

---

# **AdaptaMaterialEscolar: Herramienta para la adaptación de asignaturas a necesidades educativas especiales**

---



## **Trabajo de Fin de Grado**

Pablo Miranda Torres  
Natalia Rodríguez-Peral Valiente  
Jorge Velasco Conde

## **Directores**

Virginia Francisco Gilmartín  
Raquel Hervás Ballesteros

Departamento de Ingeniería del Software e Inteligencia Artificial  
Facultad de Informática  
Universidad Complutense de Madrid

**Junio 2021**

Documento maquetado con TEXIS v.1.0+.

Este documento está preparado para ser imprimido a doble cara.

# AdaptaMaterialEscolar: Herramienta para la adaptación de asignaturas a necesidades educativas especiales

*Memoria que presentan para optar al título de Grado en Ingeniería  
del Software*

**Pablo Miranda Torres**  
**Natalia Rodríguez-Peral Valiente**  
**Jorge Velasco Conde**

*Dirigida por los Doctores*  
**Virginia Francisco Gilmartín**  
**Raquel Hervás Ballesteros**

*Versión 1.0+*

**Departamento de Ingeniería del Software e Inteligencia  
Artificial**  
**Facultad de Informática**  
**Universidad Complutense de Madrid**

**Junio 2021**



# Índice

<b>1. Introducción</b>	<b>1</b>
1.1. Motivación . . . . .	1
1.2. Objetivos . . . . .	2
1.3. Estructura de la memoria . . . . .	3
<b>2. Estado del Arte</b>	<b>5</b>
2.1. Adaptación Curricular . . . . .	5
2.1.1. Lectura fácil para adaptación curricular . . . . .	6
2.1.2. Pictogramas para adaptación curricular . . . . .	7
2.1.3. Herramientas que facilitan adaptaciones curriculares .	9
2.2. Diseño Centrado en el Usuario . . . . .	10
2.2.1. Conocer al usuario . . . . .	11
2.2.2. Prototipado . . . . .	12
2.2.3. Evaluación . . . . .	14
<b>3. Herramientas empleadas</b>	<b>17</b>
3.1. Moqups . . . . .	17
3.2. React . . . . .	18
<b>4. Metodología de desarrollo</b>	<b>21</b>
4.1. Introducción . . . . .	21
4.2. Kanban . . . . .	23
4.3. Tipos de pruebas . . . . .	27
4.3.1. Pruebas de memoria . . . . .	28
4.3.2. Pruebas de implementación . . . . .	28
<b>5. AdaptaMaterialEscolar</b>	<b>33</b>
5.1. Captura de requisitos . . . . .	33
5.1.1. Primera iteración . . . . .	33
5.1.2. Segunda iteración . . . . .	36
5.1.3. Tercera iteración . . . . .	40
5.2. Diseño de la aplicación . . . . .	41

---

5.2.1. Primera etapa . . . . .	41
5.2.2. Segunda etapa . . . . .	41
5.2.3. Tercera etapa . . . . .	44
<b>6. Trabajo Individual</b>	<b>53</b>
6.1. Pablo . . . . .	53
6.2. Natalia . . . . .	53
6.3. Jorge . . . . .	53
<b>Bibliografía</b>	<b>55</b>

# Índice de figuras

2.1. Ejemplo de pictograma de una rosa. . . . .	8
2.2. Ejemplo de pictograma que representa la acción comer. . . . .	9
2.3. Ejemplo de Sistema Pictográfico de Comunicación . . . . .	10
2.4. Ejemplo del buscador de pictogramas de ARASAAC y las opciones que ofrece . . . . .	11
2.5. Interfaz de ARAWORD. . . . .	12
2.6. Ejemplo de guión gráfico. . . . .	13
2.7. Ejemplo de boceto en papel. . . . .	14
2.8. Ejemplo de prototipo en papel interactivo. . . . .	15
3.1. Interfaz de Moqups . . . . .	18
3.2. Menú lateral de interacciones . . . . .	19
3.3. Funcionalidad sin usar JSX . . . . .	20
3.4. Funcionalidad empleando JSX . . . . .	20
4.1. Modelo en cascada . . . . .	23
4.2. Modelo en espiral . . . . .	24
4.3. Tablero <i>Kanban</i> al inicio del proyecto . . . . .	27
4.4. Tarea asignada a varios usuarios . . . . .	31
5.1. Boceto inicial de la aplicación . . . . .	42
5.2. Versión 1.0 de la página principal . . . . .	43
5.3. Versión 1.0 del editor . . . . .	44
5.4. Versión 1.0 de la búsqueda de pictogramas . . . . .	45
5.5. Versión 1.0 de la adaptación de actividades . . . . .	46
5.6. Versión 1.0 de la adaptación de temario . . . . .	47
5.7. Prototipo de la página principal (Jorge) . . . . .	48
5.8. Prototipo de la página principal (Natalia) . . . . .	48
5.9. Prototipo de la página principal (Pablo) . . . . .	49
5.10. Prototipo del editor (Jorge) . . . . .	49
5.11. Prototipo del editor con un desplegable en uno de los menús (Jorge) . . . . .	50

5.12. Prototipo de la vista previa (Jorge) . . . . .	50
5.13. Prototipo del editor (Natalia) . . . . .	51
5.14. Prototipo del editor (Pablo) . . . . .	51

# Índice de Tablas

5.1.	Leyenda de puntuaciones. . . . .	38
5.2.	Puntuación de los requisitos I. . . . .	38
5.3.	Lista de funcionalidades ordenada por prioridad. . . . .	39
5.4.	Lista de opciones de personalización ordenada por prioridad. .	39
5.5.	Puntuación de los requisitos II. . . . .	40
5.6.	Lista de requisitos ordenada por prioridad. . . . .	41



# Capítulo 1

## Introducción

**RESUMEN:** En este capítulo se realiza la introducción del Trabajo de Fin de Grado que va a ser presentado. En la sección 1.1 se explicará la motivación que ha dado lugar al trabajo. En la sección 1.2 el objetivo que se pretende alcanzar. Por último, en la sección 1.3 se explica cómo está estructurada la memoria.

### 1.1. Motivación

La educación escolar tiene como finalidad promover el desarrollo de ciertas capacidades y el aprendizaje de determinados contenidos necesarios para que los alumnos puedan ser miembros activos de la sociedad. Para ello, la escuela debe ofrecer una respuesta educativa que evite la discriminación y asegure la igualdad de oportunidades.

El sistema educativo español, en la actualidad, está organizado en ocho etapas o niveles que garantizan el derecho a una educación inclusiva para el alumnado en cada fase de su desarrollo cognitivo y emocional; de éstas, únicamente dos son obligatorias: Educación Primaria (EP) y Educación Secundaria Obligatoria (ESO).

En España hay aproximadamente algo menos de 2 millones de alumnos con necesidades educativas especiales (NEE), aunque no todos ellos presentan algún tipo de discapacidad, frente a los 8 millones de estudiantes que hay en enseñanzas de régimen general no universitarias.

La motivación del proyecto es proporcionar una herramienta de trabajo para el profesorado que facilite la inclusión de cualquier alumno que necesite algún tipo de adaptación curricular a lo largo de su formación académica y, para ello, se va a seguir un diseño orientado al usuario.

La inclusión es “[...] el proceso de identificar y responder a la diversidad de las necesidades de todos los estudiantes a través de la mayor participación”

*en el aprendizaje, las culturas y las comunidades, y reduciendo la exclusión en la educación. Involucra cambios y modificaciones en contenidos, aproximaciones, estructuras y estrategias, con una visión común que incluye a todos los niño/as del rango de edad apropiado y la convicción de que es la responsabilidad del sistema regular, educar a todos los niño/as.”*

En este TFG nos vamos a centrar en la adaptación curricular no significativa, este tipo de adaptación supone a los profesores tiempo y esfuerzo, y este TFG se propone como objetivo reducir el tiempo y el esfuerzo que deben dedicarle a estas tareas.

## 1.2. Objetivos

El objetivo de este TFG es desarrollar una herramienta de trabajo para el profesorado que permita adaptar los recursos de las asignaturas en cualquier formato de manera intuitiva, fácil y rápida, y por tanto se puedan crear unidades didácticas personalizadas que se ajusten a las necesidades de cada alumno.

Los materiales, recursos y contenidos que se presentan y usan en los centros educativos están regulados por ley en el currículum educativo o escolar. El currículum educativo trata de garantizar que todos los alumnos terminan cada curso igual de preparados.

El currículo educativo es la regulación de los elementos que determinan los procesos de enseñanza y aprendizaje de cada una de las asignaturas e incluye: los objetivos de cada enseñanza y etapa educativa, las competencias y contenidos, la metodología didáctica, los estándares y resultados de aprendizaje y los criterios de evaluación.

En el currículo escolar existen unas necesidades educativas comunes, compartidas por todos los alumnos. Sin embargo, no todos los estudiantes se enfrentan con las mismas herramientas al aprendizaje, cada estudiante tiene una necesidad individual. La mayoría de las necesidades individuales de los estudiantes son resueltas a través de actuaciones “sencillas”: dar más tiempo al alumno para el aprendizaje de determinados contenidos, diseñar actividades complementarias... Sin embargo, existen necesidades individuales que no pueden ser resueltas por estos medios, siendo necesarias una serie de medidas pedagógicas especiales distintas de las que requieren habitualmente la mayoría de los alumnos. En este caso se habla de necesidades educativas especiales, para atender estas necesidades son necesarias adaptaciones curriculares. Existen dos tipos de adaptaciones curriculares:

- Adaptación no significativa: no se modifican los contenidos curriculares de las asignaturas, sino que se adaptan los materiales, exámenes... Los encargados de realizar estas adaptaciones serán los profesores.
- Adaptación significativa: se eliminan apartados del currículum oficial.

Los profesores dedican demasiado tiempo a la realización del material académico de los alumnos que necesitan adaptaciones de cualquier tipo. Entre otras cosas, tienen que ajustar la fuente del texto y el tamaño, buscar imágenes en Internet o escanearlas de los libros, redactar resúmenes resaltando la información más relevante, etc.

La herramienta que resulte de este TFG permitirá crear material escolar personalizado ajustado a cada alumno, permitiendo generar resúmenes, esquemas, traducciones a pictogramas, cambios de formato o diferentes tipos de ejercicios (como por ejemplo, llenar espacios en blanco, sopas de letras o relacionar conceptos), etc.

El fin de la aplicación es ayudar a reducir la dificultad y el tiempo de preparación de temario, actividades o exámenes para alumnos que necesiten adaptaciones curriculares significativas o no significativas.

En cuanto a los objetivos académicos, nuestra meta principal es aplicar en un proyecto real los conocimientos adquiridos durante el Grado de Ingeniería del Software y ampliarlos.

### **1.3. Estructura de la memoria**



# Capítulo 2

## Estado del Arte

**RESUMEN:** En este apartado nos centraremos en explicar el contexto en el que se encuentra nuestro proyecto. Explicaremos en detalle las necesidades de las personas con discapacidad cognitiva y de cómo se tratan de satisfacer mediante adaptaciones curriculares. Además, mencionaremos algunas de las herramientas actuales para realizar adaptaciones curriculares orientadas a nuestro objetivo.

### 2.1. Adaptación Curricular

Los materiales, recursos y contenidos que se presentan y usan en los centros educativos están regulados por ley en el currículo educativo o escolar. El currículo educativo trata de garantizar que todos los alumnos terminan cada curso igual de preparados.

El currículo educativo incluye todos los recursos académicos, materiales y humanos necesarios para llevar a cabo el proyecto educativo marcado por la legislación.

Las principales funciones del currículo educativo son: determinar las asignaturas, contenidos y temáticas comunes a todos los alumnos, determinar los criterios de evaluación y logros que debe superar el alumnado y formalizar los estándares educativos que permitan cumplir unos objetivos comunes.

En el currículo escolar existen unas necesidades educativas comunes, compartidas por todos los alumnos. Sin embargo, no todos los estudiantes se enfrentan con las mismas herramientas al aprendizaje, cada estudiante tiene una necesidad individual. La mayoría de las necesidades individuales de los estudiantes son resueltas a través de actuaciones “sencillas”: dar más tiempo al alumno para el aprendizaje de determinados contenidos, diseñar actividades complementarias... Sin embargo, existen necesidades individuales que no pueden ser resueltas por estos medios, siendo necesarias una serie de medidas

pedagógicas especiales distintas de las que requieren habitualmente la mayoría de los alumnos. En este caso se habla de necesidades educativas especiales, para atender estas necesidades son necesarias adaptaciones curriculares.

La adaptación curricular (o adecuación curricular) es la modificación de elementos del currículo a fin de dar respuesta a las necesidades del alumnado. Las adaptaciones curriculares se clasifican en:

- **Adaptaciones Curriculares de Acceso al Currículo.** Responden a las necesidades de un grupo de personas. Estos pueden ser:
  - **De Acceso Físico.** Representa a los recursos materiales y espaciales. Ejemplos de estos son mobiliario adaptado, iluminación y sonoridad adaptada o profesorado especializado.
  - **De Acceso a la Comunicación.** Incluye materiales específicos de enseñanza tales como: aprendizaje, ayudas tecnológicas o sistemas de computación complementarios. Algunos ejemplos son el braille, el lenguaje de signos o la comunicación a través del ordenador.
- **Adaptaciones Curriculares Individualizadas.** Se realizan para un único alumno con el fin de responder a necesidades educativas especiales y que no pueden ser compartidas por el resto de los compañeros. Pueden ser:
  - **No Significativas.** Adaptan materiales, tiempos, actividades, metodologías, técnicas e instrumentos de evaluación sin modificar los contenidos curriculares de las asignaturas. Es la estrategia principal para conseguir la individualización de la enseñanza y por tanto, tienen un carácter preventivo y compensador.
  - **Significativas.** Modificaciones que se realizan desde la programación, previa evaluación psicopedagógica, y que afectan a los elementos prescriptivos del currículo oficial a modificar e incluso eliminar objetivos generales de la etapa, contenidos básicos y nucleares de las diferentes áreas curriculares y criterios de evaluación.

Unas de las adaptaciones curriculares más comunes y a las que está orientada nuestro proyecto son la adaptación de los documentos a la lectura fácil y la adición de pictogramas, que se explicarán en los siguientes apartados.

### 2.1.1. Lectura fácil para adaptación curricular

No todas las personas tienen la misma capacidad de lectura. Esta capacidad puede estar limitada por diferentes causas. Según el manual "Lectura fácil" de Óscar García Muñoz, las personas más afectadas por este problema

son las personas con discapacidad intelectual, como por ejemplo las personas con Síndrome de Down, las personas con enfermedades y trastornos mentales y del comportamiento, como las personas con Trastorno del Espectro Autista, las personas con dificultad para el desarrollo del lenguaje por discapacidad auditiva y las personas con circunstancias transitorias de dificultad en la comprensión lectora. Por ello, las adaptaciones a lectura fácil pueden suponer un gran beneficio para estos usuarios. Las adaptaciones curriculares para estos casos requieren documentos con una estructura más simple y esquemática. El manual de Lectura Fácil establece un gran número de reglas, tanto estéticas como de redacción, para poder facilitar estas adaptaciones y garantizar un texto de lectura fácil. Algunas de las medidas que nos resultan de gran interés son:

- Gramática. Evitar ciertos tiempos verbales, como el futuro o el subjuntivo o usar oraciones simples y cortas son algunas de las medidas gramaticales que propone.
- Ortografía. El punto y aparte realiza la función del punto y seguido, evitar el punto y coma y los puntos suspensivos, evitar corchetes y símbolos poco habituales, escribir números en cifra...
- Léxico. Reglas orientadas al vocabulario a usar. Son medidas como utilizar palabras sencillas expresadas de forma simple, evitar palabras largas, utilizar siempre el mismo sinónimo o evitar abreviaturas.
- Tipografía. Definen cómo debe ser el formato y estilo de la letra. Por ejemplo, el tamaño de letra debe de ser lo suficientemente grande, entre 12 y 16 puntos, utilizar tipografías sin remate como Arial o Helvetica o reforzar la nitidez de los números.
- Composición del texto. Describen cómo debe ser la apariencia del texto en conjunto. Algunas medidas son intentar tener una oración por línea, organizar el texto en bloques cortos, evitar partir las palabras con guiones, no partir una frase en dos páginas y utilizar amplios márgenes.
- Imágenes. Utilizar imágenes de apoyo al texto que estén bien referenciadas, utilizar símbolos o dibujos que para ideas, conceptos o temas abstractos. Uno de los principales tipos de imágenes de apoyo son los pictogramas, que se explicarán en el siguiente apartado.

### 2.1.2. Pictogramas para adaptación curricular

Las personas con carencias sensoriales, cognitivas o con un conocimiento insuficiente de la lengua de comunicación pueden presentar problemas a la hora de comunicarse. Por ello, los Sistemas Aumentativos y Alternativos de Comunicación (SAACs) se presentan como una gran opción para la

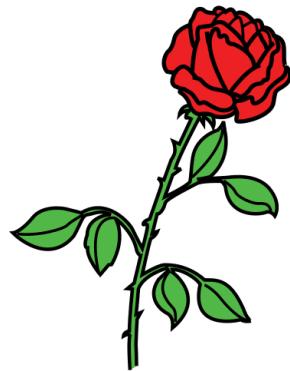


Figura 2.1: Ejemplo de pictograma de una rosa.

comunicación de estas personas. Uno de los SAACs más comunes son los pictogramas.

Un pictograma es un signo claro y esquemático que representa un objeto real, figura o concepto. Sintetiza un mensaje que puede señalar o informar sobre pasando la barrera de las lenguas. Es un recurso comunicativo de carácter visual que podemos encontrar en diversos contextos de nuestra vida diaria y nos aporta información útil por todos conocida. Los pictogramas son:

- Universales. Pueden ser entendidos por cualquier persona, independientemente de su idioma.
- Visuales. Muestran con claridad y sencillez al objeto/acción que representan.
- Inmediatos. La comunicación se establece entre el emisor y el receptor tan solo señalando sobre el pictograma adecuado.

Las figuras 2.1 y 2.2 son ejemplos de pictogramas, en el primero se muestra un lugar físico como puede ser una tienda y en el segundo se representa la acción comer.

Además, estos pictogramas se suelen representar en forma de Sistema Pictográfico de Comunicación (SPC). Los SPC facilitan la comunicación ya que representan la realidad mediante dibujos, con diferente grado de abstracción y apoyan al lenguaje oral o signado. La figura 2.3 muestra un ejemplo de SPC, donde se ve una serie de pictogramas que facilitan la comunicación. Una de las páginas que proporciona la facilidad de crear Sistemas Pictográficos es ARASAAC. El *Portal Aragonés de Comunicación Aumentativa y Alternativa (ARASAAC)*<sup>1</sup>, creado en 2007, ofrece un amplio catálogo de

---

<sup>1</sup><https://arasaac.org/>

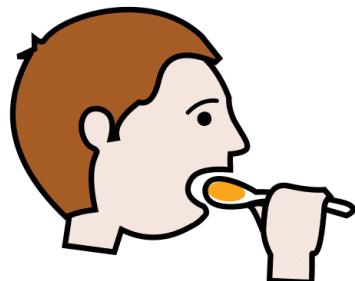


Figura 2.2: Ejemplo de pictograma que representa la acción comer.

pictogramas de libre acceso. Su principal herramienta es un buscador de pictogramas que ofrece varios resultados para una palabra introducida. Además, incluye opciones que permiten elegir el estilo del pictograma al gusto del usuario, como por ejemplo elegir si se desea en color o blanco y negro, incluir el título del pictograma, tacharlo, incluir señales que identifiquen símbolo plural, un identificador, etc. En la figura 2.4 se muestra un ejemplo del buscador con todas las opciones de personalización del pictograma. El portal de ARASAAC ofrece una *API*<sup>2</sup> (Interfaz de Programación de Aplicaciones) para desarrolladores que permite integrar el buscador de pictogramas en cualquier aplicación. Dicha API funciona mediante servicios web que devuelven los códigos de identificación de los pictogramas asociados a un texto de búsqueda, para así poder obtener las imágenes directamente. Además, se pueden añadir parámetros a servicio para obtener el pictograma con características concretas, como el color o el tamaño.

### 2.1.3. Herramientas que facilitan adaptaciones curriculares

La existencia de herramientas que faciliten adaptaciones curriculares es muy limitada. Existen herramientas enfocadas al uso de pictogramas, otras que mezclan pictogramas con procesamiento de texto.

Una de las herramientas que fusiona procesamiento de texto con pictogramas es *ARAWORD* una herramienta de procesador de textos creado por ARASAAC que permite la escritura simultánea de textos y pictogramas. Facilita la elaboración de materiales de comunicación aumentativa, la elaboración de documentos accesibles, y la adaptación de documentos para personas que presentan dificultades en estos ámbitos. ARAWORD resulta también una herramienta muy útil para ser usada por usuarios que están adquiriendo el proceso de la lectoescritura, ya que la aparición del pictograma,

<sup>2</sup><https://arasaac.org/developers/api>

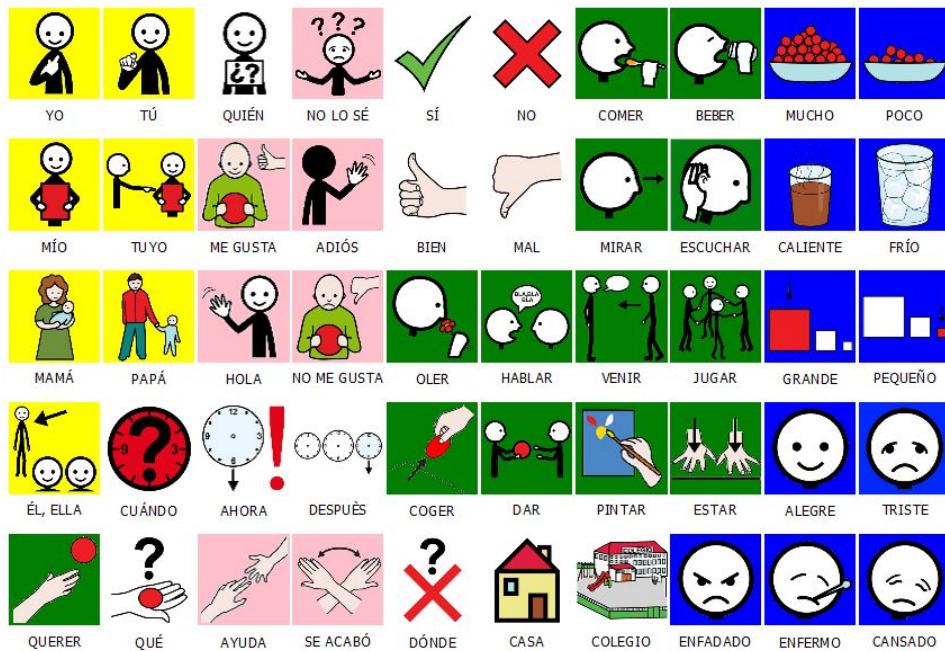


Figura 2.3: Ejemplo de Sistema Pictográfico de Comunicación

a la vez que se escribe, es un refuerzo muy positivo para reconocer y evaluar que la palabra o la frase escrita es correcta. En la figura 2.5 se muestra un ejemplo de la interfaz de ARAWORD, donde se traducen las oraciones a pictogramas.

Sin embargo, ofrece una experiencia de procesamiento de textos con pictogramas muy limitada y lejos de lo que es una experiencia completa de lectura fácil, ya que sólo admite pictogramas sobre texto sin posibilidad de personalización.

Esta limitación hace que las únicas herramientas que puedan conseguir el objetivo sean los procesadores de texto comunes, como por ejemplo Microsoft Word, pero de una manera tediosa e ineficiente, ya que estas aplicaciones no están diseñadas específicamente para ello.

## 2.2. Diseño Centrado en el Usuario

El Diseño Centrado en el Usuario (DSU) consiste en una forma de diseño cuyo objetivo es crear un producto que resuelva las necesidades de los usuarios finales, consiguiendo así una mayor calidad de producto y por tanto una mayor satisfacción de los usuarios finales. Con dicha metodología se pretende entender previamente a los usuarios para poder encontrar soluciones acordes a sus necesidades. Este proceso se basa en tres pilares: conocer al usuario,

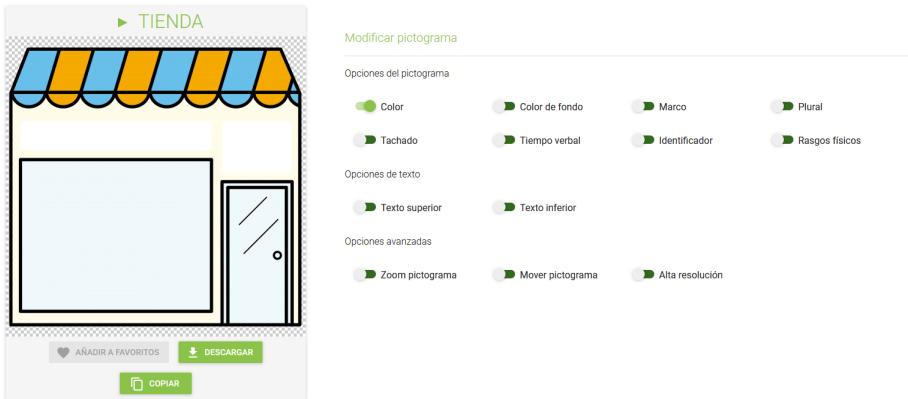


Figura 2.4: Ejemplo del buscador de pictogramas de ARASAAC y las opciones que ofrece

prototipado del producto y evaluación, que se explicarán en los siguientes apartados.

### 2.2.1. Conocer al usuario

Entender a los usuarios finales, el entorno y el uso que puedan darle al producto ayuda a diseñar productos que puedan encajar de mejor forma en dicho contexto y mejorar la forma en la que trabajan, comunican e interactúan. Por ello, se realizan estudios en los que se planea obtener información para conocer quiénes van a ser los usuarios reales del sistema, cómo se comportan y cuál es el contexto en el que se desenvuelven.

Algunas de las técnicas más usadas para poder investigar cuáles son las necesidades del usuario son:

- **Entrevistas.** Se realizan preguntas al sujeto de estudio. Responden a un conjunto de preguntas con el fin de aportar la información necesaria para poder comenzar con el diseño. Se suelen realizar diferentes tipos de preguntas, como las orientadas a objetivos, que pretenden descubrir las prioridades del usuario, orientadas al sistema, que identifican las funcionalidades más usadas, orientadas a flujos de trabajo, que identifican procesos y recurrencia de actividades, y las orientadas a actitudes.
- **Observación del usuario.** Analizar el comportamiento del usuario mientras realiza una actividad relacionada con el problema a resolver, para poder detectar si las necesidades reales coinciden con las necesidades escritas.
- **Diarios de uso.** Se proporciona a los usuarios un diario en el que

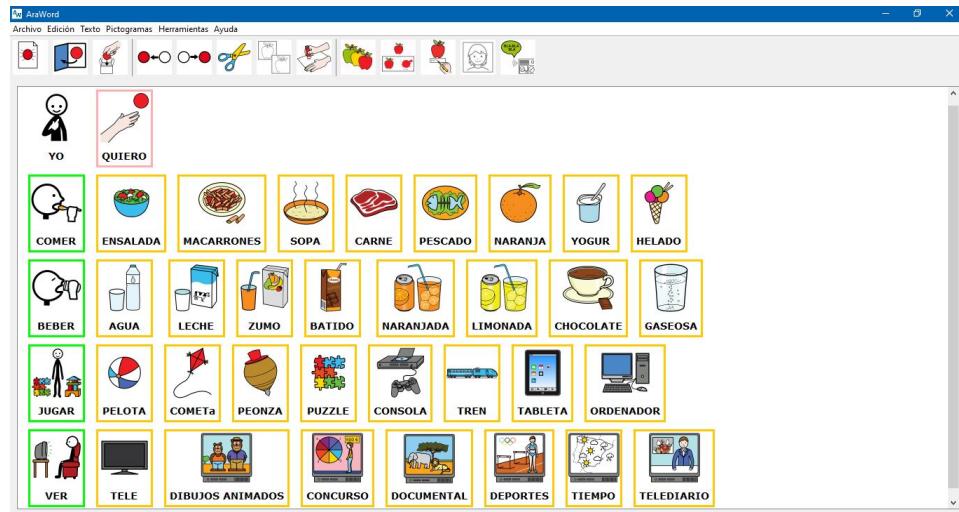


Figura 2.5: Interfaz de ARAWORD.

escriban información, como interacciones con el sistema, necesidades o un registro de comportamientos que engloban informaciones del usuario. Esta información servirá para conocer más a fondo lo que busca el usuario.

- **Analizar a la competencia.** Estudiar cómo los usuarios usan un sistema ya existente para poder descubrir qué características del producto son útiles y cuáles son los problemas que los usuarios encuentran en estos.

Gracias a estas técnicas se podrán definir de manera más precisa los requisitos del sistema para posteriormente proceder al prototipado de la aplicación.

### 2.2.2. Prototipado

Es importante tener clara cuál va a ser la interacción del usuario con el producto. Por ello, se hacen prototipos que simulan el diseño y comportamiento del sistema para que los clientes finales puedan dar su feedback y confirmen si es lo que realmente buscan. Dichos prototipos pueden ser realizados en papel o digital. Los prototipos en papel son:

- **Guiones gráficos.** Conjunto de ilustraciones pintadas a mano en forma de cómic, en el que se muestra un escenario y una secuencia de acciones que simulan casos de uso de la aplicación. En la figura 2.6 se muestra un ejemplo de guión gráfico que secuencia una situación en la



Figura 2.6: Ejemplo de guión gráfico.

que una aplicación de controlar eventualmente el estado de ánimo de una persona cercana pudiera ser útil.

- **Bocetos.** Diseños de interfaces dibujados a mano en papel. Se enfocan más en aspectos genéricos y de alto nivel, ignorando los detalles. En la figura 2.7 se muestra un ejemplo de un boceto en papel de una aplicación para móvil.
- **Prototipos en papel interactivos.** Es una maqueta o mockup de la interfaz hecha en papel y compuesta de las piezas que representan a la aplicación. Estos prototipos son interactivos, es decir, muestran una simulación del comportamiento del sistema ante acciones del usuario. Algunas formas de representar estas interacciones son mostrar cambios en la interfaz a raíz de acciones realizadas (como clicks o pulsaciones de botones del teclado). La figura 2.8 muestra un ejemplo de prototipo en papel interactivo en el que se muestran los cambios al realizar acciones sobre la aplicación.

Los prototipos digitales suelen ser mucho más fieles al diseño final del sistema. Además, permiten diseñar interacciones de manera más real. Algunas herramientas para realizar prototipos digitales son Photoshop, Moqups, o incluso HTML y CSS.

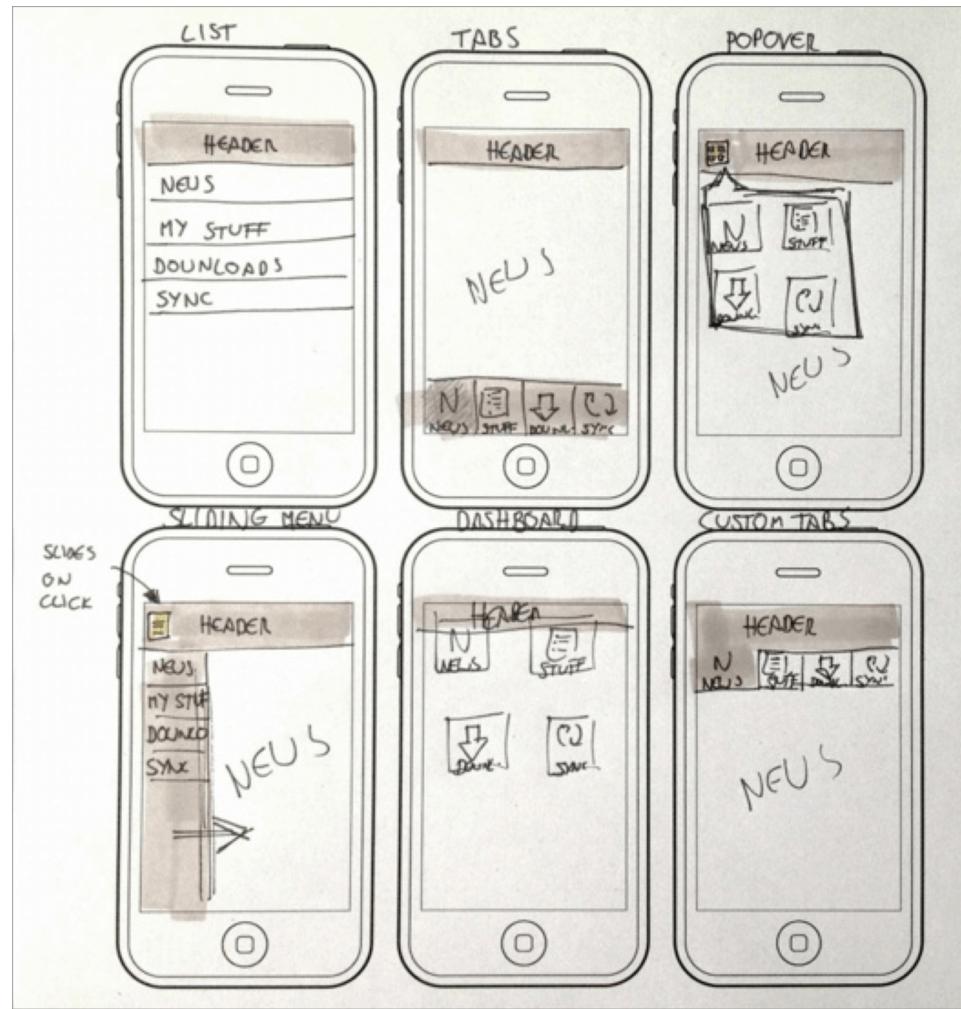


Figura 2.7: Ejemplo de boceto en papel.

### 2.2.3. Evaluación

Una interfaz tiene que ser probada y evaluada por el usuario final para garantizar que responde a las expectativas o detectar problemas de usabilidad. Existen dos tipos de evaluación, las heurísticas (sin usuarios) y las que se realizan con usuarios:

- **Evaluación heurística.** Identifica los problemas de usabilidad siguiendo directrices universales. Estas evaluaciones suelen realizarse por expertos en experiencia de usuario y facilitan la identificación temprana de problemas de usabilidad.
- **Evaluación con usuarios.** Simula ejecuciones del sistema realizadas

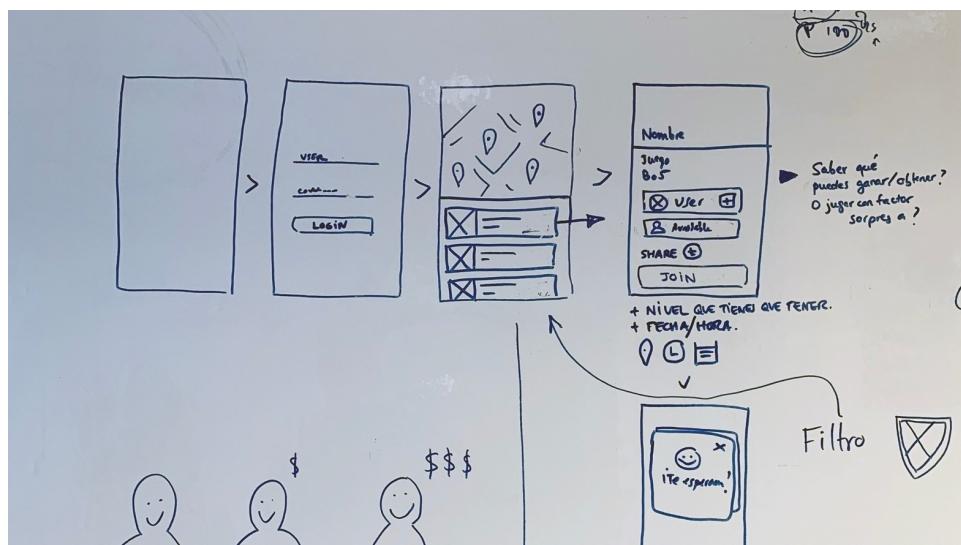


Figura 2.8: Ejemplo de prototipo en papel interactivo.

por los usuarios finales. Los usuarios interactúan con el programa y observan hasta qué punto son capaces de realizar determinados conjuntos de tareas. Además, se obtiene un feedback completo del proceso mediante un plan de evaluación que incluye, entre otros, preguntas a los usuarios.



# Capítulo 3

## Herramientas empleadas

**RESUMEN:** En este capítulo se explican las herramientas y librerías que se han empleado en el desarrollo de este proyecto. En la sección 3.1 se introduce la herramienta online Moqups. En la sección 3.2 se enseña el “framework” que se ha utilizado para la creación de la página web.

### 3.1. Moquups

Moquups<sup>1</sup> es una página web para la creación de bocetos, prototipos, diagramas, etc. Permite diseñar una interfaz, o simplemente ver cómo es el flujo de un algoritmo, arrastrando, desde los menús al espacio de trabajo, los distintos elementos (llamados “plantillas”) que podemos encontrar en una aplicación (barras de progreso, etiquetas o *labels*, enlaces, diferentes tipos de ventanas, etc.).

Existe la posibilidad de añadir comentarios e iconos, y poder crear diferentes páginas en un mismo proyecto (por ejemplo, crear varias vistas de una aplicación), así como añadir interacción a los elementos (como por ejemplo, cuando le des clic a un botón, éste realice una función específica). También es colaborativo, es decir, puedes invitar a más miembros para trabajar en equipo; y permite la gestión de roles.

En la Figura 3.1 se puede ver la interfaz de un proyecto en blanco. Se observa que en el menú lateral de la izquierda, se encuentran los apartados para la creación del prototipo (plantillas, páginas que tiene el proyecto,

---

<sup>1</sup><https://moqups.com/>

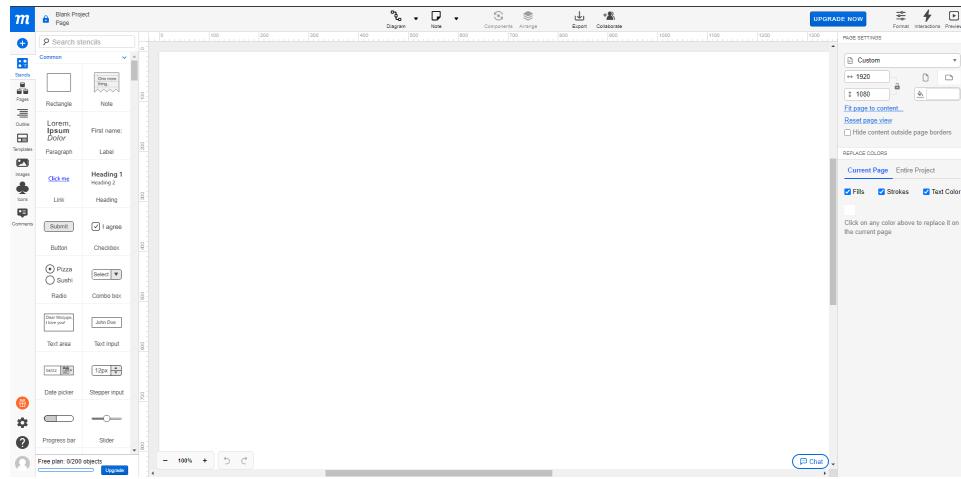


Figura 3.1: Interfaz de Moquups

comentarios, imágenes, iconos, etc). En la parte superior de la página, se pueden crear figuras geométricas y añadir notas, así como poder agregar a otros usuarios, y la posibilidad de exportar el proyecto como una serie de imágenes en formato PNG o PDF; y a formato HTML. Por último, en el menú lateral de la derecha, podemos encontrar el formato de las diferentes páginas (o componentes), y las interacciones disponibles (Figura 3.2).

### 3.2. React

*React*<sup>2</sup> es una librería de *JavaScript*, creada por Facebook y de código abierto, que permite crear interfaces de usuario interactivas de forma sencilla. Está basada en la programación orientada a componentes, donde cada componente se puede ver como una funcionalidad distinta, es decir, como una “pieza” de un puzzle. La ventaja de usar componentes es que, al ser independientes unos de otros, si en la carga de una página web falla uno, no afectaría al resto de componentes, por lo que dicha página quedaría cargada sin ese componente. Así mismo, al usar un DOM (Modelo de Objetos del Documento) virtual, deja que la propia librería actualice las partes que han cambiado, en lugar de actualizar todos los componentes.

La sintaxis que emplea *React* es muy parecida a la sintaxis HTML. Para definir los componentes, se emplean etiquetas definidas por el usuario dentro de código *Javascript*. Esta sintaxis se llama *JSX*. No es obligatorio su uso,

---

<sup>2</sup><https://es.reactjs.org/>

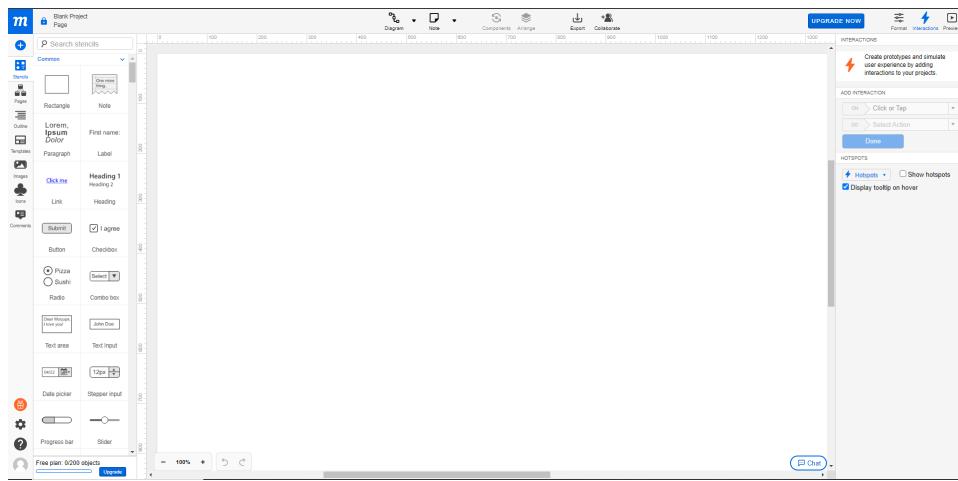


Figura 3.2: Menú lateral de interacciones

pero empleándolo facilita tanto la codificación como la lectura del código. En la Figura 3.3 se puede ver un ejemplo de una funcionalidad sin emplear el formato *JSX*; y en la Figura 3.4, la misma funcionalidad pero usando *JSX*.

*React* aporta rendimiento, flexibilidad y organización de código, frente a la creación de una página web de forma clásica (es decir, sin usar ninguna librería o *framework*). Tiene una documentación bastante completa<sup>3</sup>, junto con un tutorial para aprender desde cero<sup>4</sup>.

<sup>3</sup><https://es.reactjs.org/docs/getting-started.html>

<sup>4</sup><https://es.reactjs.org/tutorial/tutorial.html>

The screenshot shows a dark-themed code editor titled "EDITOR EN VIVO DE JSX". In the top right corner, there is a checkbox labeled "JSX?" which is unchecked. The code in the editor is:

```
class HelloMessage extends React.Component {
  render() {
    return React.createElement(
      "div",
      null,
      "Hola ",
      this.props.name
    );
  }
}

ReactDOM.render(React.createElement(HelloMessage, { name: "Taylor" }),
document.getElementById('hello-example'));
```

Figura 3.3: Funcionalidad sin usar JSX

The screenshot shows a dark-themed code editor titled "EDITOR EN VIVO DE JSX". In the top right corner, there is a checked checkbox labeled "JSX?". The code in the editor is identical to Figure 3.3, but the JSX syntax is now correctly highlighted and rendered:

```
class HelloMessage extends React.Component {
  render() {
    return (
      <div>
        Hola {this.props.name}
      </div>
    );
  }
}

ReactDOM.render(
  <HelloMessage name="Taylor" />,
  document.getElementById('hello-example')
);
```

Figura 3.4: Funcionalidad empleando JSX

## Capítulo 4

# Metodología de desarrollo

**RESUMEN:** En este capítulo se describe la metodología de desarrollo que se usa en el proyecto. En la sección 4.1 se hace una introducción de los tipos de metodologías que existen. En la sección 4.2 se explica la metodología que se ha elegido. Por último, en la sección 4.3 se describen los tipos de pruebas que se realizarán a lo largo del proyecto.

### 4.1. Introducción

La metodología (del griego *metá* “más allá”, *odós* “camino” y *logos* “razón, estudio”) hace referencia al “conjunto de métodos que se siguen en una investigación científica o una exposición doctrinal”<sup>1</sup>. Si este término se lleva al ámbito de la gestión de proyectos, se puede definir como un conjunto de técnicas, herramientas y procedimientos que permite organizar los procesos de un proyecto. Existen dos tipos de metodologías: tradicionales y ágiles. Las metodologías tradicionales tienen un enfoque predictivo, en el que se sigue un proceso secuencial en una dirección y sin marcha atrás. El desarrollo del proyecto (el cual no se subdivide en proyectos más pequeños, se concibe como un único proyecto) se inicia mediante una serie de etapas: captura de requisitos, análisis y diseño/desarrollo. En la primera etapa, captura de requisitos, existe una abundante comunicación con el cliente, al contrario que en las etapas posteriores, en las que apenas existe comunicación (prácticamente nula), debido a que los requisitos se deben quedar fijos desde el principio, por lo que no se esperan cambios en ellos a lo largo del proyecto. Así mismo, la entrega de software se realiza al finalizar el desarrollo, por lo que el cliente

---

<sup>1</sup><https://dle.rae.es/metodología>

solo puede ver el resultado al final. Este tipo de metodología se suele usar cuando se tiene mucha experiencia sobre un determinado producto y tiene el conocimiento necesario para estimarlo (hay un problema conocido y se sabe su solución), o si los requisitos no cambian. También se suele aplicar cuando los proyectos son de cualquier tamaño, y los equipos pueden estar dispersos (no trabajan en un mismo lugar). Algunos ejemplos de metodologías tradicionales son: *ITIL*<sup>2</sup>, *Métrica 3*,<sup>3</sup> *RUP*<sup>4</sup>, etc. En las Figuras 4.1 y 4.2 se ve un ejemplo de un modelo de proceso clásico y un modelo de proceso evolutivo, respectivamente. El primer caso (Figura 4.1) se trata de un modelo en cascada (modelo de proceso clásico), en el se pueden observar siete fases: Captura de requisitos, análisis de requisitos, diseño, implementación, pruebas y lanzamiento. Para pasar a una fase posterior, se debe haber terminado la fase actual (por ejemplo, para hacer la fase de implementación, se deben haber terminado primero las fases anteriores a esta). El segundo caso (Figura 4.2<sup>5</sup>) se trata de un modelo en espiral (modelo de proceso evolutivo), en el que podemos distinguir 6 fases: Comunicación con el cliente, planificación, análisis de riesgos, ingeniería, evaluación del cliente y construcción y entrega. Pasar por cada fase una vez hasta llegar de nuevo a la primera, es una iteración. Cada iteración tiene una etapa distinta, representada con los colores verde, azul, naranja y gris. Siempre se empieza iterando de dentro hacia fuera de la espiral.

Por otro lado, las metodologías ágiles tienen un enfoque adaptativo, en el que cualquier cambio se acoge con normalidad (se espera que ocurran cambios). La razón de que éstos se acojan con normalidad es debido a que esta metodología tiene mecanismos de gestión del cambio, lo que implica un menor esfuerzo adaptarse a éste. Como lo que se pretende es generar valor para el cliente, se necesita un representante de éste, que puede ser un miembro más del equipo, y, sobre todo, una comunicación constante con el mismo. Al contrario que las metodologías tradicionales, el proyecto sí se subdivide en proyectos más pequeños. Estos subproyectos se tratan de forma independiente, cuyo tiempo de desarrollo suele ser corto (aproximadamente entre 2 y 6 semanas), en los que trabajarán equipos no dispersos y pequeños (máximo 10 personas). Uno de los objetivos de las metodologías ágiles es conseguir, lo antes posible, un producto que sea funcional, y que genere valor al cliente, que se logra a través de constantes entregas de software, lo que implica que el cliente pueda proporcionar *feedback* que le permita aumentar dicho valor. Este tipo de metodología funciona bien cuando el entorno va cambiando frecuentemente (los requisitos pueden cambiar semanal o mensualmente) y

---

<sup>2</sup><https://www.heflo.com/es/blog/itil/que-es-metodologia-itil/>

<sup>3</sup>[https://administracionelectronica.gob.es/pae\\_Home/pae\\_Documentacion/pae\\_Metodolog/pae\\_Metrica\\_v](https://administracionelectronica.gob.es/pae_Home/pae_Documentacion/pae_Metodolog/pae_Metrica_v)

<sup>4</sup>[http://ima.udg.edu/\\_sellares/EINF-ES2/Present1011/MetodoPesadesRUP.pdf](http://ima.udg.edu/_sellares/EINF-ES2/Present1011/MetodoPesadesRUP.pdf)

<sup>5</sup><https://www2.deloitte.com/es/es/pages/technology/articles/que-es-el-desarrollo-en-espiral.html>

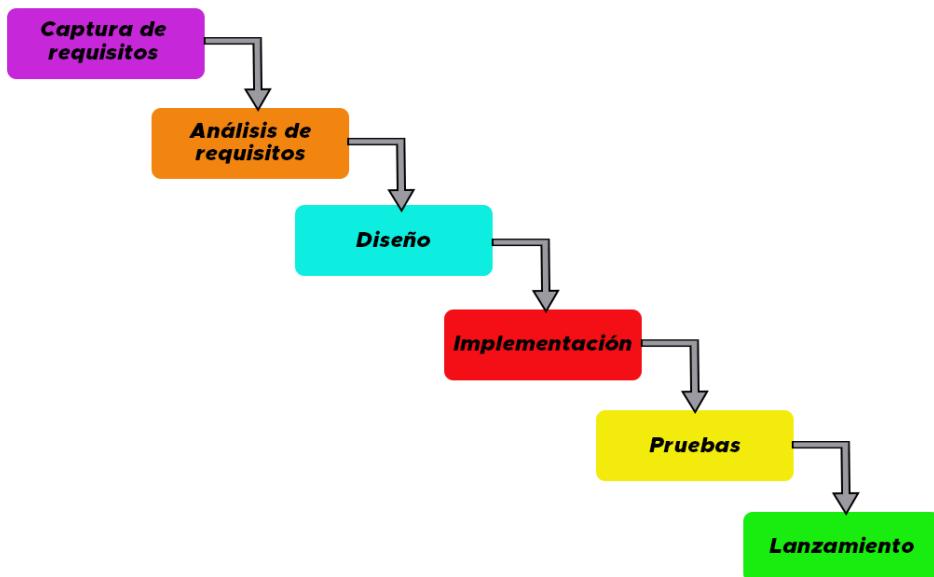


Figura 4.1: Modelo en cascada

no está claro el problema a solucionar ni tampoco de qué manera se puede desarrollar. Algunos ejemplos de metodologías ágiles son: *Scrum*<sup>6</sup>, *Extreme Programming (XP)*<sup>7</sup>, *Kanban*<sup>8</sup>, etc.

Este proyecto seguirá la metodología *Kanban* debido a que es menos prescriptivo que otras metodologías ágiles (tiene solo tres reglas) y da más flexibilidad a la hora de organizar y desarrollar el trabajo, y de adaptarse a los cambios que puedan surgir durante el proyecto. En la siguiente sección se cuenta con más detalle la metodología *Kanban*.

## 4.2. Kanban

El término “*Kanban*” proviene del japonés, cuyo significado es “tarjetas visuales”. Fue creado en la empresa Toyota en la década de los 50 para controlar el avance del trabajo con los materiales disponibles.

A diferencia de otras metodologías ágiles, *Kanban* es menos prescriptivo,

<sup>6</sup><https://www.wearemarketing.com/es/blog/metodologia-scrum-que-es-y-como-funciona.html>

<sup>7</sup><https://openwebinars.net/blog/extreme-programming-que-es-y-como-aplicarlo/>

<sup>8</sup><https://www.illusionstudio.es/metodologia-kanban>

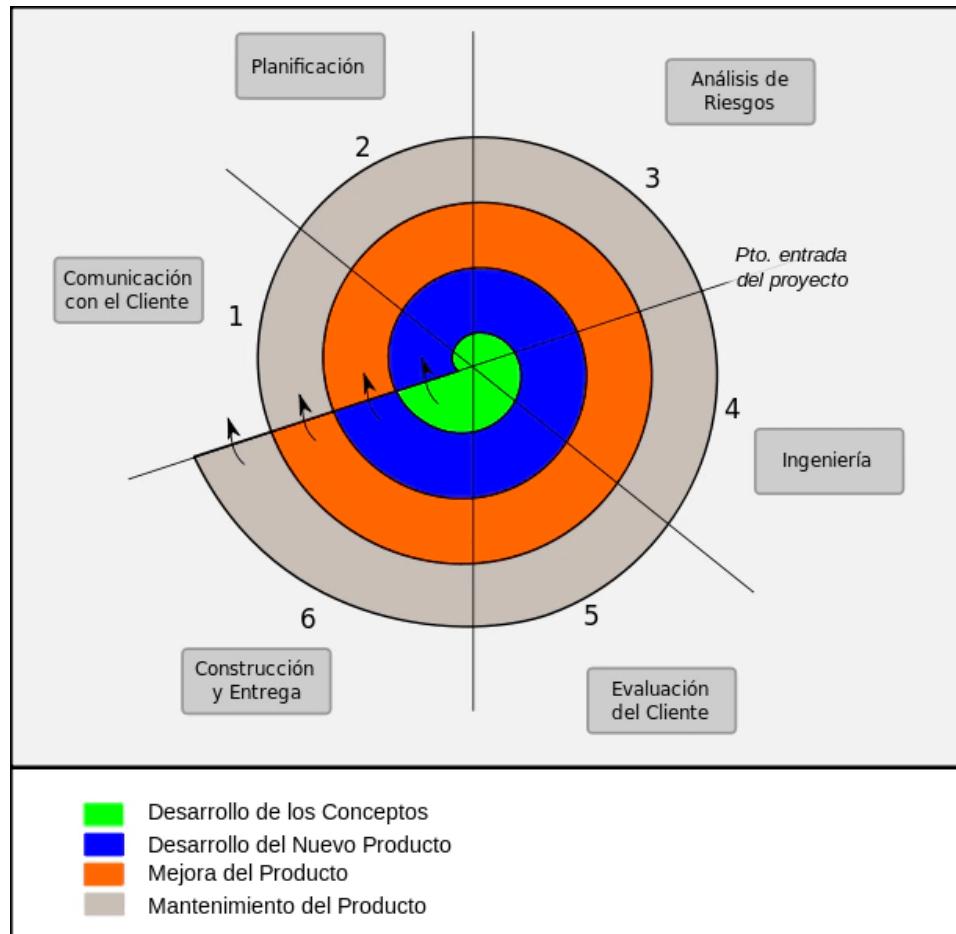


Figura 4.2: Modelo en espiral

es decir, solo tiene tres reglas:

- 1. Visualizar el trabajo y las fases del ciclo de producción o flujo de trabajo:** El trabajo está dividido en partes o tareas. Para visualizar todas estas tareas, se usa una pizarra o tablero llamado “tablero *Kanban*”. Una de las ventajas de usar el tablero es que cualquier persona del equipo puede saber el trabajo realizado y el trabajo pendiente, evita que éste último se acumule y se conoce en qué tarea está trabajando cada uno. La calidad del trabajo y la productividad se ven aumentadas por la mejora del flujo de trabajo en equipo.
- 2. Determinar y respetar el trabajo en curso:** Como se debe saber cuál es el estado del proyecto en cada momento, debe haber un límite máximo de tareas que se pueda realizar en cada fase para evitar cuellos

de botella. Este límite, llamado *Work In Progress* (WIP), debe ser algo conocido y hay varias formas de calcularlo. Lo suele establecer el equipo de desarrollo y se puede cambiar en cualquier momento. El WIP impide comenzar tareas hasta que no se hayan finalizado otras en curso, evitando así la acumulación de tareas.

3. **Medir el tiempo en completar una tarea:** Se pueden distinguir dos tipos de tiempo:

- **Lead Time** o tiempo de entrega: es el tiempo en el que se tarda en completar una tarea, desde que entra en el flujo de trabajo hasta que se realiza su entrega. Este tiempo hay que medirlo siempre.
- **Cycle Time** o tiempo de ciclo: es el tiempo que pasa el equipo trabajando en una tarea, desde que se inicia su desarrollo hasta que se da por terminada.

Gracias a estos tiempos podemos saber la productividad y eficiencia del equipo, y ajustar, en caso necesario, el flujo de trabajo.

Para representar las fases y tareas se usa un tablero *Kanban*. Dicho tablero se divide en varias columnas que representan las distintas fases por las que puede pasar una tarea. Las fases las decide cada equipo, aunque el modelo más común es tener tres columnas:

- *To Do*: En esta lista se tienen las tareas pendientes por realizar.
- *Doing*: Cuando un grupo empieza a trabajar en una tarea, deberá moverla de “*To Do*” a “*Doing*”.
- *Done*: Tras saber que se ha realizado correctamente la tarea, y se ha dado por validada y aprobada, se podrá dar por terminada, por lo que se moverá a “*Done*”. Una vez en esta fase, no se podrá volver a mover a ninguna anterior, ya que el cliente lo ha validado, y le ha generado valor.

En nuestro proyecto, distinguimos dos tipos de tareas:

- Tareas de memoria: Corresponde con tareas de redacción de la memoria.
- Tareas de implementación: Son tareas de diseño y/o desarrollo de código.

En la Figura 4.3 se puede ver nuestro tablero *Kanban* al inicio del proyecto. Este tablero tiene un total de seis columnas:

- *To Do*: Corresponden con las tareas que todavía no se han realizado. Cuando se crea una tarea, se debe crear sus correspondientes tareas de prueba. En la sección 4.3 se da más detalles de los tipos de pruebas que hay.
- *In Progress*: Son tareas en las que el equipo de desarrollo ha comenzado a trabajar en ellas, por lo que se mueve la correspondiente tarea de “*To Do*” a “*In Progress*”.
- *Ready to Testing*: Una vez terminada con una tarea, se debe mover la tarea de prueba de “*To Do*” a “*Ready To Testing*”. Son tareas listas para ser probadas, pero que todavía no se están probando.
- *Testing*: Cuando una persona del equipo de desarrollo tiene asignada alguna tarea de prueba, deberá que moverla de “*Ready To Testing*” a “*Testing*”. Dependiendo del tipo de tarea se tratará de una forma u otra. En la sección 4.3 se da más detalles de cómo se prueban las tareas de memoria y las de implementación.
- *Needs approval*: En esta lista se encuentran las tareas que necesitan ser aprobadas por las tutoras antes de darlas por finalizadas. Una vez que se ha terminado de trabajar en una tarea o se ha acabado de hacerle las pruebas, habrá que moverla de “*In Progress*” a “*Needs Approval*” (en el caso de las tareas de prueba, habrá que moverla de “*Testing*”).
- *Done*: Una vez que las tutoras han dado el visto bueno a esta tarea, se podrá mover de “*Needs Approval*” a “*Done*”, incluyendo las tareas de prueba.

El WIP será de máximo dos tareas por persona, lo que hace un total de seis tareas. Las columnas de “*Ready to Testing*” y “*Needs approval*” no están contempladas debido a que, en el primer caso, aunque las tareas puedan estar asignadas a algún miembro del equipo, no están comenzadas. En el segundo caso, son tareas que ya han sido trabajadas, pero que todavía no se pueden dar finalizadas.

Algunas tareas podrán ser asignadas a varios usuarios en el caso de que ésta sea extensa, o requiera que alguna o todas las partes necesiten hacer lo mismo. En la Figura 4.4, se puede ver un ejemplo de este tipo de tarea, en el que todos los componentes del grupo han tenido que realizar un prototipo, por lo que solo se movería a “*Done*” en el caso de que la lista esté completa.

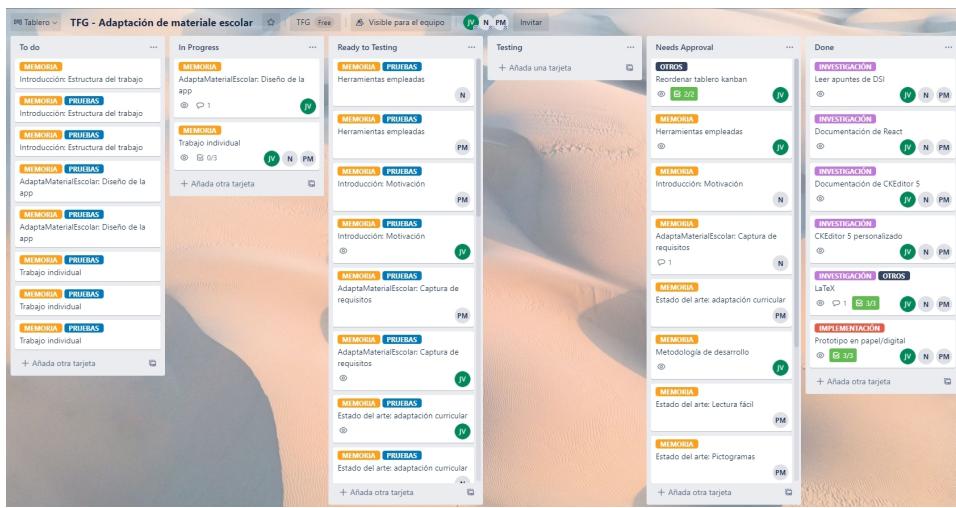


Figura 4.3: Tablero Kanban al inicio del proyecto

Así mismo, en la Figura 4.3 se observa que las tareas tienen asignadas unas etiquetas de colores:

- Tareas relacionadas con la memoria.
- Tareas correspondientes con diseño y desarrollo de código (implementación).
- Tareas que necesitan ser probadas.
- Aquí se encuentran las tareas que requiere investigación, por parte del equipo, antes de empezar a codificar o realizar cualquier otro tipo de tarea.
- Tareas que no corresponden con ninguna de las anteriores.

### 4.3. Tipos de pruebas

Como tenemos dos tipos de tareas, de memoria y de implementación, cada una se tratará de una forma diferente. Tanto en uno como en otro, habrá que tener en cuenta el WIP.

### 4.3.1. Pruebas de memoria

Las pruebas de memoria consiste en buscar errores léxicos y gramaticales en la memoria, y corregirlos antes de su entrega, además de completarlo con más información en el caso de que sea posible. Cada vez que se termina una tarea de memoria, se asignan dos tareas de prueba a los miembros del equipo que no hayan participado en esa tarea, y habrá que moverlas a “*Ready To Testing*”. Por ejemplo, en la figura 4.3, se puede ver como la tarea “Introducción: Motivación” está a la espera de ser aprobada (está en la columna “*Needs approval*”). Eso significa que ese apartado está listo para ser revisado, por lo que hay dos tareas de prueba (que previamente estaban en la columna “*To Do*”), de la tarea redactada, asignada a cada miembro que no ha participado en ese apartado, es decir, si esta tarea la ha realizado Natalia, hay una tarea de prueba asignada a Pablo, y otra asignada a Jorge.

### 4.3.2. Pruebas de implementación

Para las pruebas de implementación, se contempla dos tipos de pruebas: pruebas unitarias y pruebas de integración. Todas estas pruebas las haremos con *Jest*<sup>9</sup>, una librería de testing para *Javascript*, que además es compatible con el *framework* que hemos elegido (*React*). *Jest* tiene una instalación muy sencilla, de pocos pasos, y su configuración es mínima. La documentación es completa, y contiene lo necesario para poder desarrollar estos tipos de pruebas, junto con una serie de ejemplos, realizados paso a paso. Estas pruebas las desarrollará y realizará alguno de los miembros que no haya implementado esa parte del código, y las hará cuando la tarea correspondiente de prueba esté en la columna “*Ready to Testing*”. La razón de esto es porque las personas que no han escrito el código pueden sacar más casos de prueba que las personas que lo han escrito.

#### 4.3.2.1. Pruebas unitarias

Una prueba unitaria se utiliza para comprobar que un método implementado funciona como se esperaba. Debe cumplir una serie de características:

- Deben ser **automáticas**: se deben poder ejecutar sin que haya una intervención manual.

---

<sup>9</sup><https://jestjs.io/es-ES/>

- Deben ser **completas**: es decir, deben cubrir la totalidad del código.
- Deben ser **independientes**: debido a que se ha creado para comprobar una parte concreta del código, no debería interferir con otras partes, y se deben poder ejecutar en cualquier entorno.
- Deben ser **repetibles**: se deben repetir todas las veces que queramos, y el resultado debe ser el mismo en todas.

Ventajas de las pruebas unitarias:

- **Aumento de la calidad** del código: debido a que estas pruebas se ejecutan de forma regular, permite detectar errores a tiempo y poder corregirlos antes de completar el código, y liberar la aplicación.
- **Facilitan los cambios**: se pueden aplicar cambios para mejorar el código, ya que ese cambio solo afectaría a una parte del código. En el caso de que al aplicar el cambio éste no estuviera correctamente realizado, es decir, no hiciera lo que esperase, la prueba unitaria nos avisaría de que hay errores.
- **Reduce los tiempos** de integración: ya que podemos probar partes del código sin disponer del código completo.
- **Reduce el coste**: teniendo en cuenta que permite detectar errores tempranos, los tiempos de entrega mejoran respecto a no usarlos.

#### 4.3.2.2. Pruebas de integración

Una prueba de integración se utiliza para comprobar que dos o más componentes, ya validados por las pruebas unitarias, son compatibles entre sí y funcionan correctamente. Hay varios tipos de pruebas de integración:

- **Top-Down**: Es una estrategia de “arriba a abajo”, se va probando los componentes que no son llamados por ningún otro, y se integran los componentes que son llamados desde la parte integrada.
- **Bottom-Up**: Es una estrategia de “abajo a arriba”, se prueban los componentes que no llaman a otras partes de la aplicación, y se continua por componentes que solo llaman a la parte integrada.

- ***End-to-End***: Es una estrategia orientada al proceso de negocio, en la cual integran los componentes necesarios por parte de un proceso de negocio para ver si el flujo de la aplicación funciona tal y como fue diseñado.
- **Funciones**: Este tipo de integración está orientada a una función del sistema, el cual se va integrando cada uno de los componentes que necesita esa función.
- ***Big-Bang***: Esta estrategia solo es válida cuando se dispone de todos los componentes, es decir, no se integra hasta que el software no esté completo.

The screenshot shows a digital task card titled "Prototipo en papel/digital". The card includes sections for "MIEMBROS" (JV, N, PM), "ETIQUETAS" (IMPLEMENTACIÓN), and "AÑADIR A LA TARJETA" (Miembros, Etiquetas, Checklist, Vencimiento, Adjunto, Portada). The "Checklist" section shows a progress bar at 100% completion with three items: "Prototipo-Pablo", "Prototipo-Natalia", and "Prototipo-Jorge", all marked as completed. There is also a "POWER-UPS" section with an "Actualizar el equipo" button. The "ACCIONES" section includes options like "Mover", "Copiar", "Seguir" (with a checked checkbox), "Archivar", and "Compartir". A comment input field at the bottom left says "Escriba un comentario...".

Figura 4.4: Tarea asignada a varios usuarios



## Capítulo 5

# AdaptaMaterialEscolar

**RESUMEN:** En este capítulo se explicará, en la sección 5.1, para quién va a desarrollarse el proyecto de AdaptaMaterialEscolar, cuándo y cómo se comenzó a planificar y los requisitos para la aplicación web; y, en la sección 5.2, cómo se ha ido diseñando la aplicación.

### 5.1. Captura de requisitos

El Trabajo de Fin de Grado de AdaptaMaterialEscolar es un proyecto sin ánimo de lucro que ha sido desarrollado con la colaboración de las profesoras del Aula TEA del IES Maestro Juan de Ávila de Ciudad Real.

Para su implementación se realizó la captura de requisitos del proyecto, de forma que se pudieron conocer las necesidades reales de un centro educativo, siguiendo un diseño centrado en el usuario.

Se realizaron una serie de reuniones con el usuario para obtener los requisitos de la aplicación.

#### 5.1.1. Primera iteración

El 25 de julio de 2019 se organizó una primera reunión presencial con las dos profesoras que forman parte del Aula TEA en el IES Maestro Juan de

Ávila, Ana María Alonso Frades, especialista en PT (Pedagogía Terapéutica) y Ana María Díaz Valle, especialista en AL (Audición y Lenguaje).

Las profesoras nos hablaron sobre los diferentes perfiles de alumnos TEA con los que trabajaban en el centro y explicaron la importancia de contar con una aplicación especializada en adaptaciones curriculares, que fuera flexible y permitiera generar diferentes modelos de temario, actividades y exámenes con el menor esfuerzo posible. El proyecto debía poder convertir un mismo párrafo o ejercicio en otro más sencillo de comprender, asimilar o resolver, de forma que pudiera personalizarse para cada alumno de acuerdo a su capacidad cognitiva y así facilitarles el aprendizaje sin excluir temario.

A pesar de que hay unas pautas a seguir para adaptar el temario, las actividades y los exámenes, como por ejemplo usar letra grande e imágenes, no hay un estándar definido que sigan todos los profesores y a algunos alumnos les resulta confuso el cambio de una asignatura a otra.

Después de hablar con las profesoras y conocer a un alumno TEA que nos explicó su rutina en el instituto, vimos un ejemplo de tema y examen adaptado de Ciencias Naturales.

Además de la captura de requisitos se tomaron las siguientes decisiones:

- Tipo de aplicación: Inicialmente las profesoras plantearon desarrollar una aplicación de escritorio que pudieran instalar tanto en los ordenadores del centro, como en los suyos propios, pero se tuvo en cuenta que probablemente no todo el profesorado contaría con un ordenador personal con el que poder trabajar desde casa. Por ello, se tomó la decisión de desarrollar el proyecto como aplicación web, de manera que se pudiera utilizar desde cualquier dispositivo de escritorio con acceso a Internet.
- Registro: Las profesoras indicaron que era mejor no tener que registrarse en la aplicación para poder trabajar con ella, ya que se consideró que no era conveniente almacenar usuarios ni contraseñas para evitar que los profesores tuvieran que utilizar datos personales con los que identificarse.

A partir de ahí, como resultado de la reunión, listamos las tareas que debería realizar la aplicación junto con las profesoras, para ser realmente útil y sintetizamos los requisitos necesarios para la realización del proyecto.

Adaptaciones para temario:

La motivación de las adaptaciones de temario es conseguir, en el menor tiempo posible, una unidad didáctica teórica acorde al alumno de forma casi inmediata y automática, seleccionando el texto a convertir.

Los requisitos necesarios para realizar adaptaciones de temario fueron:

- Generar un resumen a partir de un texto.
- Crear esquemas seleccionando contenido de un texto.
- Crear tablas seleccionando contenido de un texto.

Adaptaciones de actividades y/o exámenes:

La motivación de las adaptaciones de actividades y/o temario es conseguir unidades didácticas prácticas o actividades de forma fácil, seleccionando un ejercicio ya redactado y se adapte de forma automática al tipo de ejercicio que se quiere obtener.

Se obtuvieron los siguientes requisitos para realizar adaptaciones de actividades y/o exámenes:

- Ejercicios de relacionar contenido mediante flechas.
- Generar de manera sencilla y automática ejercicios de cualquier tipo seleccionando el texto.
- Ejercicios de sopa de letras.
- Ejercicios de completar espacios en un texto.
- Ejercicios de desarrollo en un espacio limitado para escribir.
- Ejercicios de verdadero o falso.
- Ejercicios de relacionar conceptos con su definición y viceversa.
- Ejercicios de completar los espacios en blanco en tablas.
- Ejercicios de completar los espacios en blanco en esquemas.
- Añadir ejemplos (con o sin pictogramas) explicando cómo debe resolverse el ejercicio.

Adaptaciones de formato:

La motivación de las adaptaciones de formato es conseguir estandarizar cualquier documento que se vaya a entregar al alumno, de manera que el cambio de una asignatura a otra no suponga un cambio visual y el aprendizaje sea menos costoso.

Los requisitos para realizar adaptaciones de formato fueron:

- Sustituir palabras por pictogramas.
- Sustituir palabras por imágenes.
- Resaltar palabras con colores.
- Añadir leyenda de colores con la categoría de cada tipo de palabra resaltada.
- Añadir leyenda de colores para diferenciar las asignaturas.
- Estandarizar formato en textos (tipo de fuente y tamaño).
- Estandarizar formato para títulos e índices del temario.
- Añadir imágenes buscando una palabra.
- Aumentar el tamaño de un texto o palabras clave.
- Subrayar un texto o palabras clave.
- Poner en negrita un texto o palabras clave.

Una vez recopilados los requisitos del proyecto redactamos una lista con éstos para que pudiéramos, tanto las profesoras como el equipo de desarrollo, evaluarlos.

### 5.1.2. Segunda iteración

La segunda reunión consistió en evaluar los requisitos de forma independiente entre el equipo de desarrollo y las profesoras, para no influir en la toma de decisiones sobre la importancia y dificultad de cada tarea.

Enviamos el listado de tareas obtenidas como requisitos a las profesoras por correo electrónico, con la finalidad de que revisaran el contenido o

añadieran nuevas tareas, en caso de necesitarlas, y evaluaran cada una de ellas.

Las profesoras Ana María Alonso Frades y Ana María Díaz Valle debían puntuar los requisitos según la utilidad que pudieran aportar cada una de las tareas a su trabajo. Las posibles puntuaciones debían ir de uno a tres, siendo el uno poco importante, el dos importante y el tres imprescindible.

Prácticamente todas las tareas fueron puntuadas como muy importantes, por lo que quedó claro que el proyecto cubre una necesidad real y no añadieron ningún requisito más.

Cada miembro del equipo de desarrollo puntuó los requisitos de manera individual para no influir en la decisión de los compañeros y según la dificultad que consideraban que tendría la implementación de cada tarea a nivel técnico. Las posibles puntuaciones debían ir de uno a tres, siendo el uno difícil, el dos intermedio y el tres fácil.

La finalidad de evaluar por separado los requisitos era obtener de forma aproximada un listado con las tareas más importantes que debían realizarse con prioridad para aportar valor real al proyecto, maximizando su importancia y minimizando su dificultad.

Con la puntuación que asignaron las profesoras se obtuvo el indicador de importancia de cada tarea y, con la media de las puntuaciones del equipo de desarrollo, el indicador dificultad que se esperaba durante la implementación del requisito.

El resultado obtenido se separó en dos tablas. La primera contiene las funcionalidades de la aplicación, las adaptaciones de temario (T) y actividades y/o exámenes (A), y la segunda, contiene las posibles opciones de personalización que serían necesarias en el proyecto, las adaptaciones de formato (F).

Las tablas resultantes ordena, de forma descendente, las puntuaciones obtenidas entre la multiplicación de los indicadores de importancia y dificultad, siendo 9 la máxima calificación y 1 la mínima.

Para establecer un orden en el desarrollo de los requisitos, se considera más importante la implementación de las tareas de la tabla de prioridades de las funcionalidades de la aplicación.

GRUPO	1	2	3
Profesoras Aula TEA	Poco importante	Importante	Imprescindible
Equipo de desarrollo	Difícil	Intermedio	Fácil

Tabla 5.1: Leyenda de puntuaciones.

ADAPTACIONES DE TEMARIO (T)		Evaluación	
Requisito		Importancia	Dificultad
Generar un resumen a partir de un texto		3	2
Crear esquemas seleccionando contenido de un texto		3	1,7
Crear tablas seleccionando contenido de un texto		2	1,7
ADAPTACIONES DE ACTIVIDADES Y/O TEMARIO (A)		Evaluación	
Requisito		Importancia	Dificultad
Ejercicios de relacionar contenido mediante flechas		3	2
Ejercicios de sopa de letras		2	3
Ejercicios de completar espacios en un texto		3	2,7
Ejercicios de desarrollo en un espacio limitado para escribir		3	2,7
Ejercicios de verdadero o falso		3	3
Ejercicios de relacionar conceptos con su definición		3	2,3
Ejercicios de relacionar definiciones con su concepto		3	2
Ejercicios de completar los espacios en blanco en tablas		3	2
Ejercicios de completar los espacios en blanco en esquemas		3	1
Añadir ejemplos (con o sin pictogramas) sobre cómo resolver el ejercicio		2	3
ADAPTACIONES DE FORMATO (F)		Evaluación	
Requisito		Importancia	Dificultad
Sustituir palabras por pictogramas		2	2,7
Sustituir palabras por imágenes		2	2,3
Resaltar palabras con colores		3	3
Añadir leyenda de colores con la categoría de cada tipo de palabra resaltada		1	3
Añadir leyenda de colores para diferenciar las asignaturas		1	3
Estandarizar formato en textos (tipo de fuente y tamaño)		3	2
Estandarizar formato para títulos e índices del temario		3	2
Añadir imágenes buscando una palabra		3	2,3
Aumentar el tamaño de un texto o palabras clave		3	2,7
Subrayar un texto o palabras clave		2	2,7
Poner en negrita un texto o palabras clave		3	2,7

Tabla 5.2: Puntuación de los requisitos I.

Tipo de requisito	Requisito	Puntuación
A	Ejercicios de verdadero o falso	9
A	Ejercicios de completar espacios en un texto	8,1
A	Ejercicios de desarrollo en un espacio limitado para escribir	8,1
A	Ejercicios de relacionar conceptos con su definición	6,9
T	Generar un resumen a partir de un texto	6
A	Ejercicios de relacionar contenido mediante flechas	6
A	Ejercicios de sopas de letras	6
A	Ejercicios de relacionar definiciones con su concepto	6
A	Añadir ejemplos (con o sin pictogramas) sobre cómo resolver el ejercicio	11
A	Ejercicios de completar los espacios en blanco en tablas	6
T	Crear esquemas seleccionando contenido de un texto	5,1
T	Crear tablas seleccionando contenido de un texto	3,4
A	Ejercicios de completar los espacios en blanco en esquemas	3

Tabla 5.3: Lista de funcionalidades ordenada por prioridad.

Tipo de requisito	Requisito	Puntuación
F	Resaltar palabras con colores	9
F	Aumentar el tamaño de un texto o palabras clave	8,1
F	Poner en negrita un texto o palabras clave	8,1
F	Añadir imágenes buscando una palabra	6,9
F	Añadir ejemplos (con o sin pictogramas) sobre cómo resolver el ejercicio	6
F	Estandarizar formato en textos (tipo de fuente y tamaño)	6
F	Estandarizar formato para títulos e índices del temario, para diferenciarlos de texto normal	6
F	Sustituir palabras por pictogramas	5,4
F	Subrayar un texto o palabras clave	5,4
F	Añadir leyenda de colores con la categoría de cada tipo de palabra resaltada	3

Tabla 5.4: Lista de opciones de personalización ordenada por prioridad.

ADAPTACIONES DE FORMATO (F)		Evaluación	
Requisito		Importancia	Dificultad
Añadir espacio extra entre líneas para escribir		3	2

Tabla 5.5: Puntuación de los requisitos II.

### 5.1.3. Tercera iteración

El miércoles 4 de diciembre de 2019 NIL (Natural Interaction based on Language) organizó un congreso en la Facultad de Informática de la Universidad Complutense con docentes de otros centros y asociaciones, para mostrarles las herramientas tecnológicas inclusivas que se han desarrollado en los últimos años para personas con discapacidad, siendo una de ellas nuestra aplicación web.

Un miembro del equipo de desarrollo, Jorge Velasco Conde, realizó una presentación en la que explicó a los asistentes el funcionamiento y la finalidad de nuestro proyecto, con ejemplos visuales sobre posibles adaptaciones que podrían llegar a realizarse.

Después de la presentación se realizó una ronda de opiniones con todos los docentes, para que pudieran expresar qué les había parecido la aplicación web. Todos los comentarios fueron muy positivos y preguntaron cuándo estaría disponible para uso público.

Además de dar “feedback” positivo sobre el proyecto, también propusieron un tipo de adaptación de formato (Tabla 5.5), para mejorar las opciones de personalización para los alumnos.

La motivación de este requisito es facilitar a los alumnos con letra de mayor tamaño que puedan escribir en un espacio suficientemente grande para ellos.

En este caso únicamente realizamos la evaluación del requisito los desarrolladores, ya que, al ser una petición, se consideró que el indicador de importancia debía tener la máxima puntuación, para poder aportar utilidad a un mayor número de centros y asociaciones y el indicador de dificultad volvió a calcularse de forma independiente entre el equipo, haciendo la media entre las puntuaciones.

Tipo de requisito	Requisito	Puntuación
T	Añadir espacio extra entre líneas para escribir	6

Tabla 5.6: Lista de requisitos ordenada por prioridad.

## 5.2. Diseño de la aplicación

El diseño de la aplicación web se ha desarrollado en varias iteraciones o etapas.

### 5.2.1. Primera etapa

En esta primera etapa se creó un diseño, en papel, de cómo podría ser la página principal de la aplicación web (figura 5.1). En la parte superior de este diseño, se observa que hay una barra de navegación (*navbar* en inglés) con dos menús: “Texto” y “Actividades”. Si se pasa el ratón por encima de alguno de estos menús, aparece un submenú con las diferentes adaptaciones disponibles para ese menú (en la sección 5.1.1 se pueden ver las diferentes adaptaciones que hay en la aplicación). En la lado derecho encontramos otro menú, con dos zonas diferenciadas: en la parte superior tenemos los botones de “Subir texto” (que sube un fichero en formato *.pdf* o *.docx*) y “Descargar” (que descarga el documento adaptado en formato *.pdf*). En la parte inferior del menú lateral, nos encontramos un “Resaltar información”, el cual sirve para dar formato a los títulos, al cuerpo, etc. Por último, en el centro de la página, se encuentra el fichero subido con el que podremos seleccionar textos y poder adaptarlos conforme así se requiera.

### 5.2.2. Segunda etapa

Partiendo del diseño inicial, y a través de la herramienta “*Moqups*” (ver sección 3.1), se creó un prototipo más amplio por vistas:

- **Página principal:** En la figura 5.2 se puede ver la página principal de la aplicación. Ésta es la página que se vería nada más entrar. En la parte superior, encontramos dos *navbar* con diferentes menús. En el *navbar* superior, podemos ver los menús relacionados con un editor de texto (es decir, tienen una funcionalidad similar a los que tendría, por

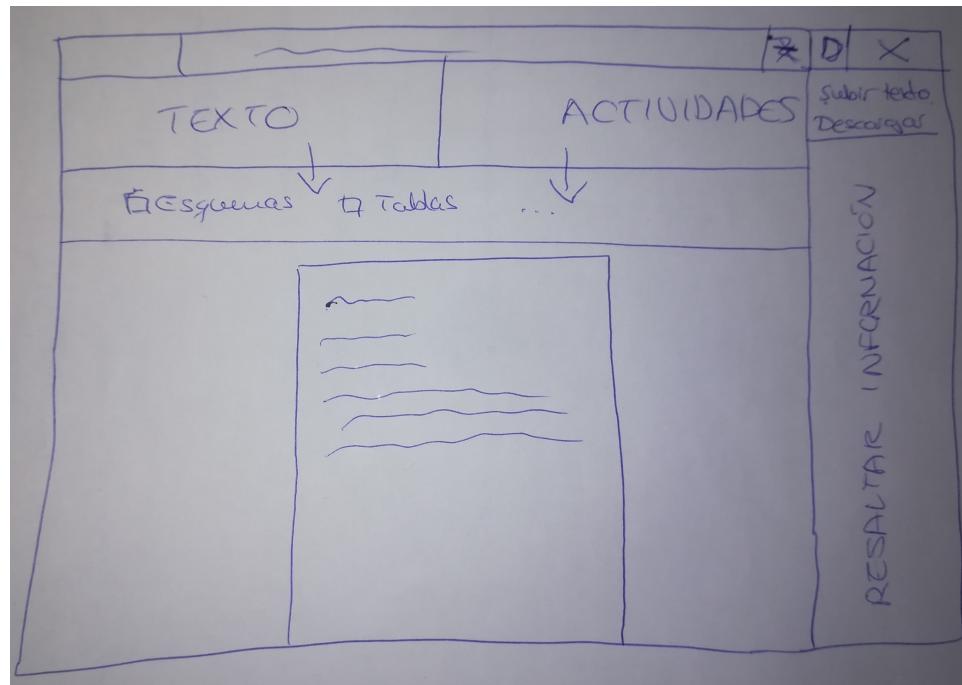


Figura 5.1: Boceto inicial de la aplicación

ejemplo, *Microsoft Word*<sup>1</sup>): “Archivo”, “Edición”, “Formato”, “Insertar” y “Ayuda”. En el navbar inferior, se encuentra los menús correspondientes a las adaptaciones de “Temario” y “Actividades”, además de la posibilidad de buscar imágenes libres<sup>2</sup> (también llamadas imágenes *CC0* o *Creative Commons Zero*<sup>3</sup>) y pictogramas. En el centro de la página web, podemos observar dos elementos: el logo de la aplicación “AdaptaMaterialEscolar” y una zona para poder subir el fichero en formato *.pdf* o *.docx*, ya sea seleccionándolo de una carpeta o arrastrando desde la carpeta a la zona de subida.

- **Editor:** Una vez subido el fichero, se pasa a la vista del editor. En la figura 5.3 se puede observar dicha vista. En ésta, encontramos, en la parte superior, las dos barras de navegación descritas en el punto anterior. En el centro de la página, observamos dos elementos: el elemento de la izquierda es el fichero subido, donde se podrá seleccionar las partes del texto que se quiera adaptar. El elemento de la derecha es el editor donde se podrá, también, adaptar e insertar imágenes,

<sup>1</sup><https://www.microsoft.com/es-es/microsoft-365/word>

<sup>2</sup>Las imágenes libres son aquellas imágenes que son de dominio público y que no tienen derechos reservados.

<sup>3</sup><https://creativecommons.org/publicdomain/zero/1.0/deed.es>



Figura 5.2: Versión 1.0 de la página principal

pictogramas, dar formato, etc.

- **Búsqueda de pictogramas:** En la figura 5.4 se puede ver la búsqueda de pictogramas. Se observa que se sigue en la página del editor, solo que, en la parte superior derecha de la aplicación, aparece una ventana con los diferentes pictogramas, resultado de buscar la palabra “Perro”. Previamente, para poder realizar la búsqueda de pictogramas, se ha tenido que seleccionar el campo “Pictos”, que está a la izquierda de la barra de búsqueda.
- **Adaptación de actividades:** En la figura 5.5 se puede ver un ejemplo de una adaptación de una actividad. En este caso, el ejercicio es “Completar”, es decir, completar el texto rellenando los huecos en blanco. Para ello, se ha tenido que seleccionar el menú “Actividades” (figura 5.3), y, de entre los distintos tipos de actividades que hay, seleccionar “completar”. Una vez elegido este apartado, habría que ir señalando, en la parte de la izquierda, las palabras en las que se quiera dejar el hueco, por lo que en la parte de la derecha (editor) saldría huecos en blanco en vez de la palabra señalada. Si se quiere cerrar este menú, bastaría con darle clic en la flecha que hay en la parte de la izquierda del menú.

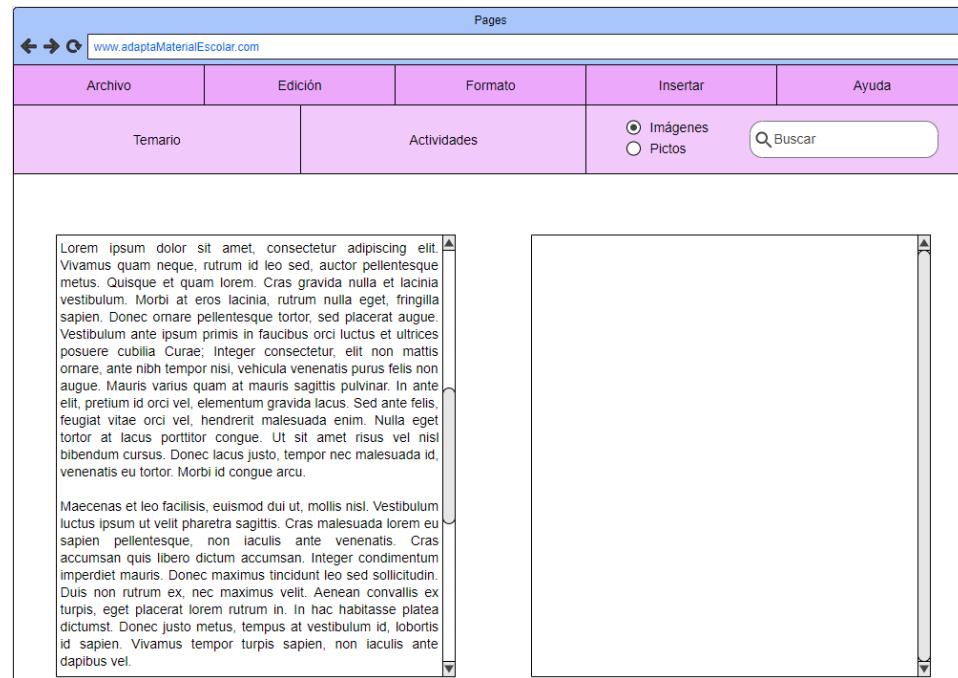


Figura 5.3: Versión 1.0 del editor

- **Adaptación de temario:** En la figura 5.6 se puede encontrar un ejemplo de una adaptación de temario. En este caso, la adaptación es “Resumir texto”. Para ello, se ha tenido que seleccionar el menú “Temario” (figura 5.3). Una vez elegido este apartado, lo primero es seleccionar el texto que se quiera resumir, en la parte izquierda, y después, habría que darle clic a “Resumir texto” para que salga, en la parte de la derecha, el texto resumido.

### 5.2.3. Tercera etapa

Una vez que se hizo la versión 1.0 del prototipo, se propuso hacer una iteración competitiva para ver si este prototipo se podía mejorar. Cada miembro del equipo realizó, de forma individual y sin influir los unos en los otros, un prototipo propio partiendo del inicial. Con esto se consigue tener diferentes puntos de vista sobre una misma funcionalidad. Una vez que todos los prototipos fueron hechos, se realizó una puesta en común para discutir cada uno, y sobre todo, el por qué de esa distribución. Los resultados fueron los siguientes:

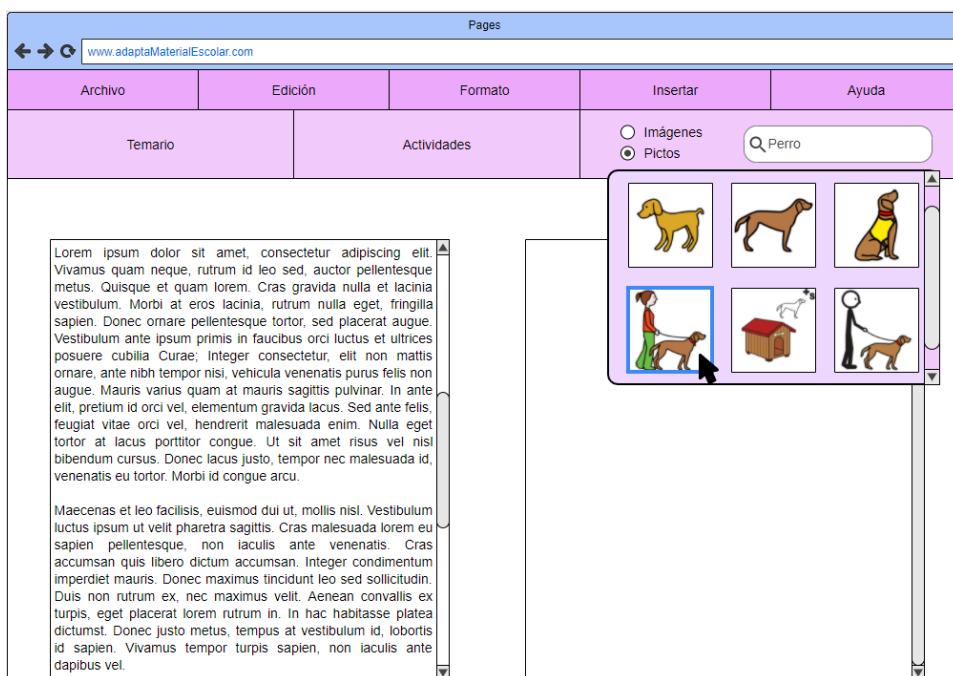


Figura 5.4: Versión 1.0 de la búsqueda de pictogramas

- **Página principal:** En los tres diseños realizados, la página principal era prácticamente igual. Todos coincidimos en la manera de distribuir dicha página: Un *header* con el logo de “AdaptaMaterialEscolar” en la parte de la izquierda; y en la parte de la derecha algún sistema que fuera útil para la aplicación. En el centro de la página estaría la funcionalidad de poner subir un fichero a la aplicación. En las figuras 5.7, 5.8 y 5.9 se puede observar los resultados de los tres prototipos para la página principal. En el caso de la figura 5.7, en la parte inferior (*footer*) están los créditos y los logotipos de ARASAAC<sup>4</sup>, la Universidad Complutense de Madrid<sup>5</sup> y la licencia de *Creative Commons*<sup>6</sup>. También se observa que en la parte de la derecha de la barra de navegación se propuso hacer un sistema de *login* para poder guardar el trabajo realizado. Ésto último no resultó debido a que los usuarios finales que usen la aplicación especificaron que no querían recordar ningún nombre de usuario y contraseña. También, la parte inferior de dicha figura, coincidimos que sería buena idea establecerla en un botón, tal y como ocurre en las figuras 5.8 (aquí se observa que hay dos botones: “Créditos”, que es donde iría la parte inferior de la figura 5.7; y “Ayuda”, por si el usu-

<sup>4</sup><https://arasaac.org/>

<sup>5</sup><https://www.ucm.es>

<sup>6</sup><https://creativecommons.org/>

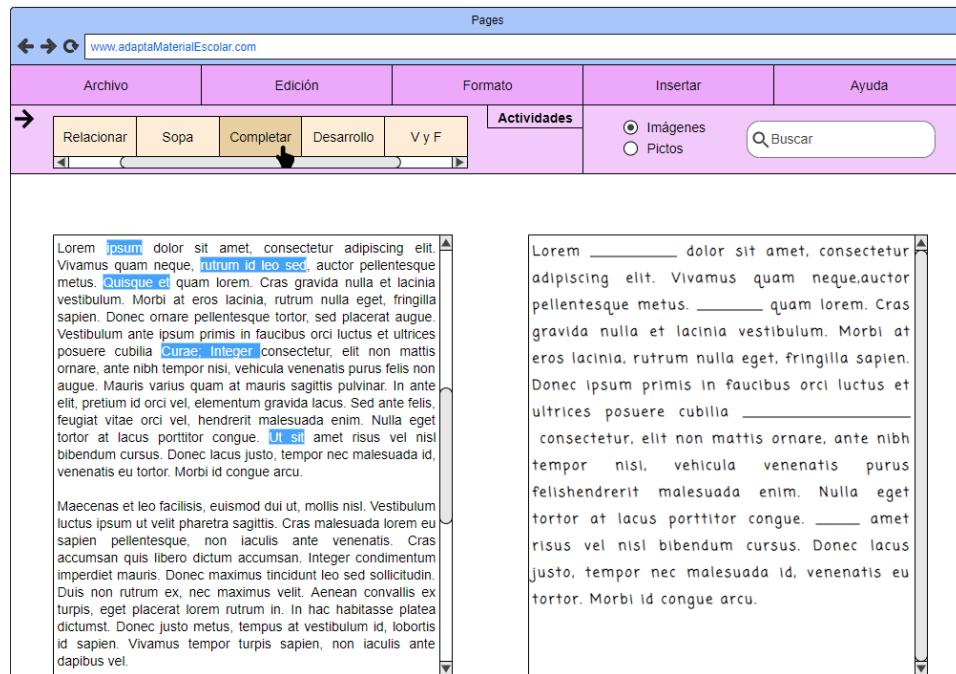


Figura 5.5: Versión 1.0 de la adaptación de actividades

rio necesita algún tipo de ayuda con la aplicación) y 5.9 (en este caso se ve tres botones: “Inicio”, que te llevaría a la página principal; “Ayuda”, donde se incluiría también la parte de “Créditos”; y “Contacto”, por si el usuario tiene algún problema y/o sugerencia pueda contactar con los desarrolladores y proporcionarle una respuesta).

- **Editor:** En el caso del editor, los tres prototipos eran bastante similares. La forma en cómo se ha distribuido el editor y el fichero subido era igual; en la mitad izquierda uno, y en la mitad derecha el otro. En las figuras 5.10, 5.13 y 5.14 se pueden ver los resultados de la página del editor. En el caso de la figura 5.10, observamos una barra de navegación con diferentes menús, que representa los tipos de adaptaciones que hacemos en la aplicación (véase sección 5.1.1), junto con la búsqueda de imágenes y pictogramas. Si se pasa encima del ratón por uno de estos menús, saldrá un desplegable con los distintos tipos de actividades, temario y formato que hay, tal y como se puede ver en la figura 5.11. Así mismo, se ve dos elementos separados por un hueco: el elemento de la izquierda representa el fichero subido, y el de la derecha, el editor. Debajo de éste se encuentra un panel con unos botones para poder descargar, guardar y ver el documento editado, así como la posibilidad de poder borrar todo lo escrito en el editor (“Vista Previa”>,

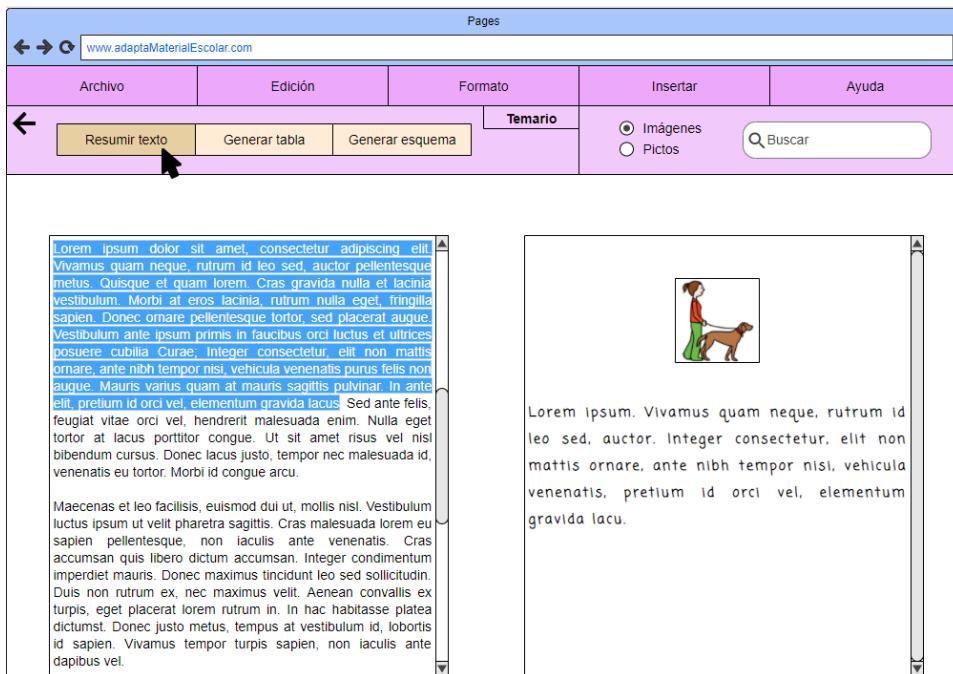


Figura 5.6: Versión 1.0 de la adaptación de temario

que se puede ver en la figura 5.12, donde la opacidad del fondo está ligeramente oscurecida, y, en el centro, cómo quedaría lo escrito en el editor; “Descargar”, “Guardar”, “Borrar todo”). En la figura 5.13, también existe esa barra de navegación con esos menús, aunque se cambia el logo de “AdaptaMaterialEscolar” por dichos menús, añadiendo el de “Archivo” y “Formato”. También tiene los dos elementos mencionados anteriormente: en la parte de la izquierda el documento subido, y en el de la derecha, el editor. En éste, se encuentra una barra de herramientas con la posibilidad de cambiar el formato de la letra, la posición del texto, y dos botones para descargar y borrar todo (representados por una cuadrado con una flecha apuntando hacia abajo, “Descargar”; y una papelera con una cruz en ella, “Borrar todo”). En cambio, en la figura 5.14, se observa que esa barra de navegación no está, sino que está incluida en el propio editor (el elemento que se encuentra en la parte de la izquierda). En esta barra de herramientas se podrá dar formato al texto (ponerle forma de título, párrafo, color, negrita, cursiva, etc), así como deshacer y rehacer lo escrito, crear tablas, y unos botones con los distintos tipos de adaptaciones que hay.

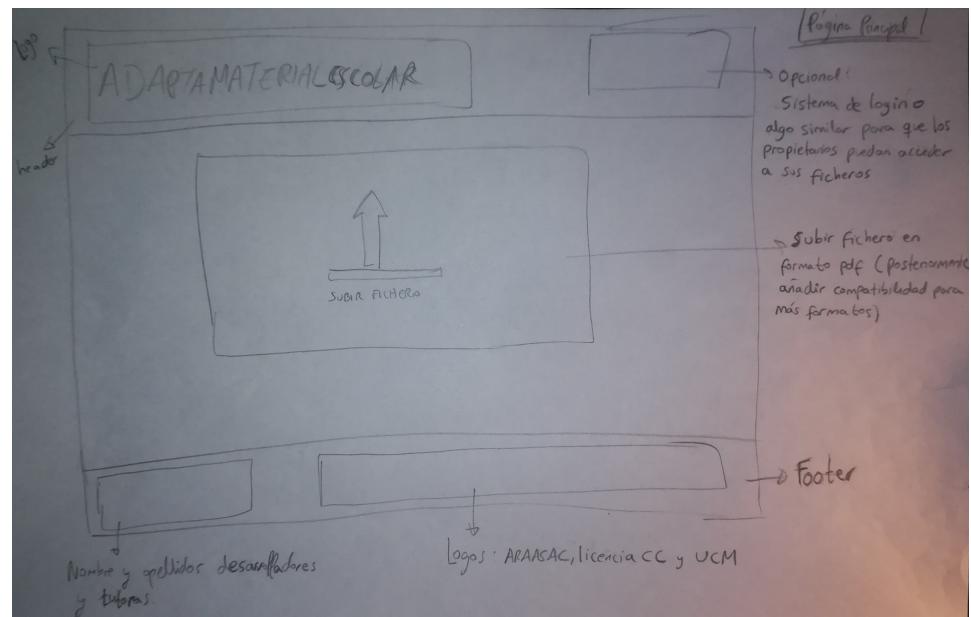


Figura 5.7: Prototipo de la página principal (Jorge)



Figura 5.8: Prototipo de la página principal (Natalia)



Figura 5.9: Prototipo de la página principal (Pablo)

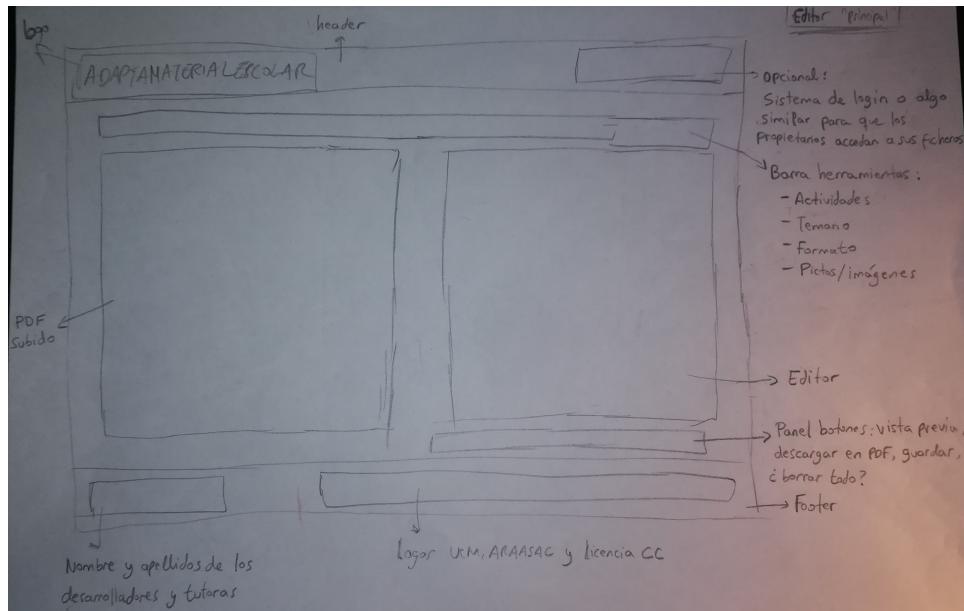


Figura 5.10: Prototipo del editor (Jorge)

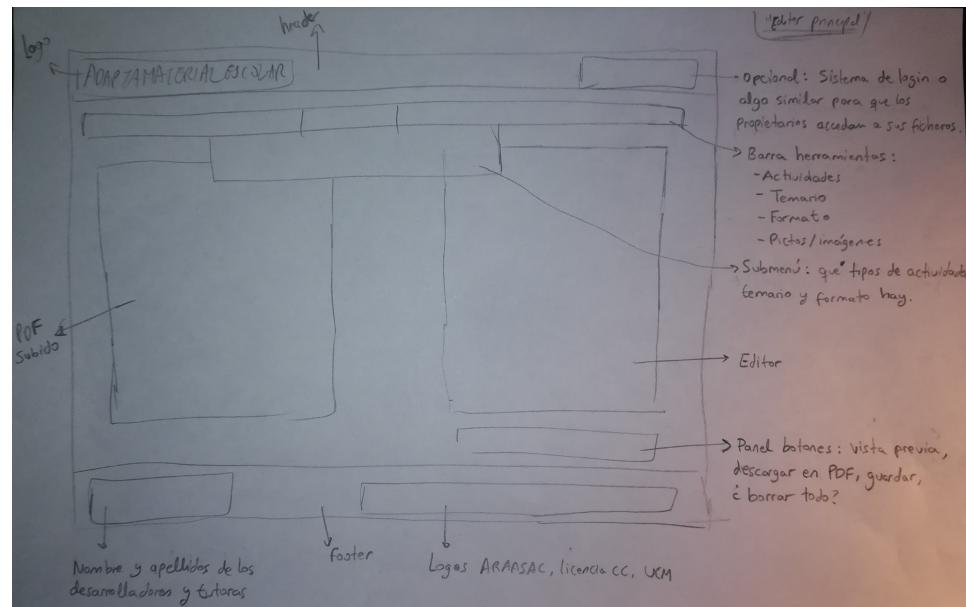


Figura 5.11: Prototipo del editor con un desplegable en uno de los menús (Jorge)

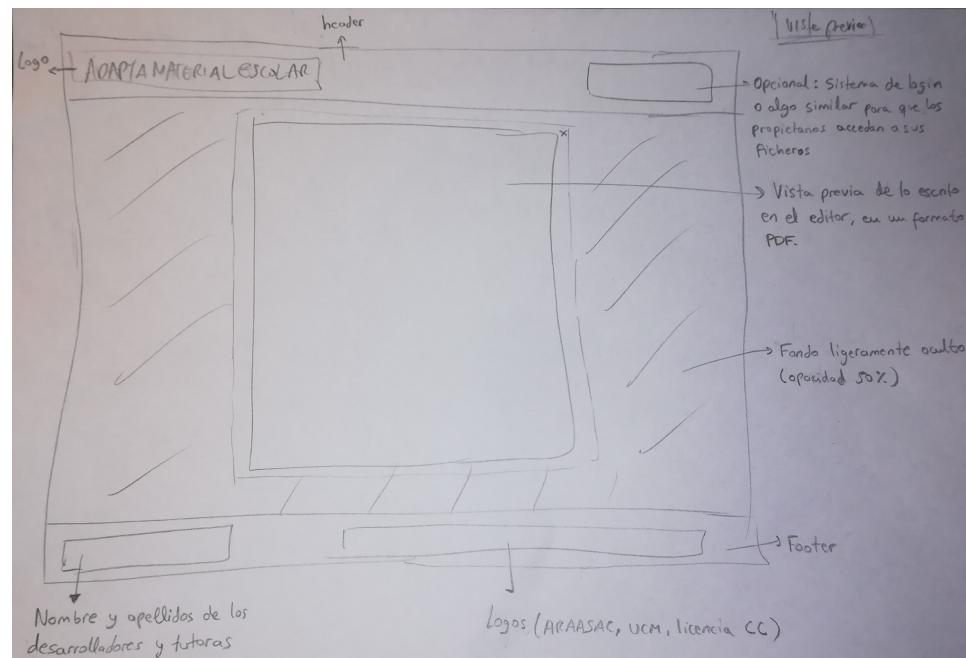


Figura 5.12: Prototipo de la vista previa (Jorge)



Figura 5.13: Prototipo del editor (Natalia)

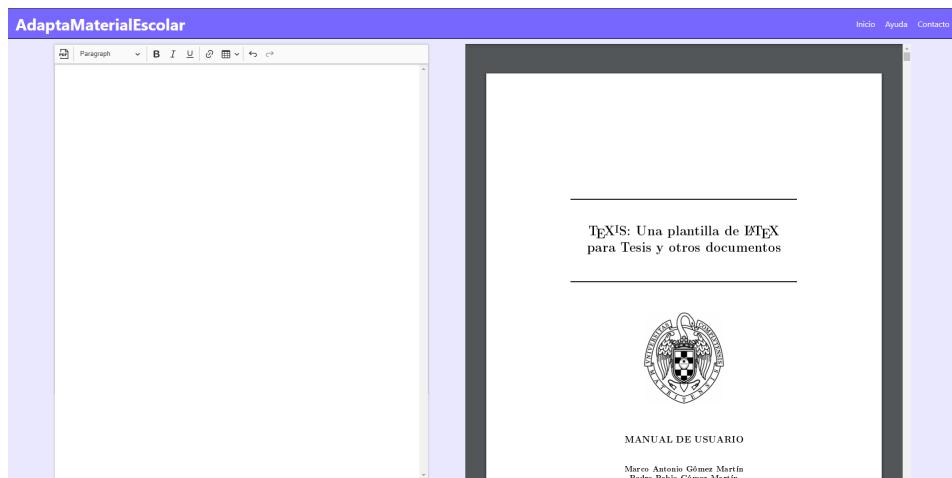


Figura 5.14: Prototipo del editor (Pablo)



## Capítulo 6

# Trabajo Individual

**RESUMEN:** En este capítulo se habla del trabajo individual que ha realizado cada miembro del equipo en el proyecto.

6.1. Pablo

6.2. Natalia

6.3. Jorge



# Bibliografía

APIUMHUB. Beneficios de las pruebas unitarias. <https://apiumhub.com/es/tech-blog-barcelona/beneficios-de-las-pruebas-unitarias/>, 2017.

CADAVID, A. N., MARTÍNEZ, J. D. F. y VÉLEZ, J. M. Revisión de metodologías ágiles para el desarrollo de software. *Prospectiva*, vol. 11, páginas 30–32, 2013. Disponible en <http://ojs.uac.edu.co/index.php/prospectiva/article/view/36>.

DEMERA, R. Metodologías... ¿tradicional vs ágil? <https://tech.tribalyte.eu/blog-metodologias-tradicional-vs-agil>, 2018.

GARZAS, J. Kanban. <https://www.javiergarzas.com/2011/11/kanban.html>, 2011.

HERRERA, M. O. *Métodos y técnicas para la gestión de proyectos software*. Trabajo de Fin de Máster, Universidad de Sevilla, 2010.

IONOS. Kanban. <https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/que-es-kanban/>, 2019.

KANBANIZE. Kanban: Tiempo de entrega vs. tiempo de ciclo. <https://kanbanize.com/es/recursos-de-kanban/software-kanban/tiempo-de-entrega-vs-tiempo-de-ciclo>, 2020.

MESH, J. Metodología kanban: revoluciona tu manera de trabajar más ágil. <https://blog.trello.com/es/metodologia-kanban>, 2020.

PRESSMAN, R. S. *Ingeniería del Software. Un enfoque práctico*. McGraw-Hill, 7<sup>a</sup> edición, 2010. Disponible en [https://tesuva.edu.co/phocadownload/Ingenieria\\_del\\_Software.\\_Un\\_Enfoque\\_Practico.pdf](https://tesuva.edu.co/phocadownload/Ingenieria_del_Software._Un_Enfoque_Practico.pdf).

RODRÍGUEZ, N. G. *Las Pruebas de Integración como Proceso de la Calidad del Software en el Ámbito de las Telecomunicaciones*. Proyecto de Fin de

Carrera, Escuela Politécnica Superior Carlos III (Universidad Carlos III de Madrid), 2015.

SOMMERVILLE, I. *Ingeniería del Software*. Addison-Wesley, 7<sup>a</sup> edición, 2005. Disponible en <https://emtinfoada.files.wordpress.com/2015/03/ingenieria-del-software-ian-sommerville.pdf>.