apps* más conocidas. | OPORTUNIDADES  - Aprender a utilizar herramientas de PLN.  - Aprender *Python.*  - Conseguir experiencia.  - Pocas aplicaciones *web* con el mismo objetivo que el proyecto. |
| Factores internos | DEBILIDADES  - Falta de experiencia en proyectos de tales dimensiones.  - Creación y desarrollo de un proyecto en solitario.  - Poco conocimiento de Python y PLN. | FORTALEZAS  - Ambición tanto a la hora de afrontar el proyecto como de aprender nuevas tecnologías.  - Conocimiento en creación de aplicaciones *web*. |

En la Tabla 1 se ha realizado el análisis DAFO del proyecto, y a continuación se realizará de una forma más extendida:

**Factores externos:** agentes positivos y negativos externos al desarrollador del proyecto

* Amenazas: a la hora de hablar de las amenazas del proyecto no encontramos demasiadas más allá de la más obvia, que es la competencia con otras *apps* más conocidas, ya que existen alternativas a crear rutas turísticas o ver lugares de interés de una ciudad que visites.
* Oportunidades: a nivel personal y profesional, existe la oportunidad de aprender nuevas técnicas como son el uso de herramientas de procesamiento de lenguaje natural (PLN) y el lenguaje de programación de *Python*, con el que se implementarán la mayoría de los algoritmos de PLN. Por la parte más relacionada con el proyecto, la oportunidad de crear una *app web* como ninguna otra, aunque en amenazas se haya comentado que hay alternativas, existe la oportunidad de realizar un proyecto único como ningún otro. Además, todo esto aportará experiencia al desarrollador a la hora de realizar un proyecto de dicha magnitud.

**Factores internos:** agentes positivos y negativos del desarrollador del proyecto.

* Debilidades: La principal debilidad sería la falta de experiencia a la hora de realizar aplicaciones *web.* A lo largo de los cuatro años del grado se han realizado bastantes proyectos, pero siempre han sido de forma guiada, con pautas y ayudas. Sin embargo, ahora el proyecto se realizará desde cero y prácticamente sin guías ni ayudas. Además, realizar este proyecto de tal envergadura en solitario es otro problema a tener en cuenta, teniendo que repartir todas las tareas en uno solo con el fin de conseguir un producto usable. Por último, y hablando más de la parte del desarrollo en sí, el poco uso tanto de *Python* como de PLN a lo largo de la carrera es una gran debilidad, ya que tocará aprender un nuevo lenguaje de programación.
* Fortalezas: Ante las debilidades que han aparecido, la principal fortaleza es la ambición y las ganas, tanto de aprender nuevas tecnologías como *Python* o herramientas de PLN, como de sacar adelante un proyecto de tales dimensiones como es el TFG. Además, hay bastante conocimiento previo en la creación de aplicaciones *web*.

## Análisis de riesgos

Los riesgos de un proyecto son los distintos problemas e imprevistos que pueden surgir a lo largo del desarrollo del producto los cuales pueden llegar a parar o frenar el transcurso normal de este.

Por todo esto, se realizará un análisis de riesgos del proyecto con el objetivo de intentar evitarlos, y en el caso de sufrir un problema, poder sobrepasarlos de la manera más eficiente posible y menos perjudicial para el proyecto.

De cada riesgo se estudiarán los siguientes aspectos:

* **Riesgo:** descripción del riesgo que se puede sufrir.
* **Probabilidad:**posibilidad de que ocurra el riesgo. Determinaremos distintos niveles de probabilidad:
  + Improbable
  + Poco probable
  + Probable
  + Muy probable
* **Gravedad:** consecuencias que hay si el riesgo ocurre. Dentro de este aspecto se ha realizado una división para diferenciar la gravedad de cada riesgo:
  + **1:** Muy leve, retraso de un par de días, una semana como mucho**.**
  + **2:** Leve, retraso de semanas.
  + **3:** Grave, retraso de un mes.
  + **4:** Muy grave, tener que empezar el proyecto de nuevo.
* **Plan de contingencia:** método para solucionar un riesgo que ya ha sucedido de la manera más rápida y eficiente posible.
* **Plan de prevención:** método para prevenir que suceda un riesgo.

En la se muestra el análisis de riesgos del proyecto:

Tabla . Análisis de riesgos

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Riesgo | Probabilidad | Gravedad | Plan de contingencia | Plan de prevención |
| Enfermedad del desarrollador | Probable | 2 | Descansar y seguir las pautas del médico | Intentar cuidar la salud tanto física como mental |
| Pérdida del código del proyecto | Poco probable | 4 | Realizar el código de nuevo con todo lo aprendido anteriormente | Copias de seguridad en la nube y subir código en *GitHub[[5]](#footnote-5)* |
| Avería del ordenador | Probable | 3 | Arreglarlo de la mejor forma posible mientras se utiliza otro para no perder el hilo de trabajo | Tener todo ordenador y no descargar nada extraño de Internet |
| Mala planificación del proyecto | Poco probable | 3 | Retrasar las fechas de las tareas o realizar una nueva planificación con nuevas estrategias | Realizar una buena planificación mediante las estrategias y metodologías correctas. Trabajar de forma eficiente |
| No terminar a tiempo el proyecto | Poco probable | 3 | Intentar tener una buena planificación | Organizarse bien y trabajar eficientemente |
| Pérdida de la memoria del TFG | Poco probable | 4 | Al igual que con el código, volver a realizarla e intentar que quede lo más similar posible | Realizar guardados automáticos tanto en local como en la nube |
| Problemas con el lenguaje de Python | Probable | 1 | Seguir investigando y buscando información sobre el lenguaje y ver similitudes con otros conocidos | Realizar un estudio e informarse sobre el lenguaje para tener una buena base |
| Problemas con las herramientas de PLN | Probable | 1 | Investigar e indagar más sobre el tema además de buscar otras alternativas | Realizar un estudio previo sobre PLN y las librerías necesarias para la realización del proyecto |
| Elección incorrecta de las tecnologías | Poco probable | 2 | Ver las razones por las qué no ha funcionado una tecnología y ver qué otra se puede usar | Estudiar y documentarse bien sobre las tecnologías que vamos a usar y ver cuáles son las óptimas |
| Saturación entre el TFG y ABP[[6]](#footnote-6) | Probable | 2 | Reorganizar de forma correcta para poder compaginar los dos proyectos | Tener una buena organización e intentar solventar el ABP lo antes posible para poder centrar el 100% del trabajo al TFG |

# Planificación

Antes de ponernos manos a la obra con el desarrollo del proyecto, es necesario realizar una buena planificación del proyecto

# Estado del arte.

En este apartado se realizará una investigación sobre el tema que se tratará en el TFG y el entorno al que pertenece el desarrollo de este, es decir, a aplicaciones *webs* destinadas a crear rutas turísticas, las características que podemos encontrar y las tecnologías con la que son desarrolladas. Esto permitirá conocer más sobre el tema en el que nos vamos a manejar y así poder aplicar soluciones correctas en nuestro producto final.

Por una parte, se estudiará e investigará sobre el tema principal a tratar en este proyecto, el procesamiento del lenguaje natural, y por otra parte la generación de rutas turísticas, estudiando otras opciones existentes dentro del mercado.

## 4.1 Procesamiento del Lenguaje Natural (PLN)

El procesamiento del lenguaje natural [5] es el campo de conocimiento de la Inteligencia Artificial, computación y de la lingüística que estudia el comportamiento entre el humano y la máquina, es decir, estudia y hace posible las interacciones entre ambos. Desde que la máquina fue creada, el humano ha tenido la necesidad de comunicarse con ella. Hoy en día, la relación humano-máquina está aumentando de forma exponencial, no solo en comunicarse con la máquina sino también a través de ella, como ocurre con las redes sociales. Por tanto, el PLN busca permitir, facilitar y suavizar esta comunicación, por lo que este sistema debe conocer el lenguaje, como puede ser las propias palabras, cómo realizar enunciados con sentido, saber el contexto de estos, etc.

### 4.1.1 Modelos para el PLN

Tratar computacionalmente una lengua del mundo significa un proceso de modelización matemática. Normalmente, los mensajes son codificados en los lenguajes de programación C, Java o Python. Por tanto, hay que realizar una preparación para que el programador lo implemente en un código eficiente y funcional. Existen dos modelos principales:

* **Modelos lógicos** (gramáticas): Los lingüistas escriben reglas de reconocimiento de patrones estructurales, empleando un formalismo gramatical concreto. Estas reglas, en combinación con la información almacenada en diccionarios computacionales, definen los patrones que hay que reconocer para resolver la tarea (buscar información, traducir, etc.).
* **Modelos probabilísticos del lenguaje** (basados en ejemplos): La aproximación es a la inversa: los lingüistas recogen colecciones de ejemplos y datos y a partir de ellos se calculan las frecuencias de diferentes unidades lingüísticas y su probabilidad de aparecer en un contexto determinado. Calculando esta probabilidad, se puede predecir cuál será la siguiente unidad en un contexto dado, sin necesidad de recurrir a reglas gramaticales explícitas. Es el paradigma de “aprendizaje automático” que se ha impuesto en las últimas décadas en Inteligencia Artificial: los algoritmos infieren las posibles respuestas a partir de los datos observados anteriormente.

### 4.1.2 Niveles/Componentes del PLN

La arquitectura de un sistema de PLN se basa en una definición de lenguaje natural por niveles, los cuales son los siguientes cinco:

- **Nivel fonológico**: trata de cómo las palabras se relacionan con los sonidos que representan

- **Nivel morfológico**: trata de cómo las palabras se construyen a partir de unas unidades de significado más pequeñas llamadas morfemas

- **Nivel sintáctico**: trata de cómo las palabras pueden unirse para formar oraciones, fijando el papel estructural que cada palabra juega en la oración y qué sintagmas son parte de otros sintagmas

- **Nivel semántico**: trata del significado de las palabras, y de cómo los significados se unen para dar significado a una oración, también se refiere al significado independiente del contexto, es decir, de la oración aislada.

- **Nivel pragmático**: trata de cómo las oraciones se usan en distintas situaciones y de cómo el uso afecta al significado de las oraciones.

Una vez vistos los niveles podemos ver cómo sería el algoritmo que realiza la máquina una vez se le proporcione el mensaje:

1. El usuario expresa qué es lo que desea hacer

2. La computadora analiza las oraciones proporcionadas, en el sentido morfológico y sintáctico.

3. El siguiente paso es analizar las oraciones semánticamente, es decir, saber cuál es el significado de cada oración, y asignar el significado de éstas a expresiones lógicas (V/F).

4. Una vez realizado el paso anterior, podemos hacer el análisis pragmático de la instrucción, es decir, una vez analizadas las oraciones, ahora se analizan todas juntas, tomando en cuenta la situación de cada oración. Una vez realizado este paso ya tiene la expresión final.

5. Una vez obtenida la expresión final, el siguiente paso es la ejecución de ésta, para obtener así el resultado y poder proporcionárselo al usuario.

### 4.1.3 Aplicaciones del PLN

Una vez profundizado en el tema del PLN podemos ver qué tipo de aplicaciones lo utilizan. Las aplicaciones más comunes o conocidas son el análisis y síntesis de voz, la traducción automática, la recuperación de la información, respuesta a preguntas, extracción de la información, reconocimiento del habla o creación de resúmenes, entre otras.

## 4.2 Estudio de la competencia

Tras un estudio sobre aplicaciones *web* relacionadas con el turismo, se han encontrado algunos resultados cuyo objetivo es parecido al del proyecto, es decir, crear rutas turísticas o aconsejar sobre los sitios de importancia a nivel de turismo de una ciudad. Además de forma adicional, se pedirá al ChatGPT que realice una ruta turística, ya que viendo últimamente la cantidad de usuarios que hacen uso de este, por qué no probarlo con nuestro mismo objetivo.

### 4.2.1. Tripadvisor

Posiblemente a nivel de consejos a la hora de viajar, Tripadvisor[[7]](#footnote-7) sea la página *web* más conocida por la mayoría de los usuarios. Esta página recoge las reseñas de usuarios de distintos aspectos del turismo, como pueden ser los hoteles, restaurantes, cosas que hacer en el destino que visites. Su página principal es muy sencilla: consta de un simple buscador en el cual puedes poner tu ciudad destino, como se puede ver en la Figura 2.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Figura . Página de inicio de Tripadvisor.

Fuente: <https://www.tripadvsor.es/>

Interfaz de usuario gráfica, Sitio web

Descripción generada automáticamenteUna vez realices una búsqueda, te mostrará una lista sobre las entradas con mejores opiniones de los usuarios, ordenando estas entradas en tres grupos: qué cosas hacer, alójate y come. Además, la propia página proporciona una *review* sobre cada entrada. A continuación, en la Figura 3 se mostrará el resultado al buscar Madrid.

Figura . Resultado de búsqueda de Madrid en Tripadvisor

<https://www.tripadvisor.es/Tourism-g187514-Madrid-Vacations.html>

Esta opción de mercado a parte de recoger valoraciones, comentarios y opiniones de otros usuarios es gratuita y en la mayoría de casos te da la opción de comprar o reservar, ya sea una entrada para algún museo, un hotel, etc. Sin embargo, te da una lista de resultados muy general. Por ejemplo, al buscar Madrid, como se ha hecho anteriormente ha salido una lista de resultados de más de mil opciones en cada una de las secciones, es decir, ha dado como resultado mil entradas sobre restaurantes, mil entradas sobre hoteles y mil entradas sobre lugares de interés. Al ser un resultado tan amplio no se está adaptando realmente a las características del usuario, es decir, Tripadvisor daría el mismo resultado tanto a un hombre de 50 años que a una chica de 19.

### 4.2.2 Triposo

Triposo es una aplicación para móviles completamente gratuita que aconseja al usuario sobre dónde puede ir de viaje según sus gustos. La *app* conoce los gustos del usuario ya que este se los habrá proporcionado antes o los habrá extraído de Facebook[[8]](#footnote-8), al haber iniciado sesión con él anteriormente. Además, mientras el usuario esté de viaje, esta *app* recomendará lugares de interés al instante. Sin embargo, la aplicación solo está disponible en inglés, lo que dificulta el uso a una cantidad de usuarios, además de presentar el problema de la mayoría de *webs* de este ámbito: presenta un resultado muy general.

### 4.2.3 FourSquare

FourSquare [[9]](#footnote-9) es una aplicación tanto web como para móvil que, dada una ciudad y un objetivo, es decir, qué es lo que buscas, te da una lista de resultados. Como se puede ver en la Figura 4, la interfaz de su página de búsqueda es muy sencilla.

Interfaz de usuario gráfica, Sitio web

Descripción generada automáticamente

Figura . Página principal de FourSquare

<https://es.foursquare.com/city-guide>

Mapa

Descripción generada automáticamenteA continuación, realizaremos una búsqueda para comprobar los resultados que proporciona, en este caso buscaremos Madrid, con el objetivo ‘Diversión’. Como se puede ver en la Figura 5, el resultado de la búsqueda se muestra de dos formas: una en forma de entrada, a la cual puedes acceder y ver las opiniones y valoraciones del usuario, y otra en forma de ubicación dentro del mapa de la ciudad que has indicado en el buscador

Figura . Resultado búsqueda FourSquare

<https://es.foursquare.com/explore?mode=url&ne=40.517192%2C-3.526096&q=Diversi%C3%B3n&sw=40.347591%2C-3.813801>

Foursquare da un resultado más preciso de lo que se quiere, es decir, se puede indicar si se busca un sitio para comer o un lugar de diversión. Sin embargo, se vuelve al mismo inconveniente que Tripadvisor, y es que el resultado es muy general, es decir, el mismo resultado saldrá tanto para un hombre de 50 años como para una chica de 19.

### 4.2.4 Agencias de viaje

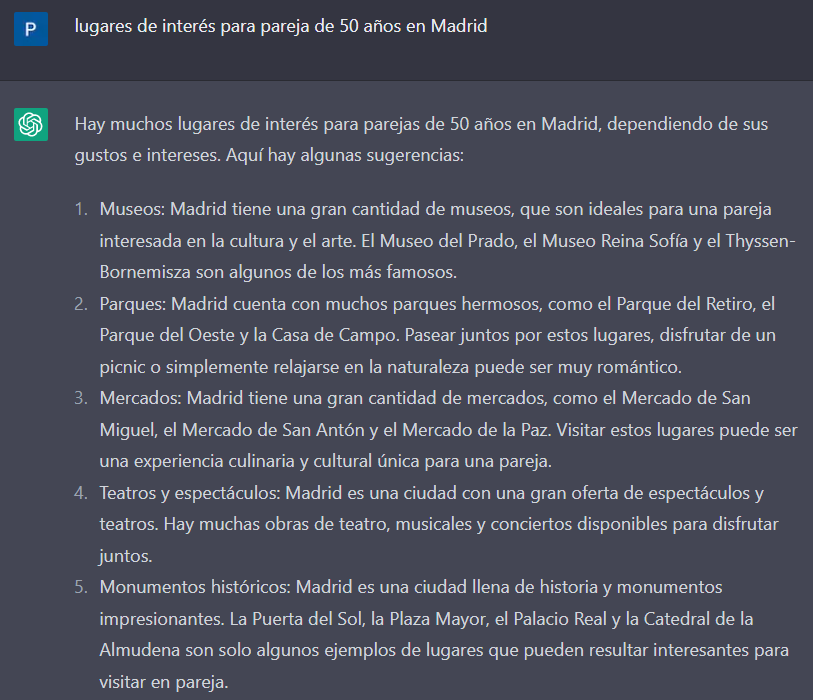
Otra opción puede ser contratar una agencia de viaje, como puede ser el caso de Viajes el Corte Inglés[[10]](#footnote-10), uno de los más conocidos a nivel nacional. Sin embargo, estas agencias se centran más en las partes de hospedaje y el medio de transporte por el que se viaja, es decir, se encargan de encontrar varias opciones de hotel y de medios de transporte. Una vez encontradas las mejores y dadas las mejores opciones al usuario, este tendrá que elegir cuál es la que más le gusta o la que más le conviene. Este no es principalmente el objetivo del proyecto. Las agencias se centran en la parte de llegar al lugar y hospedarse. Sin embargo, nosotros buscamos más la parte de turismo, qué zonas le interesan al usuario dependiendo de sus gustos. Además, las agencias no son gratuitos, se debe pagar por sus servicios.

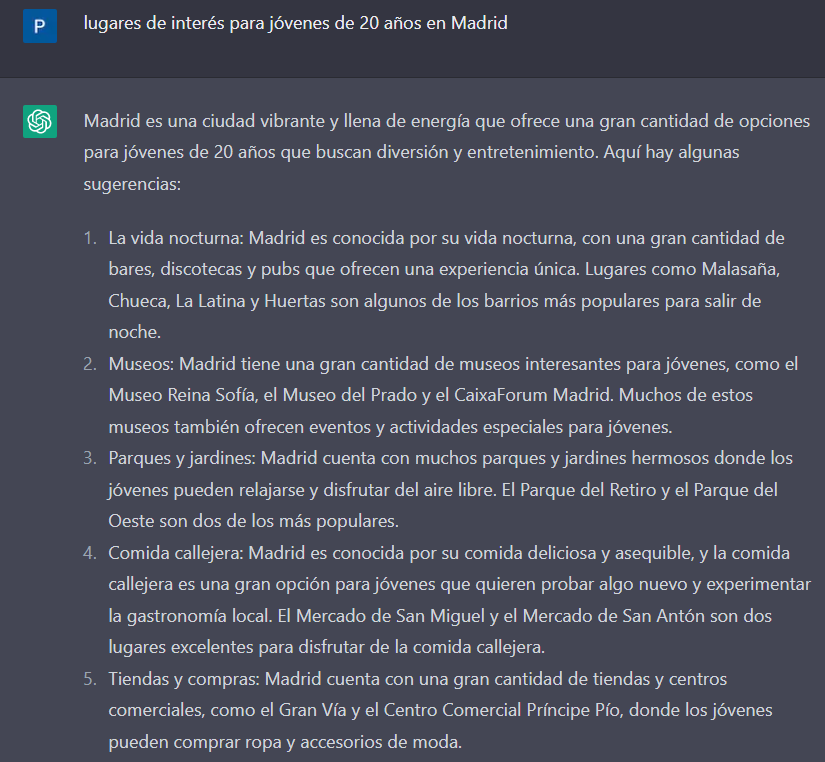
Este no es principalmente el objetivo del proyecto. Las agencias se centran en la parte de llegar al lugar y hospedarse. Sin embargo, nosotros buscamos más la parte de turismo, qué zonas le interesan al usuario dependiendo de sus gustos. Además, las agencias no son gratuitos, se debe pagar por sus servicios.

### 4.2.5 ChatGPT

El ChatGPT es [[11]](#footnote-11) una inteligencia artificial (IA) que en los en los últimos tiempos está de moda, ya que ha sido entrenada para poder tener conversación con cualquier persona. Lo más sorprendente de esta IA concreto es que es capaz de dar unas respuestas muy acertadas y completas, incluso de varios párrafos. Además, en estas respuestas es capaz de expresarte de manera natural y con información muy exacta, lo que hace muy complicado distinguir que el texto ha sido generado por IA o por un humano [6]. ChatGPT nos interesa porque realiza todo este tipo de búsquedas mediante herramientas de procesamiento natural.

Por probar cuál puede ser el resultado, se le pedirá al ChatGPT que nos realice una ruta turística por Madrid, primero para una pareja de 50 años, y después para un grupo de 20 años, y así se estudiarán los dos resultados, viendo si es capaz de adaptarse a las características del usuario.

En la Figura 6 podemos ver el resultado que daría a lugares de interés para el matrimonio de 50 años, y en la Figura 7 los lugares de interés para un grupo de 20 años, ambos viajando a Madrid.

Como se puede observar, el ChatGPT sí realiza una selección de lugares de interés en función de las características del usuario, diferenciando los lugares de ocio entre los dos casos dados. Además, puede llegar a ser más específico, como por ejemplo dando solo restaurantes y bares Texto

Descripción generada automáticamentesi se le solicita, como se puede observar en la Figura 8.

Figura Respuesta “lugares de interés para pareja de 50 años en Madrid” por el ChatGPT.

<https://chat.openai.com/chat>

Figura . Respuesta “lugares de interés para jóvenes de 20 años en Madrid” por el ChatGPT.

<https://chat.openai.com/chat>

Figura . Resultado a comida para jóvenes de 20 años por ChatGPT.

<https://chat.openai.com/chat>

Por tanto, el ChatGPT sí que da un resultado concreto y adaptado a las características del usuario.

En este proyecto se busca un resultado parecido al dado por ChatGPT, pero siendo un servicio solo de turismo y viajes, y no tan general como ChatGPT, que aborda todos los temas posibles. Además, si es posible, se apoyará la respuesta con algún tipo de elemento gráfico, como puede ser imágenes o algún mapa.

|  |  |  |
| --- | --- | --- |
|  | Ventajas | Inconvenientes |
| Tripadvisor | -Recoge opiniones variadas de distintos usuarios sobre cada entrada.  - Divide los resultados de búsqueda en ‘Come’, ‘Alójate’ y ‘Haz cosas’.  - Es gratuito.  - Interfaz sencilla e intuitiva. | - Resultado de búsqueda muy general y que no está adaptado a las características del usuario.  - Lista muy amplia de resultados (mil entradas aproximadamente por búsqueda). |
| Triposo | - Consejos sobre el viaje adaptados a los gustos y características del usuario.  - Es gratuito. | - Solo disponible en inglés.  - Resultado de búsqueda muy amplio (hasta 500 entradas por búsqueda). |
| FourSquare | - Disponible en *app web* y *app* móvil  - Interfaz sencilla e intuitiva.  - Doble resultado de búsqueda: tanto tipo entrada como ubicación dentro del mapa (apoyo gráfico).  - Se puede indicar el objetivo de la búsqueda (comida, ocio, deporte…)  - Es gratuito | - Lista muy amplia de resultados (hasta 700 entradas por búsqueda).  - Resultado de búsqueda muy general.  - Resultado no adaptado al usuario (mismo resultado a un usuario de 50 años que a uno de 20). |
| Agencias de viaje | - Se centra en los gustos del usuario  - No pierdes tiempo previo para estudiar las zonas de interés. | - Es de pago  - Se centra mayoritariamente en el hospedaje y en el medio de transporte |
| ChatGPT | - Es gratuito.  - Se centra en los gustos y características del usuario.  - No da una lista muy amplia de resultados (entre 5 y 10) | - No es una página centrada en el turismo.  - No se acompaña con elementos gráficos como imágenes. |

Tabla . Análisis de la competencia

### 4.2.6 Comparación de la competencia

Tras estudiar la competencia llegamos a la conclusión de que la mayoría dan un resultado muy amplio o un resultado no adaptado al usuario. Por tanto, buscaremos realizar un servicio que dé un resultado con un rango de 5 a 10 entradas, adaptado a las características dadas por el usuario y, si es posible, acompañado con algún elemento gráfico como imágenes.

# Objetivos

En este apartado se definirá el objetivo principal a alcanzar con el desarrollo del proyecto y los subobjetivos a desarrollar para llevar a cabo el objetivo principal.

Para ello se hará uso de la estrategia SMART (*Specific, Measurable, Attainable, Realistic and Timely*) [7], con la que podremos basar nuestros objetivos en unas características imprescindibles para dar valor a nuestro proyecto.

El objetivo principal del proyecto es proponer y desarrollar una aplicación *web* accesible y gratuita que proporcione al usuario una lista de los lugares que más le puedan interesar de una ciudad sabiendo las características y gustos de este, como la edad y el objetivo por el que viaja. Cada respuesta se acompañará tanto con elementos gráficos como por una pequeña reseña, incluso algún enlace directo a otra página externa, como a la página de un teatro para poder comprar entradas o algo por el estilo. La búsqueda de estos lugares de interés se desarrollará utilizando herramienta de procesamiento de lenguaje natural.

Además, surgen los siguientes subobjetivos:

* Diseñar de una interfaz sencilla e intuitiva.
* Elaborar un estudio de mercado de forma completa analizando las alternativas existentes a nuestro proyecto.
* Analizar y decidir las tecnologías más apropiadas para desarrollar el trabajo.
* Permitir a cualquier usuario acceder a la aplicación.
* Planificar el desarrollo de la aplicación mediante las herramientas de gestión de proyectos correspondiente.
* Evaluar la utilidad y la efectividad de la aplicación desarrollada.
* Aprender nociones básicas sobre el procesamiento del lenguaje natural y de las tecnologías del lenguaje humano, orientadas a la comprensión y generación de información.
* Aprender y utilizar herramientas y librerías específicas de Procesamiento de Lenguaje Natural, así como *frameworks* para diseñar arquitecturas.

A parte de estos subobjetivos a nivel profesional, surgen otros subobjetivos más enfocados a lo personal:

* Utilizar un lenguaje de programación nuevo como Python.
* Realizar un documento de las dimensiones de la memoria de un TFG.
* Aprender a realizar completamente una aplicación pasando por todas sus fases: desde investigar a fondo lo que existe ahora en el mercado, pensar qué se necesita – tecnologías, etc. - y cómo añadirlo en el proyecto, diseñar sus interfaces y, desarrollarlo hasta que esté terminada.

# Metodología

## 6.1 Metodología *Scrum*

La metodología que se usará a lo largo del proyecto será la metodología ágil de *Scrum*. La metodología Scrum permite abordar proyectos complejos desarrollados en entornos dinámicos y cambiantes de un modo flexible. Está basada en entregas parciales y regulares del producto final en base al valor que ofrecen a los clientes [8].

Además, en la metodología *Scrum* existen tres roles diferentes []:

* *Scrum Máster:* gestionar el correcto funcionamiento del proyecto y eliminar impedimentos
* *Product Owner:* encargado de optimizar y maximizar el valor del producto
* Equipo de desarrollo: desarrollar el producto, autoorganizándose y autogestionándose para conseguir entregar un incremento de software al final del ciclo de desarrollo.

En nuestro caso, al ser el desarrollo de un trabajo en individual no habrá reuniones de equipo, sino que serán con el tutor del TFG. En el caso de este proyecto, serán reuniones semanales. Y aunque no nos centremos en la parte de mejorar el trabajo en equipo, sí que usaremos esta metodología por la parte de las iteraciones, con límite de tiempo cada una, de modo de a cada una se le establecen unas tareas y funcionalidades.

En la Figura 10 se muestra una representación gráfica de la metodología *Scrum*. 

Figura . Metodología ágil Scrum.

<https://www.sinnaps.com/blog-gestion-proyectos/metodologia-scrum>

## 6.2 *Kanban*

El método de *Kanban* es una forma de gestionar los proyectos basados en metodología ágil. *Kanban* es un método *Lean*, muy popular, de gestión del flujo de trabajo para definir, gestionar y mejorar los servicios que proporciona el trabajo de conocimiento. Te ayuda a visualizar el trabajo, maximizar la eficiencia y mejorar continuamente. El trabajo se representa en tableros Kanban, lo que te permite optimizar la entrega de trabajo a través de múltiples equipos y manejar, incluso los proyectos más complejos en un solo entorno [10].

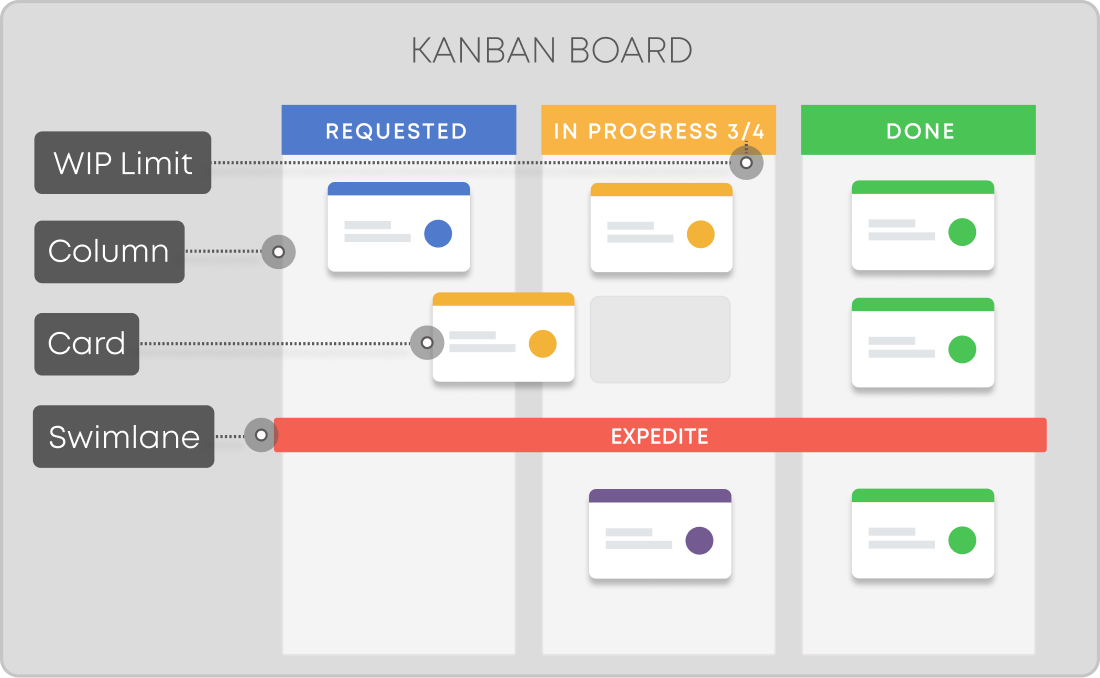


Figura 11. Lienzo Kanban.

<https://kanbanize.com/es/recursos-de-kanban/primeros-pasos/que-es-kanban>

El lienzo *Kanban* consta de un tablero dividido en cuatro secciones principalmente:

* Lista de tareas: conjunto de todas las tareas a realizar
* *To do:* conjunto de tareas con más prioridad para hacer
* En proceso: conjunto de tareas que están realizándose en ese momento
* Hecho: conjunto de tareas que ya están acabadas

Para aplicar el método *Kanban* se ha utilizado la herramienta online de *Trello[[12]](#footnote-12)*, la cual te permite crear un lienzo y poder mover las secciones entre las diferentes columnas de forma sencilla. En la Figura 12 se muestra nuestro *Kanban* desde *Trello*.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Figura . Kanban del proyecto desde Trello.

*Fuente propia*

Fuen

# 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.

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

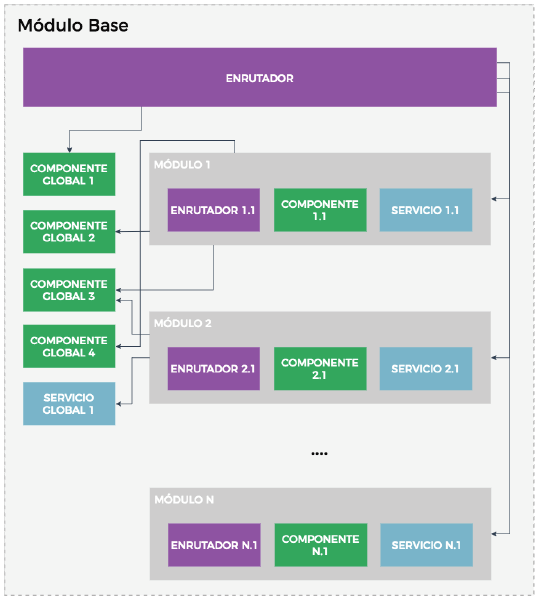
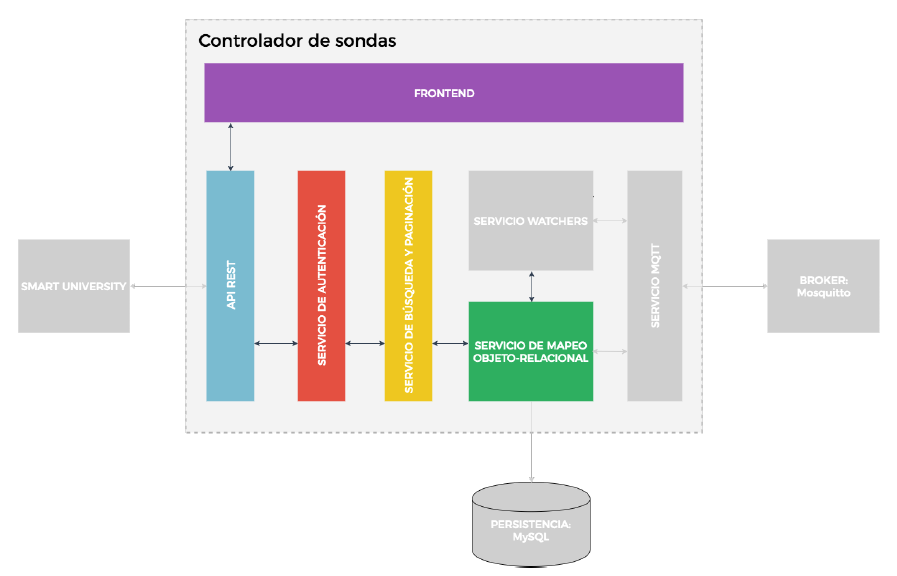


Figura . Diseño arquitectura conceptual panel IoT.

(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].

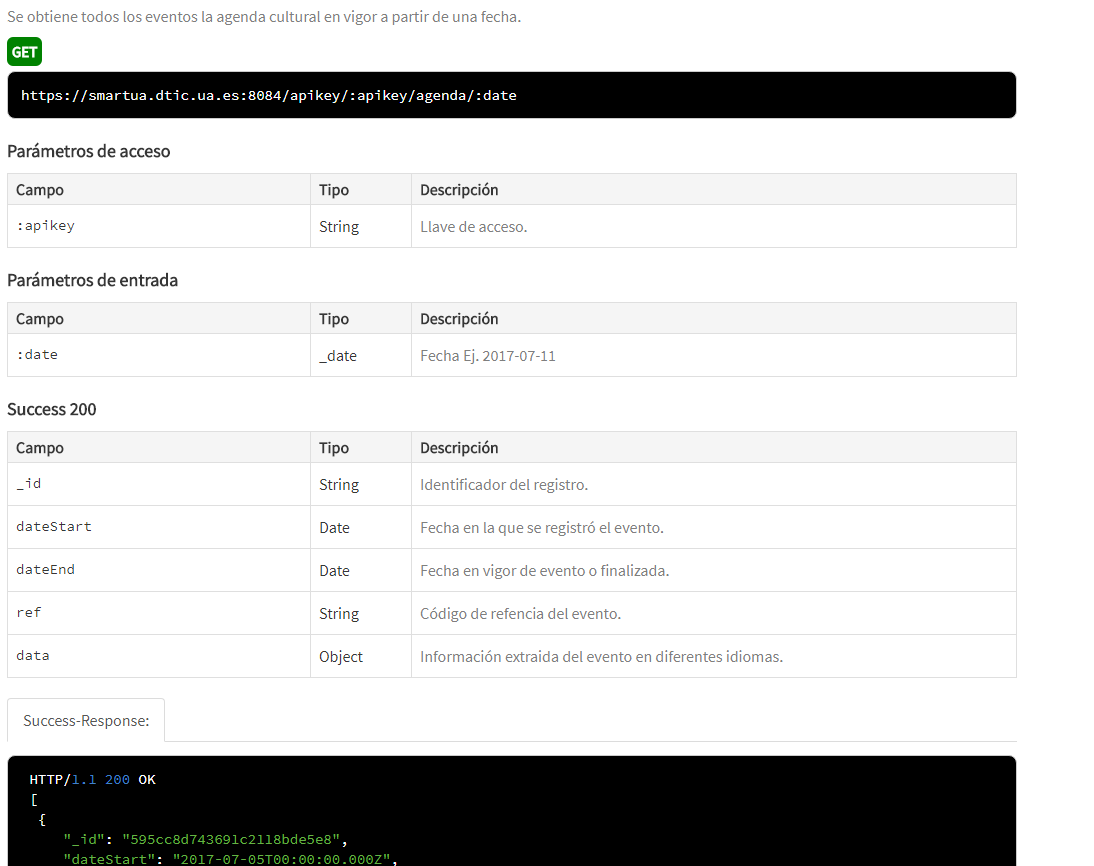


Figura . Especificación de la Open API de Smart University

(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.

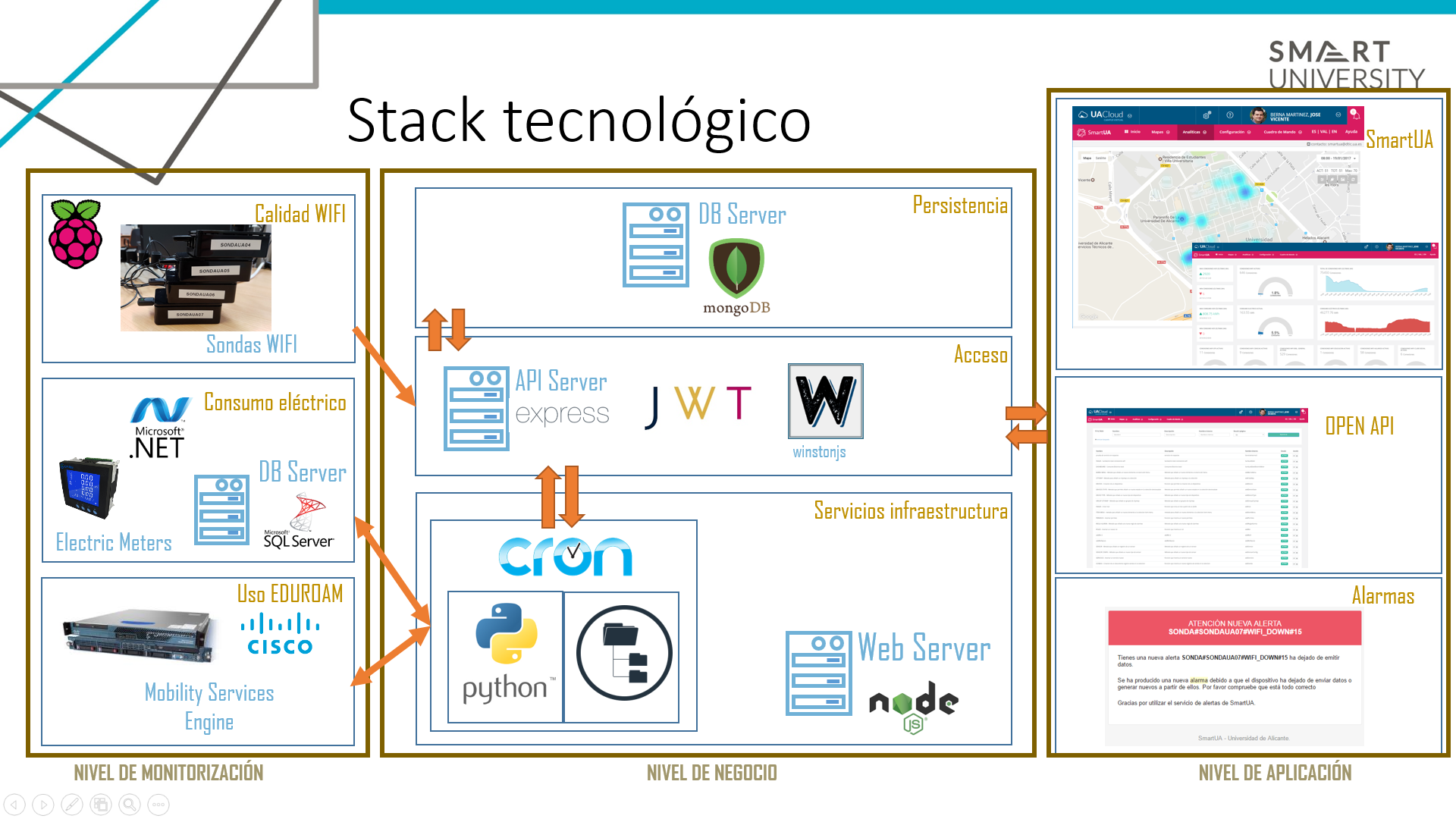


Figura . Stack tecnológico del proyecto Smart University

(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 . 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.

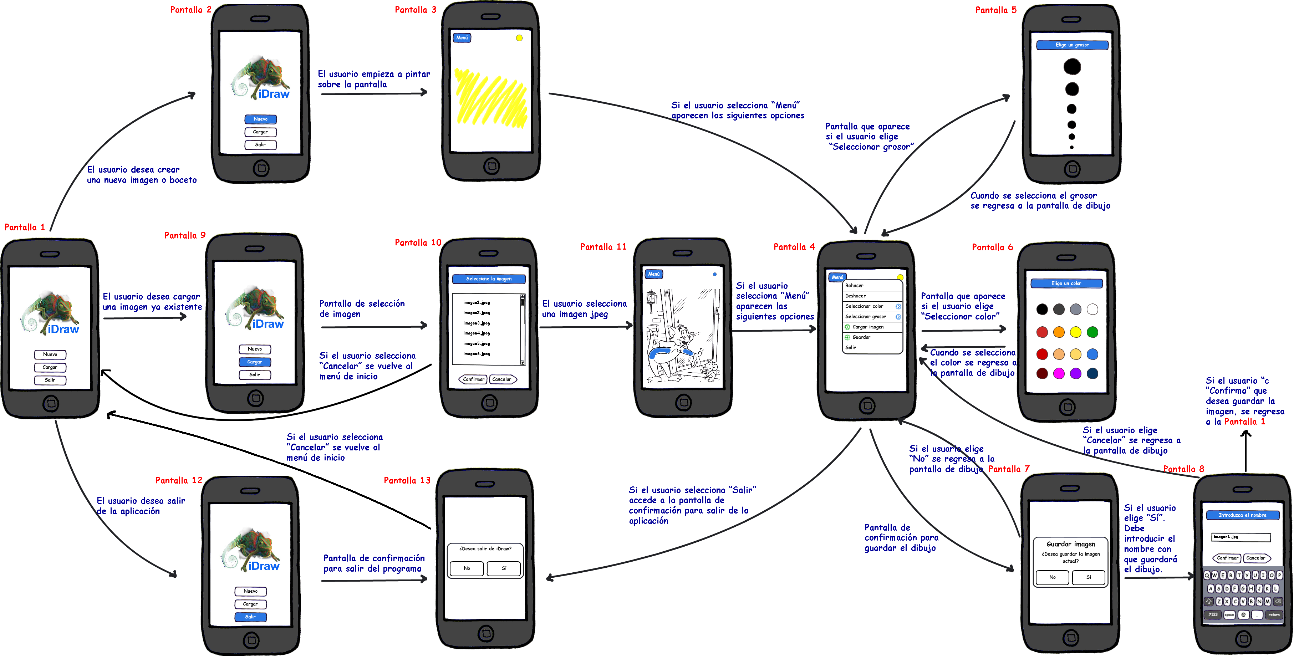


Figura . Ejemplo de mapa de interacciones.

(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].

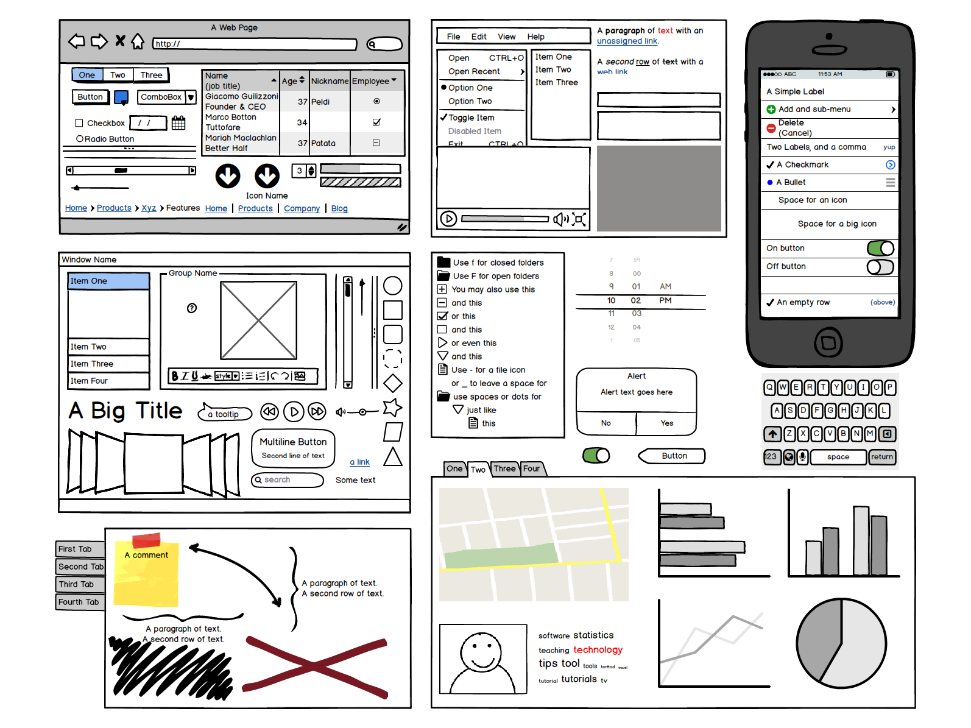


Figura . Ejemplo de wireframe con Balsamiq.

(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].

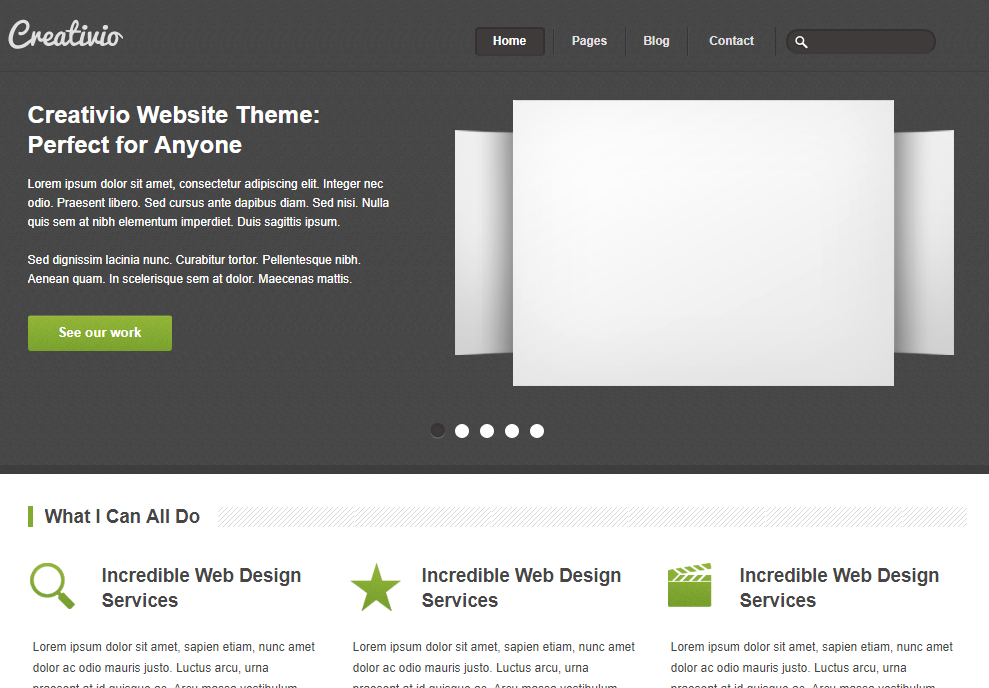


Figura . Mockup desarrollado con Justinmind

(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].

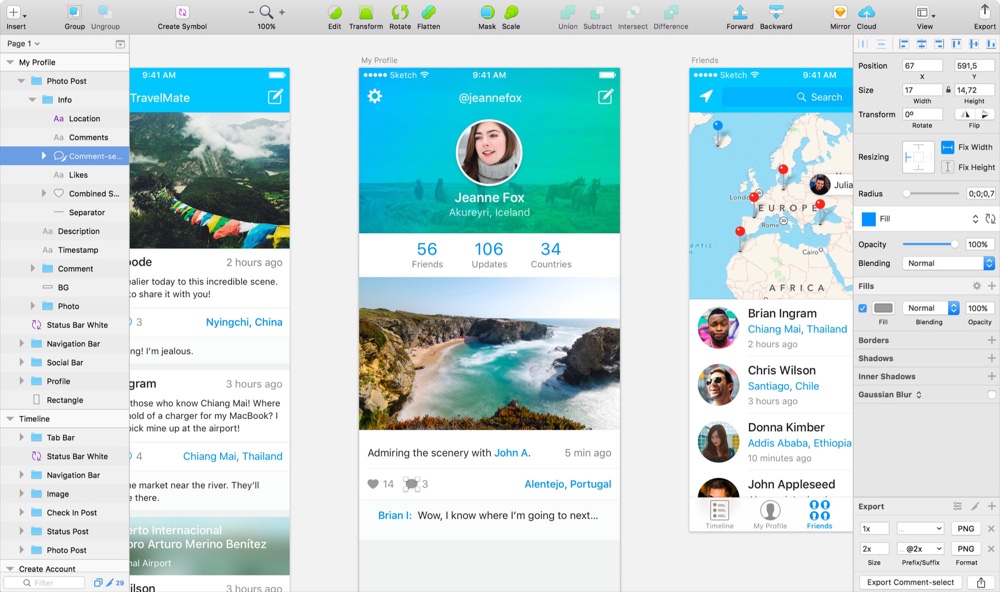


Figura . Diseño desarrollado con Sketch.

(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.

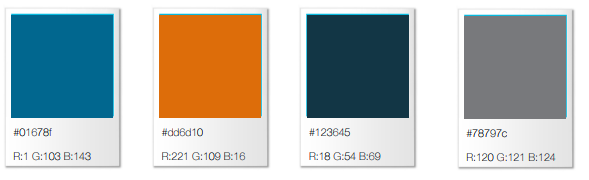


Figura . Paleta de colores Unviersidad de Cádiz

(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.

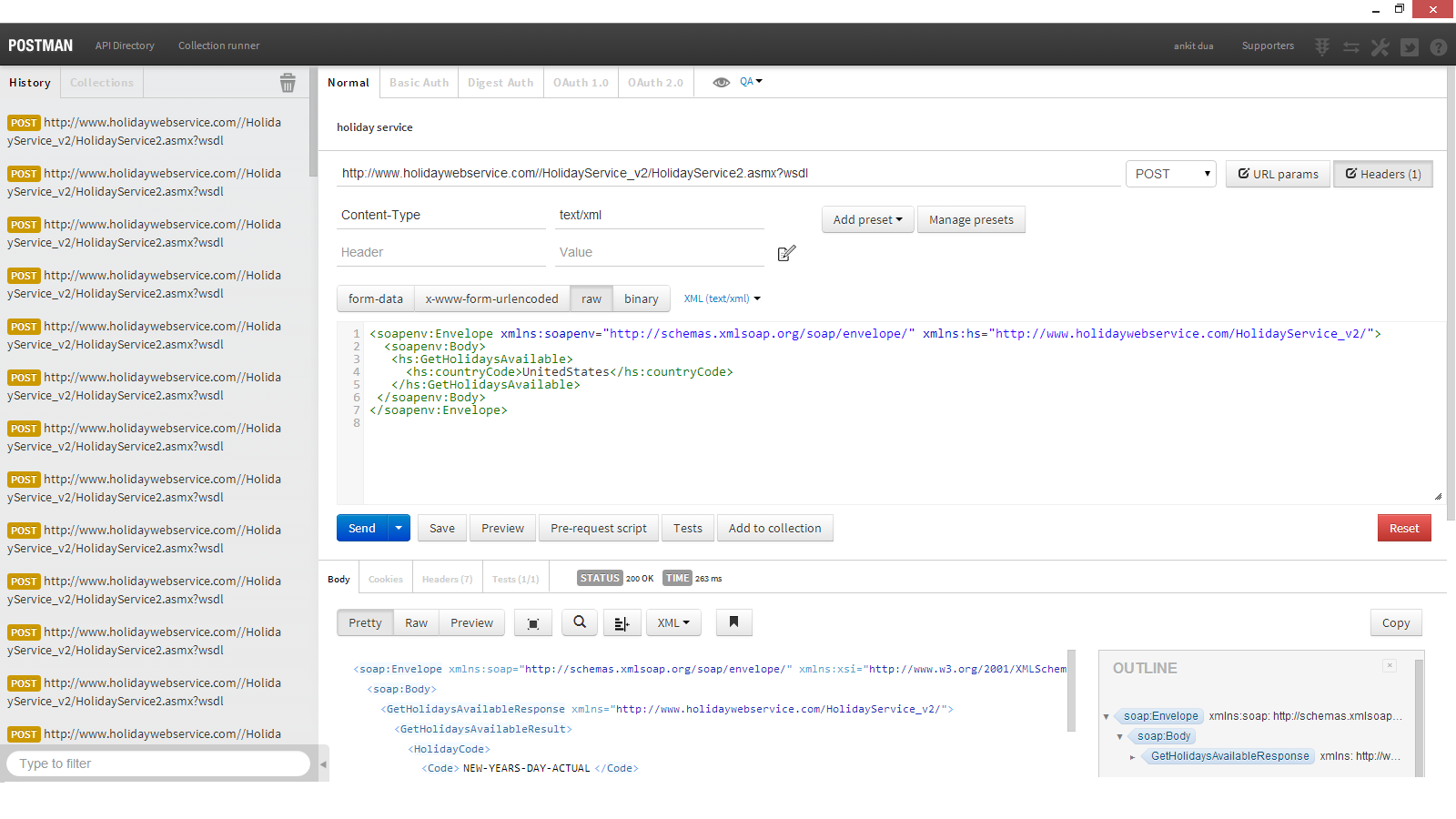


Figura . Ejemplo de ejecución de varias invocaciones a servicio.

(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

1. Esta es la razón científica por la que alguien ama viajar (2016). <https://www.semana.com/hogar-y-familia/articulo/por-que-me-gusta-viajar/60709/>
2. Hinojosa, V. (2022). Cómo ha cambiado el COVID nuestra forma de viajar: ochos nuevos hábitos.<https://www.hosteltur.com/150165_como-la-covid-ha-cambiado-nuestra-forma-de-viajar-ocho-nuevos-habitos.html>
3. Turismo en España: las cifras que reflejan cómo los viajes y escapadas hicieron crecer la economía en 2022. (2023) <https://www.cronista.com/espana/economia-finanzas/turismo-en-espana-las-cifras-que-reflejan-como-los-viajes-y-escapadas-hicieron-crecer-la-economia-en-2022/>
4. Waisbord, S. (2019). The 5Ws and 1H of digital journalism. *Digital Journalism*, *7*(3), 351-358
5. Moreno, A (2022). Procesamiento del lenguaje natural, ¿Qué es?.<https://www.iic.uam.es/inteligencia/que-es-procesamiento-del-lenguaje-natural/>
6. Fernández, Y (2023). ChatGPT: qué es, cómo usarlo y qué puedes hacer con este chat de inteligencia artificial GPT-3. <https://www.xataka.com/basics/chatgpt-que-como-usarlo-que-puedes-hacer-este-chat-inteligencia-artificial>
7. Doran, G. T. There's a S.M.A.R.T. Way to Write Management's Goals and Objectives. Management Review, Vol. 70 (11), pp. 35-36. 1981
8. Hurtado Sánchez, J (2021). Cómo funciona la Metodología Scrum: Qué es y cómo utilizarla. <https://www.iebschool.com/blog/metodologia-scrum-agile-scrum/>
9. Roche, J (2021). Scrum: roles y responsabilidades. Los 3 roles de la metodología Scrum. <https://www2.deloitte.com/es/es/pages/technology/articles/roles-y-responsabilidades-scrum.html>.
10. (2020). ¿Qué es Kanban? Explicación para principiantes. .<https://kanbanize.com/es/recursos-de-kanban/primeros-pasos/que-es-kanban>

# 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 . Primer boceto de la aplicación.

(Fuente SmartBlog <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.

1. <https://www.tripadvisor.es/> [↑](#footnote-ref-1)
2. <https://www.trivago.es/> [↑](#footnote-ref-2)
3. <https://www.viajeselcorteingles.es/> [↑](#footnote-ref-3)
4. <https://www.tripadvisor.es/> [↑](#footnote-ref-4)
5. <https://github.com/> [↑](#footnote-ref-5)
6. <https://eps.ua.es/es/ingenieria-multimedia/gestioncontenidos/que-es-abp.html> [↑](#footnote-ref-6)
7. <https://www.tripadvisor.es/> [↑](#footnote-ref-7)
8. <https://es-es.facebook.com/> [↑](#footnote-ref-8)
9. <https://es.foursquare.com/city-guide> [↑](#footnote-ref-9)
10. <https://www.viajeselcorteingles.es/> [↑](#footnote-ref-10)
11. <https://chat.openai.com/chat> [↑](#footnote-ref-11)
12. <https://trello.com/> [↑](#footnote-ref-12)