

---

# **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

## **Directoras**

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 las Doctoras*  
**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 del proyecto . . . . .	3
<b>2. Estado del Arte</b>	<b>5</b>
2.1. Adaptación Curricular . . . . .	5
2.1.1. Lectura fácil para adaptación curricular . . . . .	7
2.1.2. Pictogramas para adaptación curricular . . . . .	9
2.2. Herramientas existentes para adaptaciones curriculares . . . . .	11
2.3. Diseño Centrado en el Usuario . . . . .	13
2.3.1. Conocer al usuario . . . . .	15
2.3.2. Prototipado . . . . .	16
2.3.3. Evaluación . . . . .	18
<b>3. Herramientas empleadas</b>	<b>19</b>
3.1. Moqups . . . . .	19
3.2. React . . . . .	21
3.3. API de ARASAAC . . . . .	23
3.4. Jest . . . . .	24
3.5. CKEditor . . . . .	27
3.6. Word-search . . . . .	28
<b>4. Metodología de desarrollo</b>	<b>33</b>
4.1. Introducción . . . . .	33
4.2. Kanban . . . . .	34
4.3. Tipos de pruebas . . . . .	37
4.3.1. Pruebas de memoria . . . . .	37
4.3.2. Pruebas de implementación . . . . .	38
<b>5. AdaptaMaterialEscolar</b>	<b>41</b>
5.1. Requisitos de la aplicación . . . . .	41

5.1.1. Captura de requisitos . . . . .	42
5.1.2. Análisis de requisitos . . . . .	44
5.1.3. Requisitos adicionales . . . . .	45
5.2. Diseño de la aplicación . . . . .	48
5.2.1. Primera iteración: boceto inicial . . . . .	48
5.2.2. Segunda iteración: versión 1.0 del prototipo . . . . .	49
5.2.3. Tercera iteración: iteración competitiva . . . . .	52
5.2.4. Cuarta iteración: diseño final . . . . .	58
5.3. Implementación . . . . .	58
5.3.1. Arquitectura . . . . .	60
5.3.2. Funcionalidades . . . . .	65
5.3.3. Estructura del código . . . . .	76
<b>6. Evaluación de AdaptaMaterialEscolar</b>	<b>81</b>
6.1. Función de la evaluación . . . . .	81
6.2. Prueba con el usuario final . . . . .	86
<b>7. Trabajo Individual</b>	<b>89</b>
7.1. Natalia . . . . .	89
7.2. Jorge . . . . .	91
7.3. Pablo . . . . .	92
<b>Bibliografía</b>	<b>95</b>

# Índice de figuras

2.1. Ejemplo de documento adaptado a lectura fácil. En este caso, un documento jurídico . . . . .	8
2.2. Ejemplo de pictograma de una rosa . . . . .	9
2.3. Ejemplo de pictograma que representa la acción de comer . .	9
2.4. Ejemplo de pictogramas del Sistema Pictográfico de Comunicación (SPC) . . . . .	10
2.5. Ejemplo de sistema pictográfico Minspeak . . . . .	11
2.6. Interfaz de ARAWORD . . . . .	12
2.7. Ventana de generación de ejercicio de asociar conceptos con EasyTestMaker . . . . .	13
2.8. Ejemplo de examen generado con EasyTestMaker . . . . .	13
2.9. Interfaz y funcionamiento de Resoomer . . . . .	14
2.10. Interfaz de Canva para la creación de un esquema usando una plantilla . . . . .	14
2.11. Ejemplo de guión gráfico . . . . .	16
2.12. Ejemplo de boceto en papel . . . . .	17
2.13. Ejemplo de prototipo en papel interactivo . . . . .	18
3.1. Interfaz de Moqups . . . . .	20
3.2. Menú superior . . . . .	20
3.3. Menú lateral izquierda . . . . .	21
3.4. Menú lateral derecha (formato) . . . . .	21
3.5. Menú lateral derecha (interacciones) . . . . .	21
3.6. Funcionalidad sin usar JSX . . . . .	22
3.7. Funcionalidad empleando JSX . . . . .	22
3.8. Test pasado correctamente . . . . .	26
3.9. Test no pasado correctamente . . . . .	26
3.10. Ejemplo de CKEditor 5 . . . . .	28
3.11. Ejemplo de resultado de sopa de letras usando grid() . . . .	30
3.12. Ejemplo de resultado de sopa de letras usando toString() . .	31
3.13. Ejemplo de las palabras insertadas al usar words() . . . . .	31

3.14. Ejemplo del objeto resultante al usar dump() . . . . .	32
3.15. Ejemplo de llamada a la función read() . . . . .	32
3.16. Ejemplo del string formado cuando se usan los parámetros de la Figura 3.15 . . . . .	32
 4.1. Tablero <i>Kanban</i> . . . . .	36
 5.1. Boceto inicial de la aplicación . . . . .	49
5.2. Versión 1.0 de la página principal . . . . .	50
5.3. Versión 1.0 del editor . . . . .	51
5.4. Versión 1.0 de la búsqueda de pictogramas . . . . .	51
5.5. Versión 1.0 de la adaptación de actividades . . . . .	52
5.6. Versión 1.0 de la adaptación de temario . . . . .	53
5.7. Prototipo de la página principal (Jorge) . . . . .	54
5.8. Prototipo de la página principal (Natalia) . . . . .	54
5.9. Prototipo de la página principal (Pablo) . . . . .	55
5.10. Prototipo del editor (Jorge) . . . . .	55
5.11. Prototipo del editor con un desplegable en uno de los menús (Jorge) . . . . .	56
5.12. Prototipo de la vista previa (Jorge) . . . . .	57
5.13. Prototipo del editor (Natalia) . . . . .	57
5.14. Prototipo del editor (Pablo) . . . . .	58
5.15. Página principal de AdaptaMaterialEscolar . . . . .	59
5.16. Página principal de AdaptaMaterialEscolar tras subir un documento . . . . .	59
5.17. Ejemplo de ventana móvil . . . . .	61
5.18. Muestra de flujo . . . . .	62
5.19. Comparación de flujo de información de componentes sin y con redux. Fuente: <a href="https://blog.codecentric.de/en/2017/12/developing-modern-offline-apps-reactjs-redux-electron-part-3-reactjs-redux-basics/">https://blog.codecentric.de/en/2017/12/developing-modern-offline-apps-reactjs-redux-electron-part-3-reactjs-redux-basics/</a> . . . . .	63
5.20. Generación de la sopa de letras en AdaptaMaterialEscolar . .	65
5.21. Vista previa de la sopa de letras . . . . .	66
5.22. Diagrama de secuencia de la generación de la sopa de letras .	67
5.23. Diagrama de flujo de la generación de la sopa de letras . . .	68
5.24. Resultado de la generación de la sopa de letras . . . . .	69
5.25. Interfaz del ejercicio de definiciones/preguntas . . . . .	70
5.26. Interfaz del ejercicio de definiciones/preguntas con espacio extra activado . . . . .	71
5.27. Resultado del ejercicio de definiciones/preguntas usando las opciones de las Figuras 5.25 y 5.26 . . . . .	71
5.28. Interfaz del ejercicio pregunta para desarrollar . . . . .	72

5.29. Resultado de ejercicio pregunta para desarrollar . . . . .	73
5.30. Interfaz del ejercicio Verdadero o Falso . . . . .	74
5.31. Resultado de ejercicio Verdadero o Falso . . . . .	74
5.32. Interfaz de edición del ejercicio rellenar huecos . . . . .	75
5.33. Interfaz de selección de palabras del ejercicio rellenar huecos . . . . .	75
5.34. Resultado del ejercicio de rellenar huecos . . . . .	76
5.35. Ejemplo de organización de Components . . . . .	77
5.36. Ejemplo de organización de Pages . . . . .	77
5.37. Ejemplo de organización de Redux . . . . .	78
5.38. Ejemplo de organización de CKEditor . . . . .	79
6.1. Preguntas generales . . . . .	82
6.2. Preguntas para docentes . . . . .	83
6.3. Preguntas sobre SUS I . . . . .	83
6.4. Preguntas sobre SUS II . . . . .	84
6.5. Preguntas sobre SUS III . . . . .	84
6.6. Preguntas sobre diseño . . . . .	85
6.7. Preguntas sobre funcionalidades I . . . . .	85
6.8. Preguntas sobre funcionalidades II . . . . .	86
6.9. Preguntas sobre utilidad real . . . . .	87



# Índice de Tablas

5.1.	Puntuación de los requisitos creada por los docentes (Importancia) y los desarrolladores (Dificultad). . . . .	46
5.2.	Lista de funcionalidades relacionadas con adaptaciones de temario (T) y crear actividades (A), ordenada por prioridad. . . . .	47
5.3.	Lista de funcionalidades de adaptación de formato, ordenada por prioridad. . . . .	47



# Capítulo 1

## Introducción

**RESUMEN:** En este capítulo se realiza la introducción del Trabajo de Fin de Grado que va a ser presentado en este documento. Primero, en la Sección 1.1, se explicará la motivación que ha dado lugar al trabajo. A continuación, en la Sección 1.2, el objetivo que se pretende alcanzar. Por último, la estructura del proyecto final, en la Sección 1.3.

### 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 unos 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.

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 eta-

pa 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 currículo oficial.

Los docentes dedican demasiado tiempo a la creación del material académico de los estudiantes que necesitan adaptaciones no significativas. 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. Este TFG pretende proporcionar una herramienta al profesorado que facilite la adaptación de los contenidos curriculares a las asignaturas, reduciendo de esta forma el tiempo y esfuerzo que deben dedicar los docentes 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 así poder crear unidades didácticas personalizadas que se ajusten a las necesidades de cada estudiante.

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 la creación de diferentes tipos de ejercicios (como por ejemplo, ejercicios para llenar espacios en blanco, sopas de letras o ejercicios de relacionar conceptos), etc.

La herramienta estará basada en una Arquitectura Orientada a Servicios (SOA), es decir, principalmente recurriremos a aplicaciones externas (servicios) que proporcionen funcionalidad a nuestra aplicación.

Además, para que la aplicación se adapte a las necesidades reales de nuestros usuarios finales (los docentes), se va seguir un Diseño Centrado en el Usuario (DCU). Para ello, realizaremos entrevistas a los usuarios finales con el fin de definir qué necesidades tienen, qué diseño se ajusta más a sus necesidades y qué funcionalidades tienen mayor prioridad. También contaremos con ellos para evaluar la herramienta.

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 del proyecto

La estructura del proyecto está formada por seis capítulos, incluyendo este introductorio, en el que se explica la motivación y el objetivo del trabajo. A continuación, se expone el resumen de cada uno de ellos:

- En el **capítulo dos** se presenta el estado del arte, donde se explicará el dominio en el que se enmarca el proyecto.
- En el **capítulo tres** se describen las herramientas utilizadas en el desarrollo del proyecto.
- En el **capítulo cuatro** se explica la metodología de desarrollo que ha sido empleada durante todo el proceso y el motivo por el que se escogió dicha metodología.
- En el **capítulo cinco** se explica la fase de captura de requisitos, diseño, implementación y evaluación de la aplicación.
- En el **capítulo seis** se muestra cómo se ha hecho la evaluación de AdaptaMaterialEscolar.
- En el **capítulo siete** se presenta el trabajo individual que ha realizado cada miembro del equipo en el proyecto.
- En el **capítulo ocho** se describen las conclusiones y el trabajo futuro.



# Capítulo 2

## Estado del Arte

**RESUMEN:** En este capítulo nos centraremos en explicar el dominio en el que se enmarca nuestro proyecto. En la sección 2.1 explicaremos en detalle las necesidades de las personas con necesidades especiales, y cómo se tratan de satisfacer mediante adaptaciones curriculares. Además, en la sección 2.2, mencionaremos algunas de las herramientas actuales para realizar adaptaciones curriculares orientadas a nuestro objetivo. Por otra parte, se explicarán en la sección 2.3 las bases del Diseño Centrado en el Usuario de cara a ofrecer una experiencia de usuario satisfactoria.

### 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, e 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, com-

partidas 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 especiales 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 de apoyo especializado en metodologías que faciliten el aprendizaje de los alumnos.
  - **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, la lengua 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 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.

Aunque la adaptación curricular incluye un amplio número de posibilidades y técnicas, en general es necesario simplificar el lenguaje que se utiliza, e incluso ofrecer alternativas al mismo. De ambas cuestiones se encargan, entre otros, la lectura fácil y los pictogramas, que se explicarán a continuación.

### **2.1.1. Lectura fácil para adaptación curricular**

No todas las personas tienen la misma capacidad de entender el lenguaje natural, ya que esta capacidad puede estar limitada por diferentes causas. Según el manual “Lectura fácil: Métodos de redacción y evaluación” (Óscar García Muñoz, 2012), las personas más afectadas por este problema son las personas con discapacidad intelectual. Algunos ejemplos son las personas con Síndrome de Down o las personas con enfermedades y trastornos mentales y del comportamiento, como las personas con Trastorno del Espectro Autista (TEA). También se ven afectadas 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 un lenguaje más claro y comprensible, y una estructura más simple y esquemática. Las directrices de Lectura Fácil establecen un gran número de reglas, tanto estéticas como de redacción, para poder facilitar estas adaptaciones y garantizar un texto de fácil lectura. Algunas de las medidas que resultan de interés para este trabajo 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 el manual.
- Ortografía. Tener en cuenta ciertas pautas para evitar el uso de signos ortográficos que puedan dificultar la comprensión lectora. Se recomienda evitar el punto y seguido, y usar el punto y aparte para así dejar bien diferenciadas las frases. También se aconseja evitar el punto y coma y los puntos suspensivos. En cuanto a los números, es preferible que se escriban en formato cifra.
- Léxico. Tratar de usar un vocabulario que no sea complejo. Se aconseja utilizar palabras sencillas expresadas de forma simple y que en la medida de lo posible no sean largas. También se recomienda utilizar siempre el mismo sinónimo y evitar abreviaturas y siglas.
- Tipografía. El tamaño de letra debe de ser lo suficientemente grande, entre 12 y 16 puntos, y se deben utilizar tipografías sin remate como

Arial o Helvetica y reforzar la nitidez de los números.

- Composición del texto. Para las líneas, recomienda que cada una esté compuesta de una sola oración y que no superen los 60 caracteres. Contendrán un mínimo de 5 palabras y un máximo de 15 a 20, de modo que no queden ni muy cortas ni muy largas. En cuanto a los párrafos, aconseja alinear el texto a la izquierda, en párrafos y capítulos cortos, de forma que se favorezcan las pausas frecuentes. Recomienda que se mantenga un ritmo regular en la composición de párrafos para que así se muestre de una forma más nítida y organizada visualmente. Respecto a la distribución del espacio, recomienda una distribución ordenada y poco densa, cuanto más blanco mejor, y evitar un diseño en columnas.
- Imágenes. Se recomienda utilizar imágenes de apoyo al texto que estén bien referenciadas, utilizar símbolos o dibujos 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.

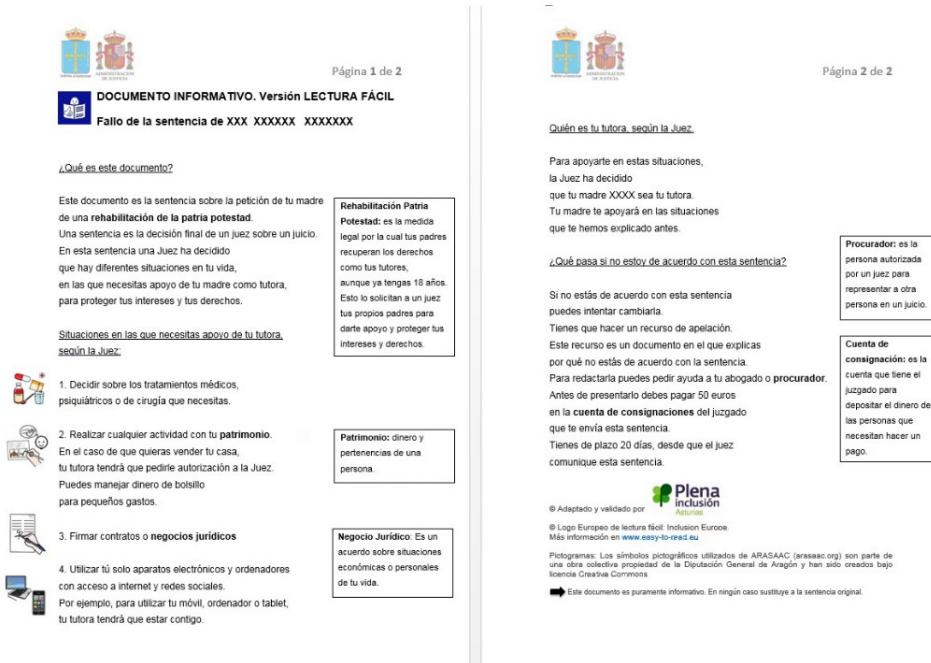


Figura 2.1: Ejemplo de documento adaptado a lectura fácil. En este caso, un documento jurídico

En la Figura 2.1 se muestra un ejemplo de un texto adaptado a lectura fácil. En este texto se puede ver un documento jurídico adaptado, en el que se aplican medidas mencionadas previamente, como la mayor sepa-

ración entre líneas, el uso de pictogramas, además de usar un lenguaje más sencillo, con frases más cortas y simples.

### 2.1.2. Pictogramas para adaptación curricular

Las personas con carencias sensoriales, cognitivas o con un conocimiento insuficiente de la lengua pueden presentar problemas a la hora de comunicarse. Por ello, los Sistemas Aumentativos y Alternativos de Comunicación (SAACs), que se encargan de proporcionar información en medios alternativos, o aumentando éste, se presentan como una gran opción para la 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, sintetizando 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 conocida por todos. 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 2.2 y 2.3 son ejemplos de pictogramas. En el primero se muestra un objeto físico, en este caso una rosa, y en el segundo se representa la acción de comer.

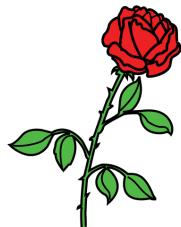


Figura 2.2: Ejemplo de pictograma de una rosa

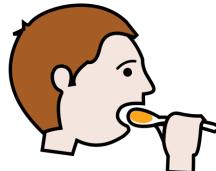


Figura 2.3: Ejemplo de pictograma que representa la acción de comer

Los sistemas pictográficos son sistemas que representan la realidad mediante pictogramas. Algunos ejemplos de sistemas pictográficos son:

- Sistema Pictográfico de Comunicación (SPC). Sus iconos representan de forma clara el concepto que pretenden transmitir. Son dibujos fácilmente diferenciables entre sí y de sencilla comprensión. La Figura 2.4 muestra un ejemplo de SPC, donde se ve una serie de pictogramas diferenciados por colores, junto con el significado de cada pictograma. Cada color representa una categoría diferente:
  - Amarillo: Se refiere a personas, incluyendo pronombres personales y posesivos.
  - Verde: Se refiere a verbos.
  - Azul: Se refiere a descripciones, principalmente adjetivos y adverbios.
  - Rosa: Se refiere a palabras de uso en interacciones sociales.
  - Blanco: Se refiere a la parte miscelánea, es decir, artículos, preposiciones, conjunciones..., todo lo que no se puede meter en las anteriores categorías.

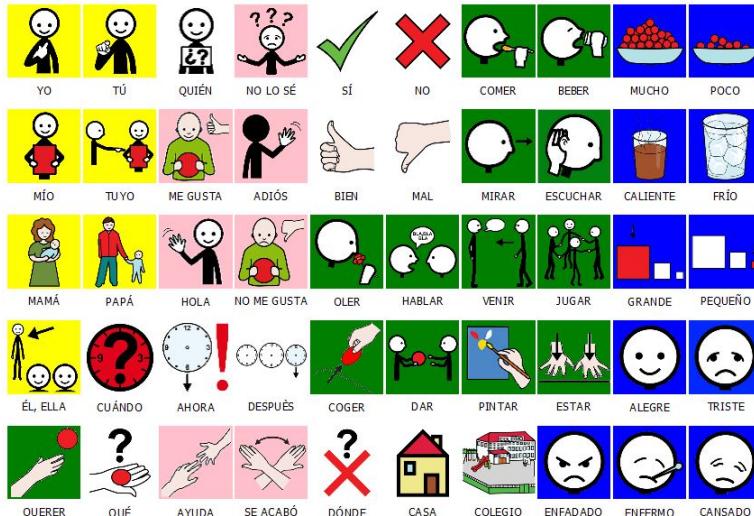


Figura 2.4: Ejemplo de pictogramas del Sistema Pictográfico de Comunicación (SPC)

- Minspeak. En este sistema los pictogramas no tienen un significado concreto preestablecido, sino que se fija por el logopeda y usuario, lo que permite personalizar los mensajes. La Figura 2.5 muestra un

ejemplo de un sistema Minspeak, donde se define el significado de los pictogramas a través de una combinación de estos.



Figura 2.5: Ejemplo de sistema pictográfico Minspeak

- ARASAAC. El Portal Aragonés de Comunicación Aumentativa y Alternativa (ARASAAC), creado en 2007, ofrece un amplio catálogo de pictogramas de libre acceso. Dichos pictogramas se encuentran divididos en categorías: Alimentación, Ocio, Lugar, Ser vivo, Educación, Tiempo, Miscelánea, Desplazamiento, Religión, Trabajo, Comunicación, Documento, Conocimiento y Objeto. Es uno de los sistemas pictográficos más usados en España ya que incluye un amplio número de herramientas dedicadas a pictogramas, como buscadores o generador de documentos con pictogramas, como pueden ser calendarios u horarios. El catálogo de ARASAAC ofrece pictogramas tanto en color, como con blanco y negro, y fondos de colores. Los pictogramas de las Figuras 2.2 y 2.3 pertenecen a ARASAAC.

## 2.2. Herramientas existentes para adaptaciones curriculares

La existencia de herramientas que faciliten adaptaciones curriculares es muy limitada. Existen herramientas enfocadas a realizar solo alguna de las adaptaciones que se pueden necesitar, pero de manera individualizada.

*ARAWORD*<sup>1</sup> es una herramienta de procesado de textos que combina la escritura de texto con pictogramas y facilita la elaboración de materiales. También resulta una herramienta muy útil para ser usada por usuarios que están adquiriendo el proceso de la lectoescritura, ya que la aparición del pictograma, 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 2.6 se

<sup>1</sup>[http://aulaabierta.arasaac.org/araword\\_inicio](http://aulaabierta.arasaac.org/araword_inicio)

muestra un ejemplo de la interfaz de ARAWORD, donde se traducen las oraciones a pictogramas.

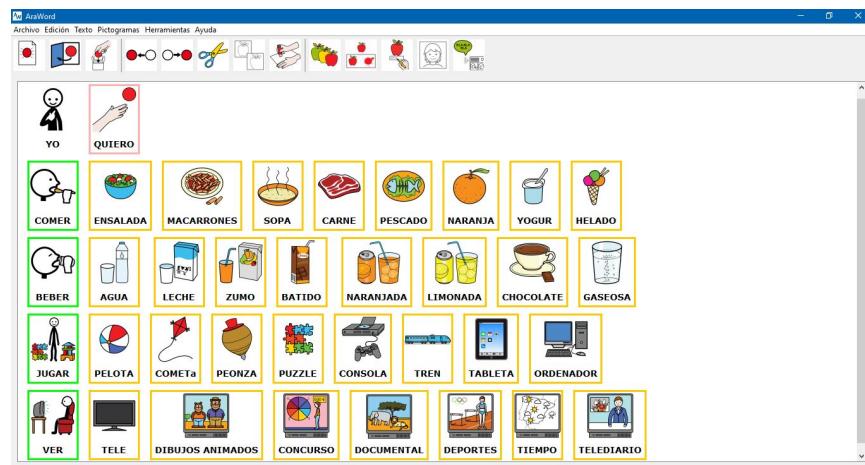


Figura 2.6: Interfaz de ARAWORD

*EasyTestMaker*<sup>2</sup> es una web de creación de exámenes online. Dicha herramienta incluye opciones de generación de ejercicios sencillos para exámenes, como ejercicios de verdadero/falso, asociación de conceptos, rellenar espacios en blanco y respuestas cortas. La Figura 2.7 muestra un ejemplo de cómo se genera un ejercicio de asociar conceptos, mientras que la Figura 2.8 muestra el resultado, con otros tipos de ejercicios. EasyTestMaker ofrece un plan gratuito que permite crear hasta 25 tests, mientras que la versión de pago ofrece más opciones como insertar imágenes, exportar a PDF y personalización del texto.

Por otra parte, existen herramientas que ayudan a la creación de esquemas y resúmenes. Una herramienta para realizar resúmenes es *Resoomer*<sup>3</sup>, la cual devuelve un texto resumido a través de un texto introducido. En la Figura 2.9 se muestra un ejemplo de la interfaz de Resoomer.

Para la generación de esquemas, habitualmente se usan herramientas de diseño como *Canva*<sup>4</sup>. Canva ofrece plantillas prediseñadas donde el usuario puede introducir las palabras a organizar. La Figura 2.10 muestra un ejemplo de cómo se hace un esquema con Canva.

<sup>2</sup><https://www.easytestmaker.com/>

<sup>3</sup><https://resoomer.com/es/>

<sup>4</sup><https://www.canva.com/>

The screenshot shows the configuration interface for a matching exercise. At the top, there are tabs for 'Matching', 'Instructions', 'Image', and 'Options'. Below the tabs, a descriptive text reads: 'Enter options on left and answer on the right of the same row. Choose your ordering or click to randomize.' The main area contains four rows of input fields. Each row has a concept name on the left and its corresponding color on the right. Row 1: 'Cielo' (left) and 'Azul' (right). Row 2: 'Tierra' (left) and 'Marrón' (right). Row 3: 'Nube' (left) and 'Blanco' (right). Row 4: An empty input field (left) and another empty input field (right). Each row has a small '1' icon with a downward arrow and a red 'X' icon at the top. To the right of each row are dropdown menus labeled 1, 2, 3, and 4. A large red 'X' is placed in the bottom right corner of the fourth row's right side. Below the rows is a button labeled '+ add option'. Underneath the rows, there is a section for 'Point Value' with a checkbox for 'Bonus question'. A numeric input field shows '1' with a dropdown arrow, and a range selector indicates '1.00 - 99.99'. A note below states: 'Point value is for each option. Total points for all options is the number of options times the point value.' At the bottom are three buttons: 'Ok - Randomize Order' (blue), 'Ok - Keep Order' (blue), and 'Cancel' (white).

Figura 2.7: Ventana de generación de ejercicio de asociar conceptos con EasyTestMaker

- 1) Los humanos están compuestos de células procariotas
  - a) True
  - b) False
  
- 2) \_\_\_\_\_ Cielo
- 3) \_\_\_\_\_ Tierra
- 4) \_\_\_\_\_ Nube
  
- a) Blanco
- b) Azul
- c) Marrón
  
- 5) La capital de Andalucía es \_\_\_\_\_, mientras que la capital de \_\_\_\_\_ es Barcelona.

Figura 2.8: Ejemplo de examen generado con EasyTestMaker

## 2.3. Diseño Centrado en el Usuario

El Diseño Centrado en el Usuario (DCU) (Ruiz-Granados, Ballesteros, Manjón y Lorenz, 2019) consiste en un enfoque de diseño de interfaces cuyo objetivo es crear un producto que resuelva las necesidades de los usuarios



Figura 2.9: Interfaz y funcionamiento de Resoomer

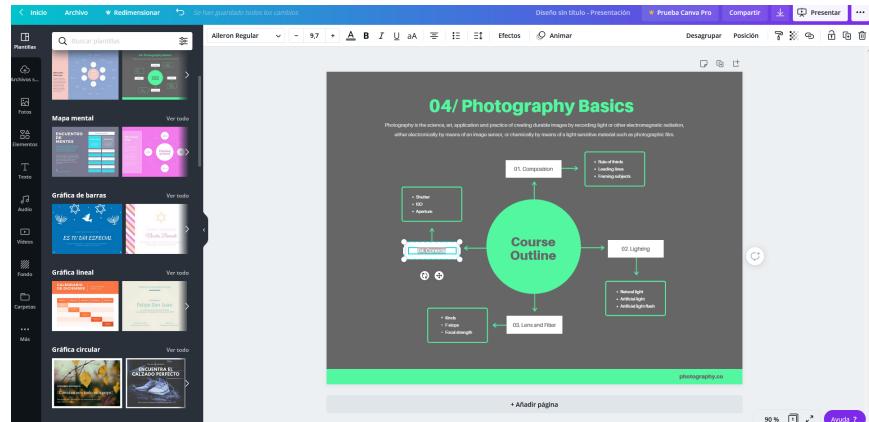


Figura 2.10: Interfaz de Canva para la creación de un esquema usando una plantilla

finales, consiguiendo así una mayor calidad de producto y por tanto una mayor satisfacción de los mismos. Con dicha metodología se pretende entender previamente a los usuarios para poder encontrar soluciones acordes a sus necesidades específicas. Este proceso se basa en tres pilares fundamentales: conocer al usuario, prototipado del producto y evaluación, que se explicarán en las siguientes secciones.

### 2.3.1. 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 realiza un conjunto de preguntas al sujeto de estudio 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 de los usuarios finales.
- **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 descritas por el usuario.
- **Diarios de uso.** Se proporciona a los usuarios un diario en el que 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.
- **Análisis de 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.

### 2.3.2. 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 en medios digitales. Los prototipos en papel pueden ser de distintos tipos:

- **Guiones gráficos.** Conjunto de ilustraciones pintadas 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 2.11 se muestra un ejemplo de guión gráfico que secuencia una situación en la que una aplicación de controlar eventualmente el estado de ánimo de una persona cercana pudiera ser útil.



Figura 2.11: Ejemplo de guión gráfico

- **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 2.12 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

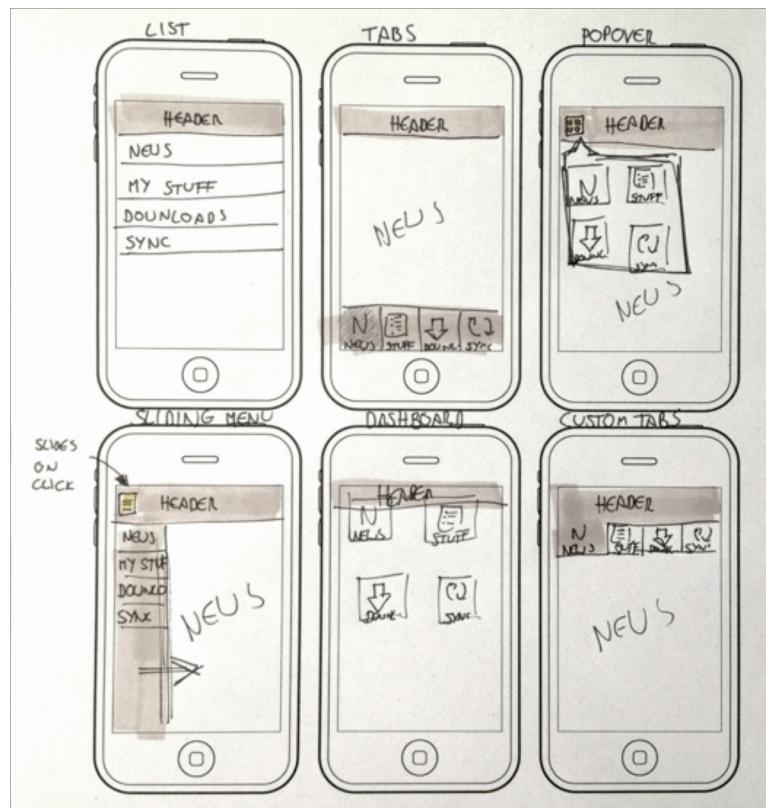


Figura 2.12: Ejemplo de boceto en papel

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 2.13 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 realista. Algunas herramientas para realizar prototipos digitales son *Balsamiq*<sup>5</sup>, *JustInMind*<sup>6</sup>, *Moqups*<sup>7</sup>, o incluso HTML y CSS y Photoshop.

<sup>5</sup><https://balsamiq.com/>

<sup>6</sup><https://www.justinmind.com/>

<sup>7</sup><https://moqups.com/>

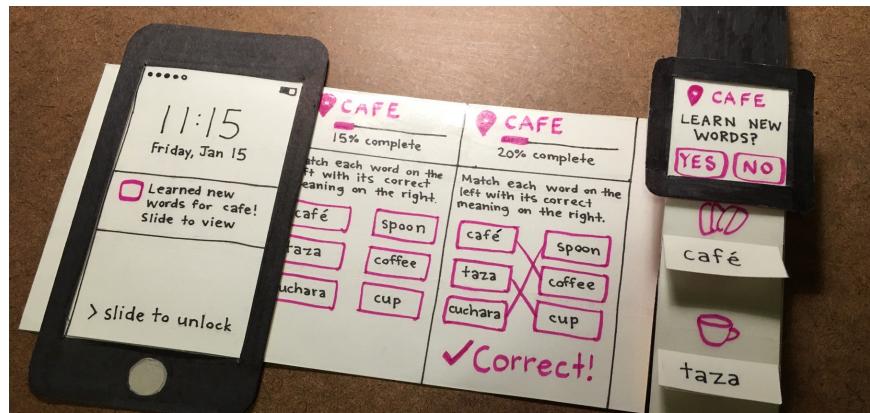


Figura 2.13: Ejemplo de prototipo en papel interactivo

### 2.3.3. 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 evaluaciones, las heurísticas (con expertos) 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 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.

Todo este proceso de diseño muestra un carácter iterativo. El usuario final se verá involucrado de manera constante con el fin de dar feedback y garantizar que el prototipo diseñado cumple con lo esperado.

# 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 muestra el *framework* que se ha utilizado para la creación de la página web. En la sección 3.3 se habla de la API de ARASAAC usada para la búsqueda de pictogramas. En la sección 3.4 se presenta la librería *Jest* usada para la realización de pruebas. En la sección 3.5 se explica CKEditor, el editor de texto que hemos usado en la aplicación. En la sección 3.6 se explica el paquete NPM usado para la generación de las sopas de letras.

### 3.1. Moqups

Moqups<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,

---

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

haciendo que un botón realice una función específica). También es colaborativo y permite la gestión de roles.

En la Figura 3.1 se puede ver la interfaz de un proyecto en blanco. En la parte superior de la página, se encuentran las opciones para crear figuras geométricas y añadir notas, así como agregar a otros usuarios, y exportar el proyecto como una serie de imágenes en formato PNG o PDF y a formato HTML (Figura 3.2). 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, comentarios, imágenes, iconos, etc.), como se puede ver en la Figura 3.3. Por último, en el menú lateral de la derecha, podemos encontrar el formato de las diferentes páginas (o componentes) (Figura 3.4), y las interacciones disponibles (Figura 3.5).

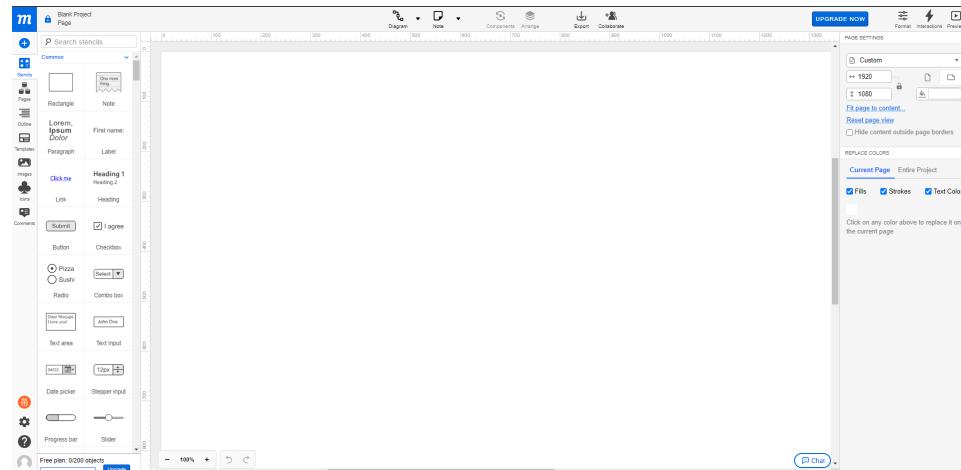


Figura 3.1: Interfaz de Moqups



Figura 3.2: Menú superior

En este proyecto se ha usado Moqups para la creación de la versión 1.0 del prototipo de la aplicación web (en la sección 5.2.2 se puede leer una explicación detallada de cómo se ha creado este prototipo y de las diferentes vistas del mismo).

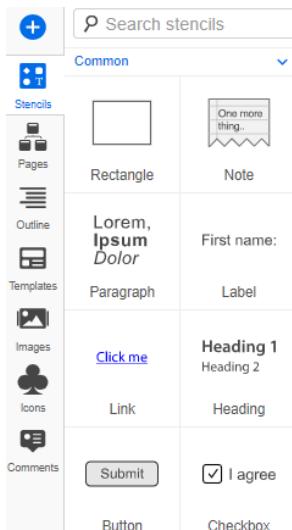


Figura 3.3: Menú lateral izquierdo

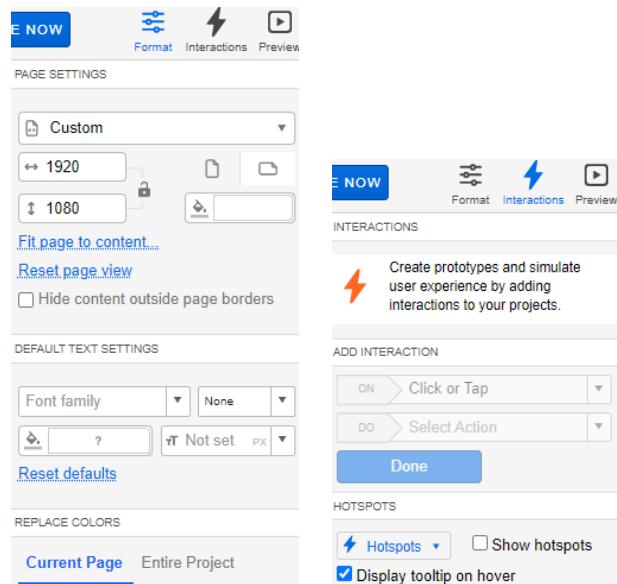


Figura 3.4: Menú lateral derecho (formato)

Figura 3.5: Menú lateral derecho (interacciones)

## 3.2. React

*React*<sup>2</sup> es una librería de *JavaScript*, creada por Facebook y de código abierto<sup>3</sup>, 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, pero facilita tanto la codificación como la lectura del código. En la Figura 3.6 se puede ver un ejemplo de una funcionalidad sin emplear el formato *JSX*

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

<sup>3</sup><https://github.com/facebook/react>

y en la Figura 3.7 la misma funcionalidad pero usando *JSX*.



```

EDITOR EN VIVO DE JSX
JSX?

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 3.6: Funcionalidad sin usar JSX



```

EDITOR EN VIVO DE JSX
JSX?

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

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

```

Figura 3.7: Funcionalidad empleando 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>4</sup>, junto

<sup>4</sup><https://es.reactjs.org/docs/getting-started.html>

con un tutorial para aprender desde cero<sup>5</sup>.

Esta librería la hemos usado en el proyecto para implementar los distintos tipos de adaptaciones en la aplicación web (ver sección 5.1.1).

### 3.3. API de ARASAAC

El Centro Aragonés para la Comunicación Aumentativa y Alternativa (ARASAAC) proporciona una API pública<sup>6</sup> (*Application Programming Interfaces*, en castellano “interfaz de programación de aplicaciones”), cuyo uso es gratuito siempre y cuando la aplicación a desarrollar sea no comercial<sup>7</sup> (está bajo la licencia de *Creative Commons BY-NC-SA*, es decir, “Reconocimiento-NoComercial-CompartirIgual”<sup>8</sup> en el que se debe dar crédito a los/as autores/as originales, no se puede usar para fines comerciales y en el caso de que se modifique o transforme ese material se debe distribuir bajo la misma licencia).

La API de ARASAAC consta de cuatro apartados, cada uno de los cuales tiene una ruta web dependiendo de lo que se quiera consultar: materiales, pictogramas, palabras clave y usuarios. Como en este proyecto usaremos solo la búsqueda de pictogramas, nos centraremos en el apartado de pictogramas. La API de ARASAAC proporciona ocho servicios web para pictogramas. Todos ellos se usan con el método de petición *GET* y con formato de salida en *JSON*. De los ocho servicios web proporcionados, en este TFG se usarán los siguientes:

- **/pictograms/{idPictogram}**: Devuelve el pictograma en formato imagen dado su identificador {idPictogram}. Si tiene éxito devolverá la imagen del pictograma en formato png. En caso contrario devolverá un mensaje de error.
- **/pictograms/{locale}/{idPictogram}**: Devuelve los datos de un pictograma dado su identificador {idPictogram} e idioma {locale}<sup>9</sup>. Si tiene éxito devolverá una lista en formato *JSON* con los datos de cada pictograma. En caso contrario devolverá un error 404.

---

<sup>5</sup><https://es.reactjs.org/tutorial/tutorial.html>

<sup>6</sup><https://arasaac.org/developers/api>

<sup>7</sup><https://arasaac.org/developers/>

<sup>8</sup><https://creativecommons.org/licenses/by-nc-sa/4.0/>

<sup>9</sup>Idiomas disponibles: an, ar, bg, br, ca, de, el, en, es, et, eu, fa, fr, gl, he, hr, hu, it, mk, nl, pl, pt, ro, ru, sk, sq, sv, sr, val, zh

- **/pictograms/{idPictogram}/languages/{languages}**: Devuelve los datos de un pictograma dado su identificador {idPictogram} en uno o más idiomas<sup>9</sup> especificados en {languages}. Si tiene éxito devolverá los datos del pictograma en formato *JSON* en los idiomas proporcionados. En caso contrario devolverá un mensaje de error.
- **/pictograms/{locale}/search/{searchText}**: Devuelve en una lista los datos de los pictogramas en el idioma {locale}<sup>9</sup> que contenga la cadena de caracteres {searchText} pasada como parámetro. Si tiene éxito devolverá dicha lista en formato *JSON* con los datos de cada pictograma encontrado. En caso contrario no devolverá nada.
- **/pictograms/{locale}/bestsearch/{searchText}**: Devuelve en una lista los datos de los pictogramas en el idioma {locale}<sup>9</sup> que contenga **exactamente** la cadena de caracteres {searchText} pasada como parámetro. Si tiene éxito devolverá dicha lista en formato *JSON* con los datos de cada pictograma encontrado. En caso contrario no devolverá nada.

### 3.4. Jest

*Jest*<sup>10</sup> es una librería para *Javascript*, creada por Facebook, que permite realizar distintos tipos de pruebas y es compatible con muchos *frameworks* incluyendo el que hemos elegido (*React*). *Jest* tiene una instalación muy sencilla, de pocos pasos, y su configuración es mínima. La documentación<sup>11</sup> es completa, y contiene lo necesario para poder desarrollar estos tipos de pruebas, junto con una serie de ejemplos realizados paso a paso. El uso de esta librería proporciona ventajas frente a no usarla o hacer tests de forma tradicional:

- **Compatibilidad.** Es compatible con la mayoría de *frameworks* que existen, tales como Angular, React, NodeJS, Babel, etc.
- **Rapidez.** Cuando las pruebas están vinculadas a la CPU, se ahorra muchísimo tiempo en la ejecución de las pruebas. Esta rapidez se consigue gracias a la combinación de varios factores:
  1. Paralelización. Gracias a la paralelización, se produce una gran ganancia en el rendimiento.

---

<sup>10</sup><https://jestjs.io/es-ES/>

<sup>11</sup><https://jestjs.io/es-ES/docs/getting-started>

2. Ejecuta primero las pruebas más lentas. Aprovecha al máximo su capacidad de procesamiento al ejecutar primero las pruebas más lentas.
  3. Tiene una caché de transformaciones babel incorporada. Aplicar transformaciones al código requiere un uso intensivo de CPU. Gracias al uso de una caché que se comparte entre procesos, se reduce mucho tiempo al ejecutar las pruebas.
- **Instantáneas.** Las pruebas de instantáneas son de gran utilidad para garantizar que la interfaz no cambia de forma inesperada.

Una vez que se ha instalado Jest, para realizar un test hay que seguir estos pasos:

1. Crear un fichero *Javascript* donde estará la función a probar (por ejemplo, *suma.js*):

```
function suma(a,b){
    return a + b;
}

module.exports = suma;
```

2. Crear un fichero *Javascript* donde contendrá la prueba (por ejemplo *suma.test.js*). Para escribir la prueba, tiene que seguir el siguiente esquema:

```
test(título, () =>{
    expect(valor, función, expresión...).comparador(
        resultadoEsperado);
});
```

donde “comparador” puede ser *.toBe()*, *.toEqual()*, *.toBeNull()*, etc., una de las funciones comparadoras que más se ajuste a lo que se quiere probar. En la documentación oficial<sup>12</sup>, hay un apartado llamado “Utilizando Comparadores”, donde enseñan algunas funciones comparador. Si se quiere ver todas las funciones que ofrece es mejor visitar la API<sup>13</sup>.

En el caso de la prueba que queremos escribir:

```
const suma = require("./suma");

test("Prueba de suma: 1 + 2 = 3", () =>{
    expect(suma(1,2)).toBe(3);
});
```

---

<sup>12</sup><https://jestjs.io/es-ES/docs/using-matchers>

<sup>13</sup><https://jestjs.io/es-ES/docs/api>

```
test("Prueba de suma incorrecta: 1 + 2 = 4", () =>{
    expect(suma(1,2)).toBe(4);
});
```

Si ejecutamos solo la primera función, nos saldrá que hemos pasado el test (Figura 3.8). En cambio, si ejecutamos ambos tests, vemos que el primero sí lo ha pasado, pero el segundo ha fallado, y nos da más información sobre lo que debería haber dado y la línea de código donde ha fallado (Figura 3.9).

```
PASS src/suma.test.js
  ✓ Prueba de suma: 1 + 2 = 3 (3 ms)

Test Suites: 1 passed, 1 total
Tests:       1 passed, 1 total
Snapshots:   0 total
Time:        0.762 s, estimated 1 s
Ran all test suites.
```

Figura 3.8: Test pasado correctamente

```
FAIL src/suma.test.js
  ✓ Prueba de suma: 1 + 2 = 3 (2 ms)
  ✗ Prueba de suma incorrecta: 1 + 2 = 4 (4 ms)

● Prueba de suma incorrecta: 1 + 2 = 4

    expect(received).toBe(expected) // Object.is equality

      Expected: 4
      Received: 3

      6 |
      7 |     test("Prueba de suma incorrecta: 1 + 2 = 4", () =>{
      > 8 |         expect(suma(1,2)).toBe(4);
          ^
      9 |     });
      at Object.<anonymous> (src/suma.test.js:8:23)

Test Suites: 1 failed, 1 total
Tests:       1 failed, 1 passed, 2 total
Snapshots:   0 total
Time:        0.644 s, estimated 1 s
Ran all test suites.
```

Figura 3.9: Test no pasado correctamente

Esta librería la hemos usado en el proyecto para realizar las pruebas unitarias y de integración de las distintas partes de la aplicación.

### 3.5. CKEditor

*CKEditor*<sup>14</sup> es un editor de código abierto de tipo “WYSIWYG”<sup>15</sup> y es totalmente personalizable. *CKEditor* tiene dos versiones diferentes:

- **CKEditor 4:** Es la versión antigua compatible con la mayoría de los navegadores. Con ella se puede tener tres tipos de editor: *Article Editor* (editor para artículos), *Document Editor* (editor para documentos) e *Inline Editor* (editor en línea). La documentación<sup>16</sup> es muy completa: contiene una guía de instalación paso a paso, además de bastantes ejemplos para diferentes propósitos (insertar contenido, personalizar la interfaz de usuario...). También tiene una demo online<sup>17</sup> que muestra los diferentes editores para esta versión. Gracias al programa LTS (*Long Term Support*) tanto el desarrollo como el soporte de esta versión están garantizados hasta 2023.
- **CKEditor 5:** Es la versión más moderna con un rediseño y mejora de la interfaz y la introducción de un nuevo modelo de datos. A diferencia de la versión anterior, esta versión tiene más tipos de editor: *Classic* (el editor clásico), *Balloon* (editor de globo, que permite insertar contenido directamente en la ubicación que se está señalando), *Balloon Block* (el mismo que el anterior pero se edita por bloques), *Inline* (el mismo que en *CKEditor 4*), *Document* (el mismo que en *CKEditor 4*) y *Pagination* (el editor de paginación, que permite ver las líneas de salto de página que coincidirían con los límites de la página cuando el documento se exporta a PDF o Word). También tiene una demo online<sup>18</sup> de los diferentes tipos de editor para esta versión, y un constructor online<sup>19</sup> donde se puede elegir el tipo de editor y los complementos que tendrá. La documentación<sup>20</sup> en esta versión también es bastante

---

<sup>14</sup><https://ckeditor.com/>

<sup>15</sup>WYSIWYG, acrónimo de *What You See Is What You Get* (en español, “lo que ves es lo que obtienes”), es una frase aplicada a los procesadores de texto y otros editores con formato, que permiten escribir un documento mostrando directamente el resultado final, frecuentemente el resultado impreso. <https://es.wikipedia.org/wiki/WYSIWYG>

<sup>16</sup><https://ckeditor.com/docs/ckeditor4/latest/>

<sup>17</sup><https://ckeditor.com/ckeditor-4/demo/>

<sup>18</sup><https://ckeditor.com/ckeditor-5/demo/>

<sup>19</sup><https://ckeditor.com/ckeditor-5/online-builder/>

<sup>20</sup><https://ckeditor.com/docs/ckeditor5/latest/>

completa: tiene una guía de instalación paso a paso, ejemplos de los editores, además de una guía de cómo trabajar con el *framework* de este editor.

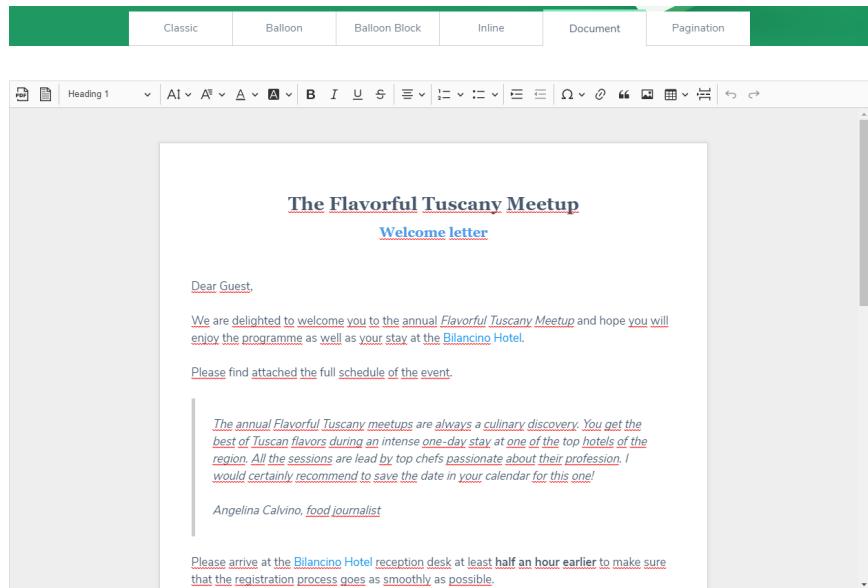


Figura 3.10: Ejemplo de CKEditor 5

En este proyecto hemos usado la versión *CKEditor 5* para crear un documento editable, en el que el usuario puede realizar los diferentes tipos de adaptaciones que ofrece la aplicación, y poder descargar el resultado final en formato PDF. En la Figura 3.10 se puede ver un ejemplo de la versión *CKEditor 5*<sup>21</sup>. En la parte superior se muestra una barra de navegación con los diferentes ejemplos de tipos de editor mencionados anteriormente, y en la inferior, el editor junto con su barra de herramientas (donde el primer elemento de ésta barra es la exportación a formato PDF).

### 3.6. Word-search

Para la creación de los ejercicios de sopas de letras empleamos el paquete NPM (*Node Package Manager*) *word-search*<sup>22</sup>. Para crear una sopa de letras con este paquete, se necesitan unas opciones de configuración, que están

<sup>21</sup>Fuente: <https://ckeditor.com/ckeditor-5/demo/#document>

<sup>22</sup><https://www.npmjs.com/package/@balex41/word-search>

encapsuladas en un objeto, y devuelve la sopa de letras en un array. Las opciones de configuración son las siguientes:

- cols: Es un número entero que representa el número de columnas que tendrá la sopa de letras. Por defecto tiene un valor de 10.
- rows: Es un número entero que representa el número de filas que tendrá la sopa de letras. Por defecto tiene un valor de 10.
- disabledDirections: Es un array de strings que representa las direcciones cardinales (“N”, “S”, “E”, “W”, “NE”, “NW”, “SE”, “SW”) no válidas para colocar las palabras en la sopa de letras. Por defecto tiene un valor vacío, es decir, todas las direcciones están permitidas.
- dictionary: Es un array de strings que contiene las palabras que debe contener la sopa de letras. Por defecto tiene un valor vacío.
- maxWords: Es un número entero que representa el número de palabras máximo que se insertarán en la sopa de letras. Por defecto tiene un valor de 20. Aunque la longitud del array *dictionary* supere este valor, solo se insertará ese número de palabras en la sopa de letras.
- backwardsProbability: Es un número decimal entre 0 y 1 que representa la probabilidad que tiene cada palabra de escribirse al revés. Solo funciona si no se deshabilita, en una dirección cardinal, uno de los extremos. Por ejemplo, para poder escribir una palabra al revés en vertical, se deben tener las direcciones cardinales “N” y “S” habilitadas, no solo una de ellas. Por defecto tiene un valor de 0.3.
- uppercase: Es un booleano que indica si las letras de la sopa de letras se quieren en mayúscula (*true*) o no (*false*). Por defecto tiene el valor *true*.
- diacritics: Es un booleano que indica si las letras deberían mantener las tildes en la sopa de letras. Por defecto tiene el valor *false*, es decir, no mantener las tildes en la sopa de letras.

En el caso de que alguna de estas opciones no se modifiquen, no daría ningún error, sino que cogería el valor por defecto que tuvieran.

Para crear la sopa de letras, es necesario hacer una llamada a Wordsearch, pasándole como parámetro las opciones de configuración. En este ejemplo, se ha establecido un tablero de 8x9, se han habilitado todas las direcciones, la probabilidad que tiene cada palabra de escribirse al revés se ha fijado a 0.5 (cada palabra tiene un 50 % de probabilidad de escribirse al revés), el

número máximo de palabras que se insertarán es 20, se quieren mantener las tildes, las letras se quieren en mayúsculas y hay que buscar las palabras perro, gata, flamenco y avestruz.

```
const WordSearch = require("@blex41/word-search");

const options = {
  cols: 9,
  rows: 8,
  disabledDirections: [],
  dictionary: ["perro", "gata", "flamenco", "avestruz"],
  maxWords: 20,
  backwardsProbability: 0.5,
  upperCase: true,
  diacritics: true
};

const ws = new WordSearch(options);
```

Una vez creada la sopa de letras, el paquete proporciona las siguientes funciones<sup>23</sup>:

- `grid()`: Devuelve un array de caracteres donde cada posición representa una letra de la sopa de letras. En la Figura 3.11 se puede ver un ejemplo del resultado de esta función.

```
Array (8 items)
▶ 0: ["H", "P", "C", "D", "E", "H", "E", "J", "A"]
▶ 1: ["Z", "I", "E", "M", "C", "K", "O", "O", "I"]
▶ 2: ["V", "G", "I", "R", "W", "X", "C", "H", "A"]
▶ 3: ["W", "W", "R", "L", "R", "R", "G", "I", "T"]
▶ 4: ["S", "K", "H", "U", "D", "O", "B", "E", "A"]
▶ 5: ["A", "V", "E", "S", "T", "R", "U", "Z", "G"]
▶ 6: ["C", "F", "L", "A", "M", "E", "N", "C", "O"]
▶ 7: ["R", "N", "T", "J", "R", "W", "L", "O", "J"]
▶ ⚡ Array Prototype
```

Figura 3.11: Ejemplo de resultado de sopa de letras usando `grid()`

- `toString()`: Devuelve un array de strings donde cada posición representa una fila de la sopa de letras. En la Figura 3.12 se puede ver un ejemplo del resultado de esta función.

<sup>23</sup>Para cada ejemplo, se han usado las opciones de configuración de la Figura ???. Así mismo, se han ejecutado dichas funciones una sola vez y seguidas, dando como resultado el mismo tablero.

```

H P C D E H E J A
Z I E M C K O O I
V G I R W X C H A
W W R L R R G I T
S K H U D O B E A
A V E S T R U Z G
C F L A M E N C O
R N T J R W L O J

```

Figura 3.12: Ejemplo de resultado de sopa de letras usando `toString()`

- `words()`: Devuelve un array de objetos con las palabras que se han insertado correctamente en la sopa de letras. Cada objeto tiene tres campos:
  - `word`: la palabra que está en el campo *dictionary*.
  - `clean`: este campo depende de las opciones de *upperCase* y *diacritics*, es decir, se escribirá la palabra del campo `word` con o sin mayúsculas y con o sin tildes (dependiendo si los valores de dichas opciones están a *true* o *false*).
  - `path`: contiene un array de objetos con las posiciones “x” e “y” de cada letra de la palabra en la sopa de letras.

En la Figura 3.13 se puede ver un ejemplo del resultado cuando se usa la función `words()`.

```

Array (4 items)
  ▶ 0: Object
    ▶ word: "avestruz"
    ▶ clean: "AVESTRUZ"
    ▶ path: [Object {x: 0, y: 5}, Object {x: 1, y: 5}, Object {x: 2, y:
      5}, Object {x: 3, y: 5}, Object {x: 4, y: 5}, ...]
      ...
  ▶ 1: Object {word: "flamenco", clean: "FLAMENCO", ...}
  ▶ 2: Object {word: "gata", clean: "GATA", ...}
  ▶ 3: Object {word: "perro", clean: "PERRO", ...}
  ▶ Array Prototype

```

Figura 3.13: Ejemplo de las palabras insertadas al usar `words()`

- `dump()`: Devuelve un objeto que representa el estado de la sopa de letras (es una *snapshot* del tablero). Esto puede resultar útil si se quiere

guardar el juego y usarlo después con la siguiente función. En la Figura 3.14 se puede ver un ejemplo del objeto resultante cuando se usa esta función.

```

Object
  settings: Object
    cols: 9
    rows: 8
    disabledDirections: []
    allowedDirections: ["N", "S", "E", "W", "NE", "NW", "SE", "SW"]
    dictionary: ["perro", "gata", "flamenco", "avestruz"]
    maxWords: 20
    backwardsProbability: 0.5
    upperCase: true
    diacritics: true
    ...
  data: Object Properties Viewer
    grid: [[H, P, C, D, E, H, E, J, A], [Z, I, E, M, C, K, O, O, I], ...]
    words: [Object {word: "avestruz", clean: "AVESTRUZ", ...}, Object {word: "flamenco", clean: "FLAMENCO", ...}, ...]
  
```

Figura 3.14: Ejemplo del objeto resultante al usar `dump()`

- `load(backup)`: Carga el parámetro *backup* que representa un estado de la sopa de letras generado previamente con `dump()`.
- `read(start, end)`: Devuelve un string con las letras que se forman al leer las posiciones de *start* a *end*. En la Figura 3.15 se puede ver la forma en la que se llama a esta función, pasándole los objetos posición *start* y *end* y, en la Figura 3.16, un ejemplo del string formado al leer dichas posiciones.

```
ws.read({ x: 1, y: 0 }, { x: 5, y: 4 })
```

Figura 3.15: Ejemplo de llamada a la función `read()`

"PERRO"

Figura 3.16: Ejemplo del string formado cuando se usan los parámetros de la Figura 3.15

## 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 principalmente: tradicionales y ágiles. Las metodologías tradicionales “buscan imponer disciplina al proceso de desarrollo de software y de esa forma volverlo predecible y eficiente” (Cadavid, Martínez y Vélez, 2013), y están orientadas a procesos. El desarrollo del proyecto se inicia mediante una serie de etapas: captura de requisitos, análisis, diseño y 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

---

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

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 solo puede ver el resultado al final. Este tipo de metodología se suele usar cuando hay un problema conocido y se sabe su solución o si los requisitos están muy claros desde el principio. Algunos ejemplos de metodologías tradicionales son: *ITIL*<sup>2</sup>, *Métrica 3*<sup>3</sup>, *RUP*<sup>4</sup>, etc.

Por otro lado, las metodologías ágiles tienen un enfoque adaptativo, en el que cualquier cambio se acoge con normalidad (se espera que ocurran cambios). Al contrario que las metodologías tradicionales, las cuales estaban orientadas a los procesos, las metodologías ágiles están orientadas a las personas. En estas metodologías lo que se pretende es generar valor para el cliente, por lo que se necesita un representante de éste, que puede ser un miembro más del equipo, y, sobre todo, una comunicación constante con él. Uno de los objetivos de las metodologías ágiles es conseguir, lo antes posible, un producto que sea funcional y que genere valor al cliente. Esto se logra a través de constantes entregas de software, lo que implica que el cliente pueda proporcionar *feedback* que permita aumentar dicho valor. Este tipo de metodología funciona bien cuando los requisitos no están claros y tampoco lo está el problema a solucionar y la manera en la que se puede desarrollar. Algunos ejemplos<sup>5</sup> de metodologías ágiles son: *Scrum*, *Extreme Programming (XP)*, *Kanban*, 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*”<sup>6</sup> 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.

*Kanban* es menos prescriptivo que otras metodologías ágiles como *Scrum*

---

<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\\_v3.html](https://administracionelectronica.gob.es/pae/Home/pae_Documentacion/pae_Metodolog/pae_Metrica_v3.html)

<sup>4</sup><http://ima.udg.edu/~sellares/EINF-ES2/Present1011/MetodoPesadesRUP.pdf>

<sup>5</sup><https://www.iebschool.com/blog/que-son-metodologias-agiles-agile-scrum/>

<sup>6</sup><https://blog.trello.com/es/metodologia-kanban>

y tiene tres reglas principales(Garzas, 2011):

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, evitando que éste último se acumule. Además, permite conocer en qué tarea está trabajando cada uno. El tablero *Kanban* 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 suele tener las siguientes:
  - *To Do*: Tareas pendientes.
  - *Doing*: Tareas en curso.
  - *Done*: Tareas terminadas, validadas y aprobadas. 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.
2. **Determinar y respetar el trabajo en curso.** 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.

En nuestro proyecto, distinguimos dos tipos de tareas:

- Tareas de memoria: Son las tareas de redacción de la memoria.

- Tareas de implementación: Son tareas de diseño y/o desarrollo de código.

En la Figura 4.1 se puede ver nuestro tablero *Kanban*. Este tablero tiene un total de seis columnas:

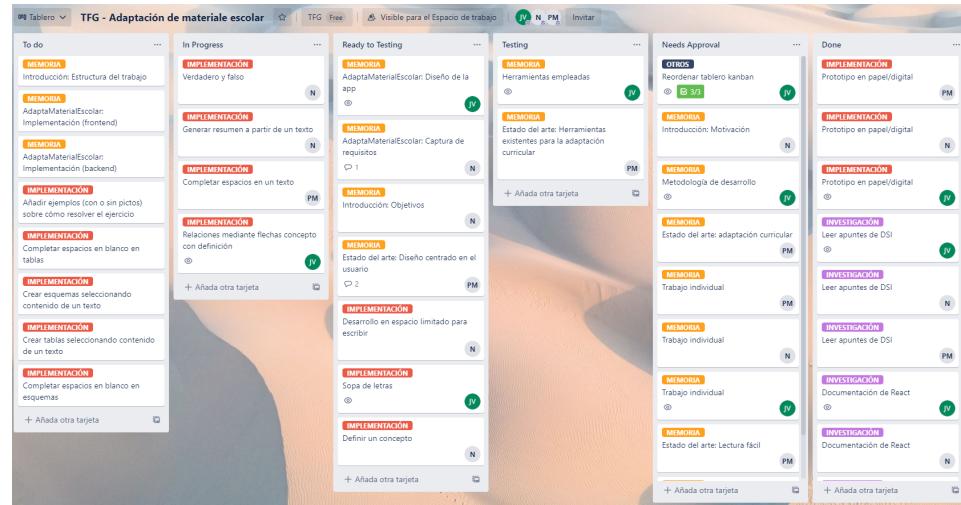


Figura 4.1: Tablero *Kanban*

- *To Do*: Se corresponden con las tareas que todavía no se han realizado.
- *In Progress*: Son tareas en las que el equipo de desarrollo ha comenzado a trabajar, por lo que se mueve la correspondiente tarea de *To Do* a esta columna.
- *Ready to Testing*: Son tareas listas para ser probadas. Una vez que se ha terminado una tarea habrá que moverla de *In Progress* a esta columna.
- *Testing*: Son tareas que se están probando. Dependiendo del tipo de tarea se tratará de una forma u otra. Para probar una tarea hay que moverla de *Ready To Testing* a esta columna. En la sección 4.3 se dan 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 y se ha acabado de hacerle las pruebas, habrá que moverla de *Testing* a esta columna.

- *Done*: Son tareas que se han dado por terminadas, validadas y aprobadas. Cuando las tutoras le han dado el visto bueno, se podrá mover de *Needs Approval* a *Done*.

Así mismo, en la Figura 4.1 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).
- Aquí se encuentran las tareas que requieren 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.

El WIP será de dos tareas por persona en cada columna (*In Progress* y *Testing*), lo que hace un total de seis tareas en cada columna.

## 4.3. Tipos de pruebas

Como tenemos dos tipos de tareas, de memoria y de implementación, cada una se tratará de una forma diferente a la hora de realizar las pruebas.

### 4.3.1. Pruebas de memoria

Las pruebas de memoria consisten en buscar errores léxicos y gramaticales en la memoria, y corregirlos antes de su entrega, además de completarla con más información en el caso de que sea posible, y comprobar que todo se entiende. Estas pruebas las hará todos los miembros del equipo de la siguiente forma: cuando haya alguna tarea de memoria en *Ready to testing*, el usuario que vaya a revisarla, se la asignará y la llevará a *Testing*. Una vez que haya terminado, y no sea la última persona en haberla revisado, la volverá a llevar a *Ready to testing*. En caso contrario, la llevará a *Needs approval*. Con este sistema se asegura completamente que todo esté correctamente escrito.

### 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* (ver sección 3.4). 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.
- Deben ser **completas**: es decir, deben cubrir la totalidad del código.
- Deben ser **independientes**: debido a que se han creado para comprobar una parte concreta del código, no deberían 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 las repeticiones.

El uso de pruebas unitarias proporcionan una serie de ventajas:

- **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 (Rodríguez, 2015):

- **Top-Down:** Es una estrategia de “arriba a abajo”, donde 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”, donde 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.
- **Ad-Hoc:** En esta estrategia el software se ve como módulos independientes, y una vez que se han implementado y validado por parte del equipo de desarrollo, se integran.
- **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.

Este proyecto seguirá la estrategia *Bottom-up*.



## Capítulo 5

# AdaptaMaterialEscolar

**RESUMEN:** En este capítulo se explicará, en la sección 5.1, para quién va a desarrollarse el proyecto de AdaptaMaterialEscolar y el proceso de captura de requisitos. En la sección 5.2, se explicará cómo se diseñó la aplicación a partir de los requisitos. Por último, en la sección 5.3, se mostrará cómo se ha implementado la aplicación.

### 5.1. Requisitos de la aplicación

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, debido a que se pretende que el proyecto sea realmente útil para los docentes, se hizo una amplia captura de requisitos con los usuarios finales, 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 los usuarios finales, tanto presenciales como online, para obtener los requisitos de la aplicación; además, se mantuvo el contacto con ellos durante todo el proceso de diseño e implementación, para irles enseñando la evolución de la aplicación.

### 5.1.1. Captura de requisitos

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 Pedagogía Terapéutica y Ana María Díaz Valle, especialista en 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 (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.

En este punto, a partir de la información proporcionada por los usuarios finales, 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 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, obtuvimos las funciona-

lidades de la aplicación. Presentamos estas funcionalidades agrupadas en: adaptaciones de temario, actividades y/o exámenes y formato.

#### Adaptaciones de temario

La motivación de las adaptaciones de temario es conseguir, en el menor tiempo posible, una unidad didáctica teórica adaptada al alumno de forma casi inmediata y automática, seleccionando el texto a convertir. En este sentido, la aplicación debe ofrecer al usuario la posibilidad de:

- Generar un resumen a partir de un texto.
- Crear esquemas que faciliten la visualización del temario y/o las actividades, seleccionando contenido de un texto ya redactado o escribiendo desde cero.
- Crear tablas que organicen el temario y/o las actividades, seleccionando contenido de un texto ya redactado o escribiendo desde cero.

#### Adaptaciones de actividades y/o exámenes:

La motivación de las adaptaciones de actividades y/o exámenes es conseguir unidades didácticas prácticas o actividades de forma fácil, seleccionando texto o un ejercicio ya redactado y adaptándolo de forma automática al tipo de ejercicio que se quiere obtener, añadiendo enunciados y ejemplos (con o sin pictogramas), siempre que sea posible, donde se explique cómo debe resolverse el ejercicio. La aplicación debe permitir crear los siguientes tipos de ejercicios:

- Ejercicios de relacionar contenido mediante flechas.
- 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 completar los espacios en blanco en tablas.
- Ejercicios de completar los espacios en blanco en esquemas.

Además, todos los ejercicios que se crean deben permitir añadir ejemplos (con o sin pictogramas) explicando cómo debe resolverse el ejercicio.

### Adaptaciones de formato:

Las adaptaciones de formato deben permitir 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. Nuestra aplicación debe permitir al usuario final que, dado un texto, pueda:

- Sustituir una palabra por un pictograma.
- Sustituir una palabra por una imagen.
- Subrayar una palabra con un color, para que los alumnos visualicen cada una de las palabras resaltadas en el texto con mayor facilidad.
- Añadir una leyenda de colores con la categoría de cada tipo de palabra resaltada, para que los alumnos puedan identificar el significado de cada una de las palabras resaltadas en el texto.
- Añadir leyenda de colores para diferenciar las asignaturas, para que los alumnos relacionen cada asignatura con un color determinado y cambien de clase sin confundirse.
- Estandarizar formato en textos, estableciendo una fuente de texto determinada y un tamaño acorde para cada alumno.
- Estandarizar formato para títulos e índices del temario, generando de forma automática una plantilla del color de la asignatura.
- Aumentar el tamaño de la fuente de una parte del texto.
- Subrayar una parte del texto.
- Poner en negrita una parte del texto.
- Añadir imágenes buscando una palabra en bases de datos de imágenes libres.

#### **5.1.2. Análisis de requisitos**

Una vez recopilados los requisitos del proyecto, redactamos una lista con éstos para que pudiéramos, tanto las profesoras del Aula TEA como el equipo de desarrollo, analizarlos de forma independiente, para no influir en la toma de decisiones sobre la importancia y dificultad de cada funcionalidad.

Enviamos el listado de requisitos a las profesoras por correo electrónico, con la finalidad de que revisaran el contenido o añadieran nuevas funcionalidades si fuera necesario y evaluaran cada uno de ellos. Las profesoras Ana María Alonso Frades y Ana María Díaz Valle debían puntuar los requisitos según la utilidad que pudieran aportar cada uno de ellos 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 todos los requisitos fueron puntuadas como imprescindible, por lo que quedó claro que el proyecto cubre una necesidad real y no añadieron ningún requisito más.

Por otro lado, cada miembro del equipo de desarrollo puntuó los requisitos de manera individual, para no influir en la decisión de los compañeros, según la dificultad que consideraban que tendría la implementación de cada requisito 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 este análisis de los requisitos era obtener un listado con las funcionalidades ordenadas por dificultad e importancia. Con la puntuación que asignaron las profesoras de forma conjunta, se obtuvo el indicador de importancia de cada tarea y, con la media de las puntuaciones del equipo de desarrollo, el indicador de dificultad.

La tabla resultante ordena, de forma descendente, las puntuaciones obtenidas de 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 prioritaria la implementación de las funcionalidades con puntuación 9 (imprescindibles y fáciles de implementar) y las menos prioritarias serán aquellas con puntuación 1 (funcionalidades poco importantes y difíciles de implementar). En la Tabla 5.1 se muestra el listado final de las funcionalidades, agrupadas por tipo de adaptación, con la media de las puntuaciones obtenidas.

El resultado final se separó en dos tablas. La primera, la Tabla 5.2, contiene las funcionalidades de la aplicación: las adaptaciones de temario (T) y actividades y/o exámenes (A), y la segunda, la Tabla 5.3, contiene las adaptaciones de formato (F) que serían necesarias en el proyecto.

### 5.1.3. Requisitos adicionales

El miércoles 4 de diciembre de 2019, el grupo de investigación NIL (Natural Interaction based on Language) organizó un workshop en la Facultad de Informática de la Universidad Complutense con docentes de otros cen-

<b>ADAPTACIONES DE TEMARIO (T)</b>		<b>Evaluación</b>	
<b>Requisito</b>		<b>Imp.</b>	<b>Dif.</b>
Generar un resumen a partir de un texto		3	2
Crear esquemas que faciliten la visualización del temario y/o las actividades, seleccionando contenido de un texto ya redactado o escribiendo desde cero		3	1,7
Crear tablas que organicen el temario y/o las actividades, seleccionando contenido de un texto ya redactado o escribiendo desde cero		2	1,7
<b>ADAPTACIONES DE ACTIVIDADES (A)</b>		<b>Evaluación</b>	
<b>Requisito</b>		<b>Imp.</b>	<b>Dif.</b>
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 completar los espacios en blanco en tablas		3	2
Ejercicios de completar los espacios en blanco en esquemas		3	1
Añadir enunciados y ejemplos (con o sin pictogramas), donde se explique cómo debe resolverse el ejercicio		2	3
<b>ADAPTACIONES DE FORMATO I (F)</b>		<b>Evaluación</b>	
<b>Requisito</b>		<b>Imp.</b>	<b>Dif.</b>
Sustituir una palabra por un pictograma		2	2,7
Sustituir una palabra por una imagen		2	2,3
Resaltar una palabra con un color		3	3
Añadir una leyenda de colores con la categoría de cada tipo de palabra resaltada, para que los alumnos puedan identificar el significado de cada una de las palabras resaltadas en el texto		1	3
Añadir leyenda de colores para diferenciar las asignaturas, para que los alumnos relacionen cada asignatura con un color determinado y cambien de clase sin confundirse		1	3
Estandarizar formato en textos, estableciendo una fuente de texto determinada y un tamaño acorde para cada alumno		3	2
Estandarizar formato para títulos e índices del temario, generando de forma automática una plantilla del color de la asignatura		3	2
Aumentar el tamaño de la fuente de una parte del texto		3	2,7
Subrayar una parte del texto		2	2,7
Poner en negrita una parte del texto		3	2,7
Añadir imágenes buscando una palabra en bases de datos de imágenes libres		3	2,3
<b>ADAPTACIONES DE FORMATO II (F)</b>		<b>Evaluación</b>	
<b>Requisito</b>		<b>Imp.</b>	<b>Dif.</b>
Añadir espacio extra entre líneas para escribir		3	2

Tabla 5.1: Puntuación de los requisitos creada por los docentes (Importancia) y los desarrolladores (Dificultad).

<b>Tipo</b>	<b>Requisito</b>	<b>Puntuación</b>
<b>A</b>	Ejercicios de verdadero o falso	<b>9</b>
<b>A</b>	Ejercicios de completar espacios en un texto	<b>8,1</b>
<b>A</b>	Ejercicios de desarrollo en un espacio limitado para escribir	<b>8,1</b>
<b>T</b>	Generar un resumen a partir de un texto	<b>6</b>
<b>A</b>	Ejercicios de relacionar contenido mediante flechas	<b>6</b>
<b>A</b>	Ejercicios de sopas de letras	<b>6</b>
<b>A</b>	Añadir ejemplos (con o sin pictogramas) sobre cómo resolver el ejercicio	<b>6</b>
<b>A</b>	Ejercicios de completar los espacios en blanco en tablas	<b>6</b>
<b>T</b>	Crear esquemas que faciliten la visualización del temario y/o las actividades, seleccionando contenido de un texto ya redactado o escribiendo desde cero	<b>5,1</b>
<b>T</b>	Crear tablas que organicen el temario y/o las actividades, seleccionando contenido de un texto ya redactado o escribiendo desde cero	<b>3,4</b>
<b>A</b>	Ejercicios de completar los espacios en blanco en esquemas	<b>3</b>

Tabla 5.2: Lista de funcionalidades relacionadas con adaptaciones de temario (T) y crear actividades (A), ordenada por prioridad.

<b>Tipo</b>	<b>Requisito</b>	<b>Puntuación</b>
<b>F</b>	Resaltar una palabra con un color	<b>9</b>
<b>F</b>	Aumentar el tamaño de la fuente de una parte del texto	<b>8,1</b>
<b>F</b>	Poner en negrita una parte del texto	<b>8,1</b>
<b>F</b>	Añadir imágenes buscando una palabra en bases de datos de imágenes libres	<b>6,9</b>
<b>F</b>	Añadir enunciados y ejemplos (con o sin pictogramas), donde se explique cómo debe resolverse el ejercicio	<b>6</b>
<b>F</b>	Estandarizar formato para títulos e índices del temario, generando de forma automática una plantilla del color de la asignatura	<b>6</b>
<b>F</b>	Estandarizar formato en textos, estableciendo una fuente de texto determinada y un tamaño acorde para cada alumno	<b>6</b>
<b>F</b>	Añadir espacio extra entre líneas para escribir	<b>6</b>
<b>F</b>	Sustituir una palabra por un pictograma	<b>5,4</b>
<b>F</b>	Subrayar una parte del texto	<b>5,4</b>
<b>F</b>	Añadir una leyenda de colores con la categoría de cada tipo de palabra resaltada, para que los alumnos puedan identificar el significado de cada una de las palabras resaltadas en el texto	<b>3</b>

Tabla 5.3: Lista de funcionalidades de adaptación de formato, ordenada por prioridad.

tros y asociaciones. Durante este workshop se mostraron las herramientas tecnológicas inclusivas que se habían desarrollado o que se encontraban en desarrollo, siendo una de las presentadas AdaptaMaterialEscolar.

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 y expertos, para que pudieran expresar qué les había parecido la propuesta. Todos los comentarios fueron muy positivos y preguntaron cuándo estaría disponible para uso público. Además de dar un feedback positivo sobre el proyecto, también propusieron un nuevo tipo de adaptación de formato, para mejorar las opciones de personalización para los alumnos. El requisito consiste en aportar un espacio lo suficientemente grande, ampliando la distancia entre los renglones si es necesario, para que los alumnos que tienen una letra de mayor tamaño puedan escribir sin problema.

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. El requisito se encuentra anexado en la Tabla 5.1 y en la Tabla 5.3 se puede ver la puntuación.

## 5.2. Diseño de la aplicación

El diseño de la aplicación web se ha desarrollado en varias iteraciones. En la primera iteración se creó un boceto de la aplicación en papel. En la segunda iteración se mejoró ese boceto, creando un prototipo por vistas a través de la herramienta online *Moqups* (ver sección 3.1). En la tercera iteración cada uno de los miembros del equipo creó su propio prototipo, sin influir los unos con los otros, para ver si el prototipo creado con *Moqups* podía ser mejorado. En la última iteración se realizó el diseño final de la aplicación. En las siguientes secciones se dan más detalles sobre estas iteraciones.

### 5.2.1. Primera iteración: boceto inicial

Para comenzar el proceso de diseño se creó un boceto en papel de la página principal de la aplicación (Figura 5.1). En la parte superior de este

diseño, se observa que hay una barra de navegación con dos menús: Texto y Actividades. Cada uno de ellos tiene un submenú con las diferentes adaptaciones disponibles para dicho menú (en la sección 5.1.1 se pueden ver las diferentes adaptaciones que habrá en la aplicación). En el 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.



Figura 5.1: Boceto inicial de la aplicación

### 5.2.2. Segunda iteración: versión 1.0 del prototipo

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 barras de navegación con diferentes menús. En la barra de navegación superior, podemos ver los menús relacionados con un editor de texto: Archivo, Edición, Formato, Insertar y Ayuda. En la barra de navegación inferior se encuentran los menús correspondientes a las adaptaciones de Temario y Actividades, además

de la posibilidad de buscar imágenes libres<sup>1</sup> (también llamadas imágenes *CC0* o *Creative Commons Zero*<sup>2</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.



Figura 5.2: Versión 1.0 de la página principal

- **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 podrán seleccionar las partes del texto que se quieran adaptar. El elemento de la derecha es el editor donde se podrá, también, adaptar e insertar imágenes, pictogramas, dar formato, etc.
- **Búsqueda de pictogramas:** En la Figura 5.4 se puede ver la búsqueda de pictogramas. En la parte superior derecha de la aplicación aparece una ventana con los diferentes pictogramas que se obtienen al buscar la palabra “Perro”. Previamente, para poder realizar la búsqueda de pictogramas, se ha tenido que seleccionar el campo Pictos, que

<sup>1</sup>Las imágenes libres son aquellas imágenes que son de dominio público y que no tienen derechos reservados.

<sup>2</sup><https://creativecommons.org/publicdomain/zero/1.0/deed.es>

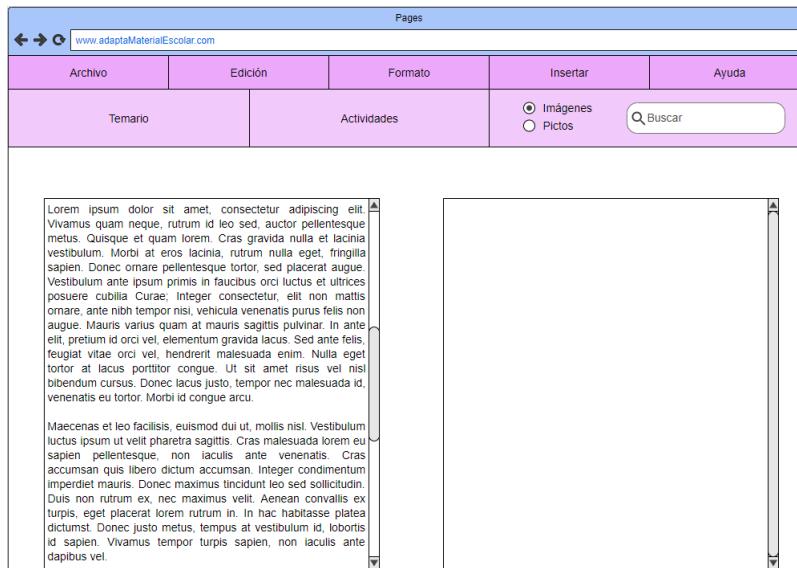


Figura 5.3: Versión 1.0 del editor

está a la izquierda de la barra de búsqueda. Si se hace clic en uno de ellos automáticamente se pondrá en el campo del editor.

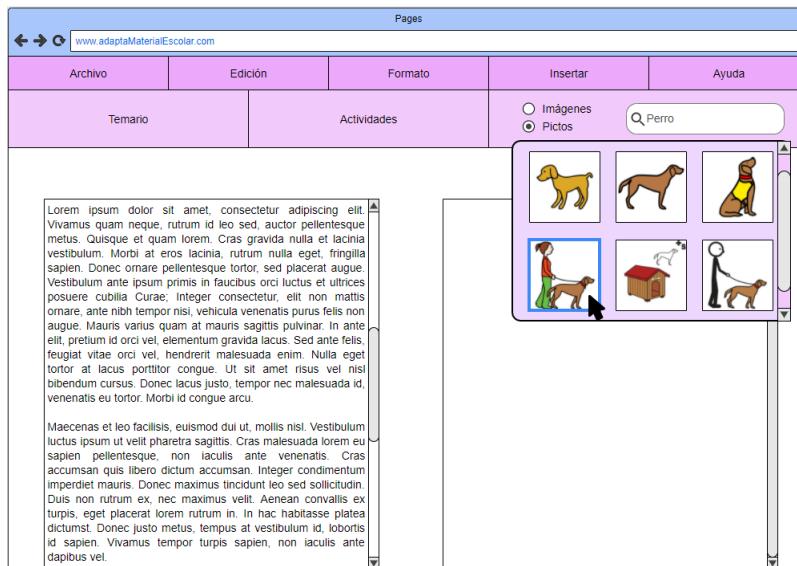


Figura 5.4: Versión 1.0 de la búsqueda de pictogramas

- **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.

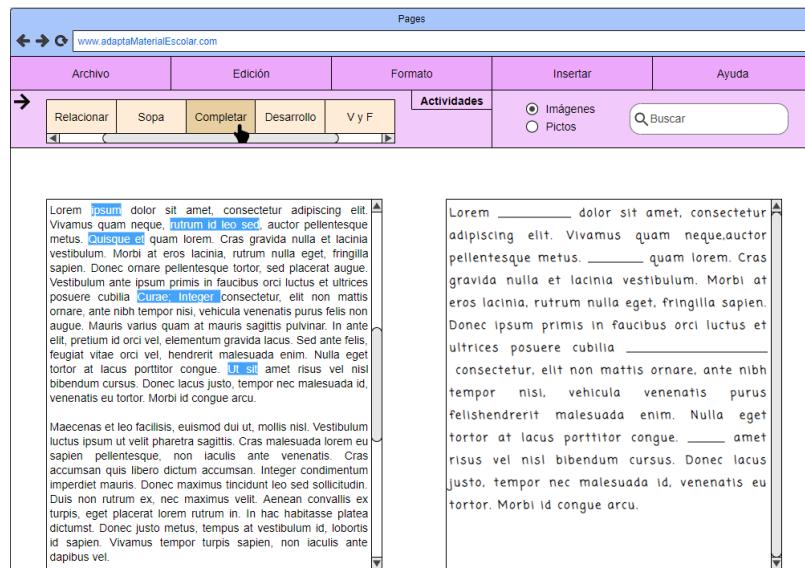


Figura 5.5: Versión 1.0 de la adaptación de actividades

- **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 iteración: iteración competitiva

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 diferen-

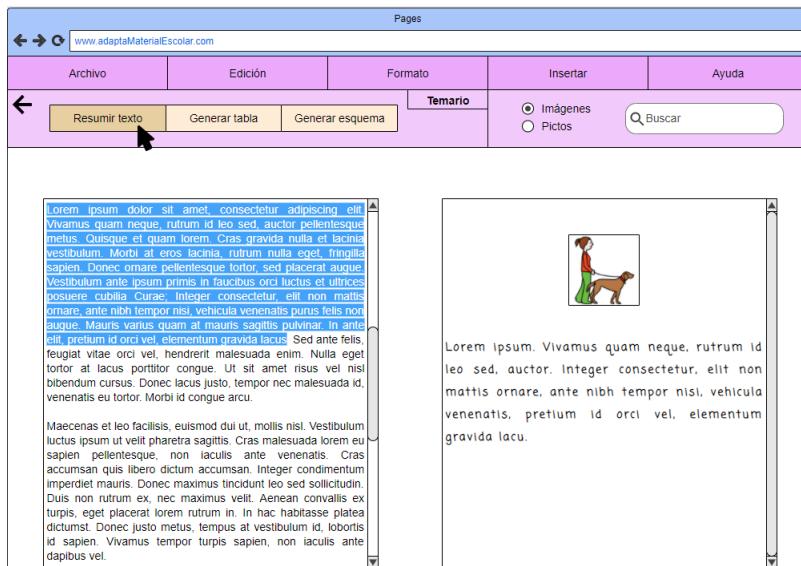


Figura 5.6: Versión 1.0 de la adaptación de temario

tes puntos de vista sobre una misma funcionalidad. Una vez que todos los prototipos fueron diseñados (dos de forma digital y otro en papel) se realizó una puesta en común para analizar las distintas propuestas. Los resultados fueron los siguientes:

- **Página principal:** 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 están los créditos y los logotipos de ARASAAC<sup>3</sup>, la Universidad Complutense de Madrid<sup>4</sup> y la licencia de *Creative Commons*<sup>5</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, en la parte inferior de dicha figura coincidimos que sería buena idea establecer algún 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 usuario 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

<sup>3</sup><https://arasaac.org/>

<sup>4</sup><https://www.ucm.es>

<sup>5</sup><https://creativecommons.org/>

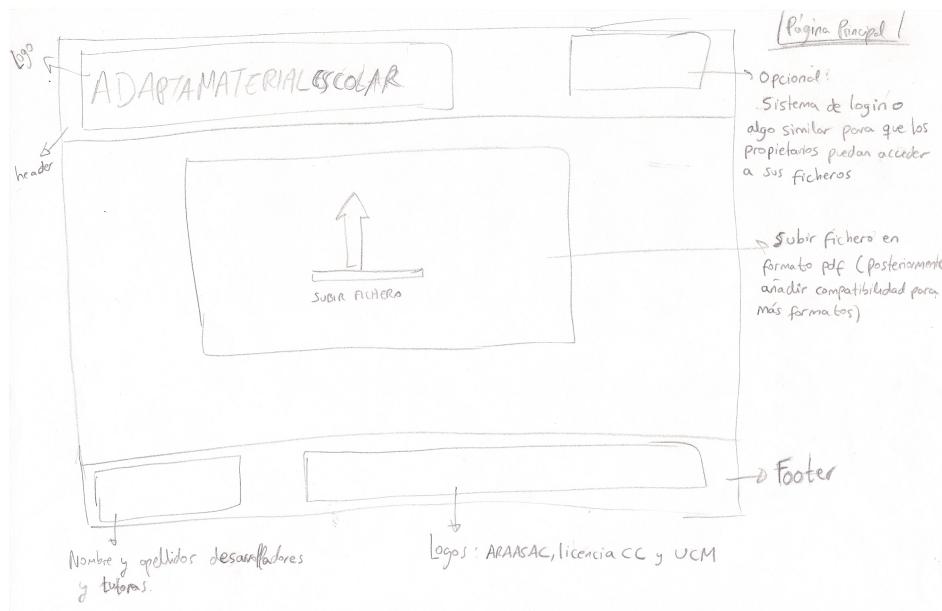


Figura 5.7: Prototipo de la página principal (Jorge)



Figura 5.8: Prototipo de la página principal (Natalia)

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 las Figuras 5.10, 5.13 y 5.14 se pueden ver los resultados



Figura 5.9: Prototipo de la página principal (Pablo)

de la página del editor. En el caso de las Figura 5.10, observamos una barra de navegación con diferentes menús, que representa los tipos de adaptaciones que tendrá en la aplicación (véase sección 5.1.1), junto con la búsqueda de imágenes y pictogramas. Si se pasa el ratón por encima de 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.

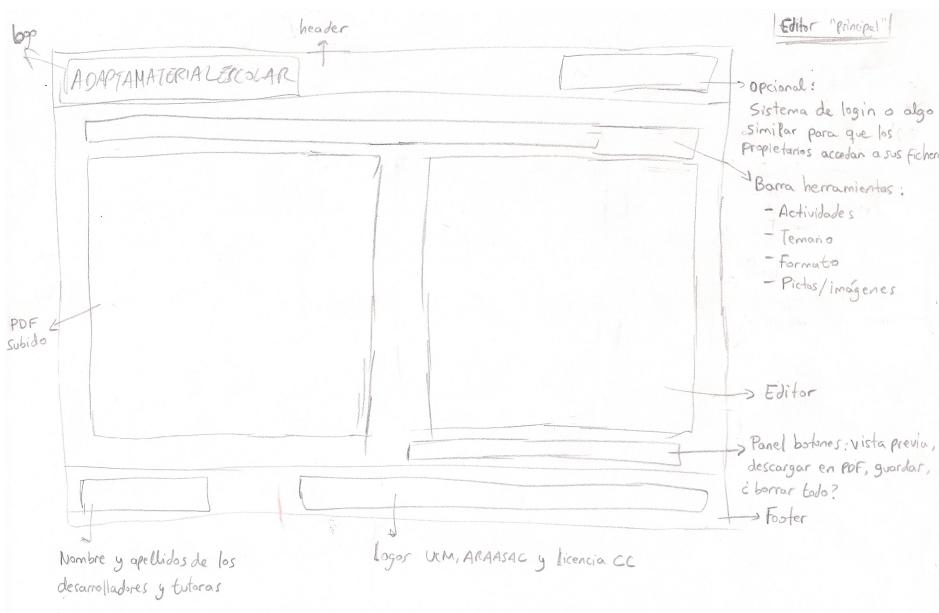


Figura 5.10: Prototipo del editor (Jorge)

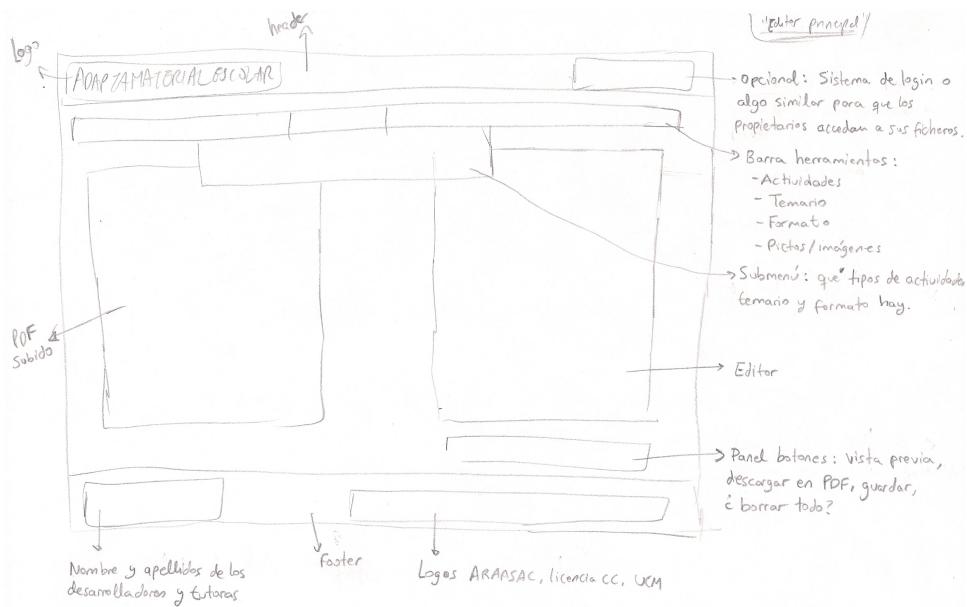


Figura 5.11: Prototipo del editor con un desplegable en uno de los menús (Jorge)

Así mismo, se ven 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, 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 y 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, co-

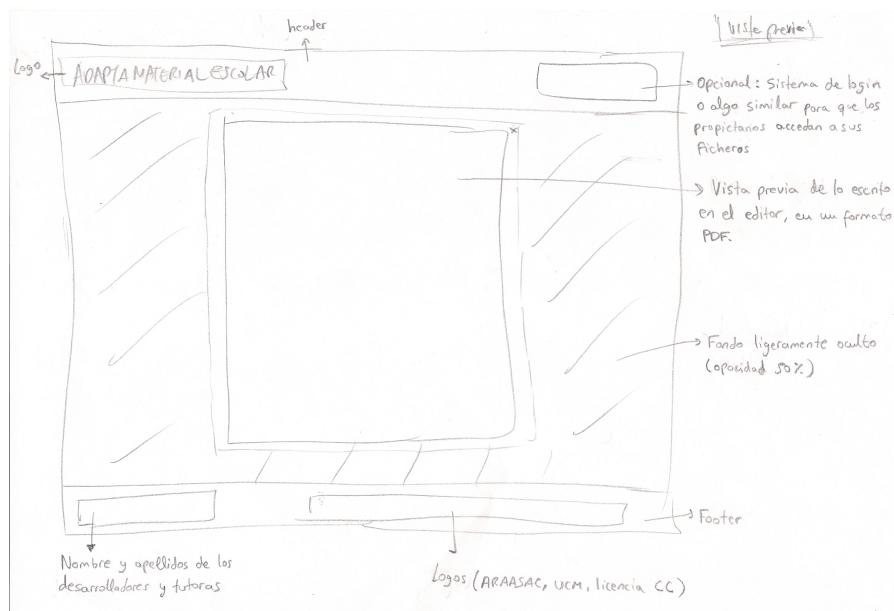


Figura 5.12: Prototipo de la vista previa (Jorge)

Navigator

www.AdaptaMaterialEscolar.com

Archivo Formato Temario Actividades Imágenes/Pictos Créditos Ayuda

Célula

Para otros usos de este término, véase [Célula](#) (desambiguación).

Micrografía al microscopio electrónico de barrido de células de *Escherichia coli*. La célula (del latín *cellula*, diminutivo de *cella*, 'celda')<sup>1</sup> es la unidad morfológica y funcional de todo ser vivo.<sup>2</sup> De hecho, la célula es el elemento de menor tamaño que puede considerarse vivo.<sup>2</sup> De este modo, puede clasificarse a los organismos vivos según el número de células que posean: si solo tienen una, se les denomina unicelulares (como pueden ser los protistas o las bacterias, organismos microscópicos); si poseen más, se les llama pluricelulares. En estos últimos el número de células es variable: de unos pocos cientos, como en algunos nematodos, a cientos de billones (10<sup>14</sup>), como en el caso del ser

La célula

La célula es la unidad morfológica y funcional de todo ser vivo. Las células suelen poseer un tamaño de 10 µm y una masa de 1 ng, si bien existen células mucho mayores.

Figura 5.13: Prototipo del editor (Natalia)

lor, negrita, cursiva, etc), así como deshacer y rehacer lo escrito, crear tablas, y unos botones con los distintos tipos de adaptaciones que hay.

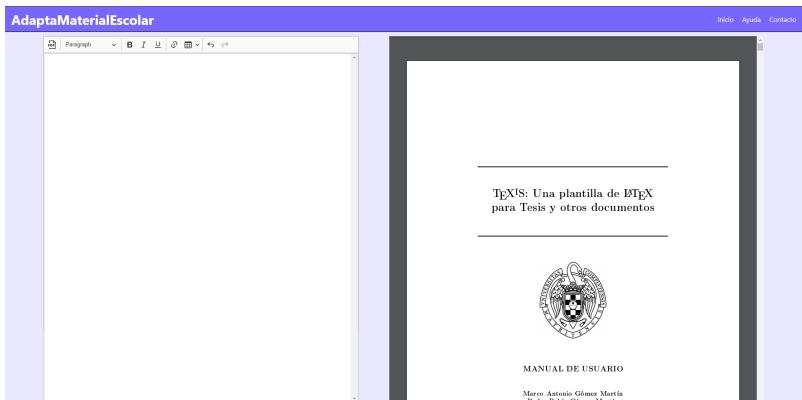


Figura 5.14: Prototipo del editor (Pablo)

#### 5.2.4. Cuarta iteración: diseño final

Una vez hecho la puesta en común de los diseños realizados por cada miembro del equipo se procedió a hacer el diseño final. Este diseño tomaba como base la Figura 5.14, debido a que estaba todo más limpio y claro visualmente, y a la que se le añadió una barra de navegación en la parte superior (tal y como aparece en las Figuras 5.10 y 5.13) para los tipos de adaptaciones que habrá en la aplicación. Esta barra de navegación surge porque algunas adaptaciones tienen un estilo de implementación distinto (algunas se podrán hacer sobre el propio editor pero otras abrirán nuevas ventanas para poder realizar la adaptación, sobre todo en las adaptaciones de actividades).

Tras realizar este diseño, se envió un email a las profesoras del Aula TEA para que nos dieran *feedback* sobre él. La respuesta fue positiva: tanto la distribución de los elementos como los colores usados eran adecuados. Así mismo, nos propusieron una nueva funcionalidad: permitir subrayar un texto con fluorescente.

### 5.3. Implementación

AdaptaMaterialEscolar es una aplicación web creada con React (ver sección 3.2), cuyo código se puede encontrar en el repositorio del grupo NIL en GitHub<sup>6</sup>. Es accesible a través de la siguiente url: <https://holstein.fdi.uned.es/>

<sup>6</sup><https://github.com/NILGroup/TFG-AdaptaMaterialEscolar/tree/master/Codigo>

[ucm.es/tfg/2021/adapta/](http://ucm.es/tfg/2021/adapta/). En la Figura 5.15 se muestra la página principal de la aplicación. En ella se puede ver el header en la parte superior, con el logo de la aplicación en la parte de la izquierda, y los botones de Inicio, Ayuda y Contacto en la parte de la derecha; y en el centro de la página una zona para poder subir el fichero .pdf. Cuando éste se ha subido, aparecerán dos zonas diferenciadas: la zona de la izquierda contiene el documento subido y la zona de la derecha es la zona que contiene las funcionalidades y el editor con su barra de herramientas, como se puede ver en la Figura 5.16.



Figura 5.15: Página principal de AdaptaMaterialEscolar

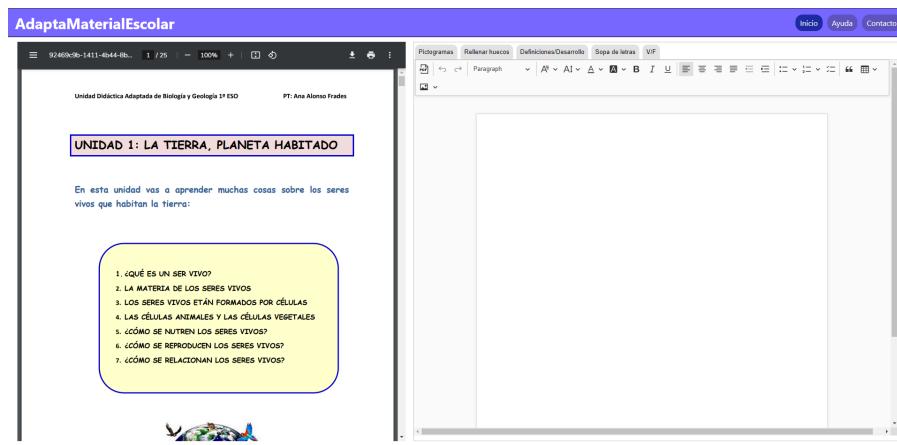


Figura 5.16: Página principal de AdaptaMaterialEscolar tras subir un documento

Las funcionalidades implementadas se darán con más detalle en la sección 5.3.2 y son las siguientes:

- Pictogramas. Permite buscar pictogramas correspondientes al texto introducido. Haciendo clic en un pictograma hace que éste se lleve al editor.
- Rellenar huecos. Permite crear un ejercicio de completar con palabras los huecos, a partir de un texto inicial, haciendo clic en las palabras en las que se quiera dejar el hueco.
- Definiciones/Desarrollo. Definiciones permite crear una lista de conceptos a definir o preguntas a contestar en un número máximo de líneas. Desarrollo permite crear un ejercicio de desarrollo insertando el enunciado en un número de líneas máximo. En ambos, el número de líneas lo establece el usuario y también tiene una opción para añadir espacio extra entre líneas para estudiantes que tengan una letra más grande.
- Sopa de letras. Permite crear una sopa de letras estableciendo el número de filas, columnas, las palabras que se quiere insertar y las direcciones cardinales para buscar la palabra. También se puede establecer la escritura al revés de cada palabra y mostrar en el ejercicio las palabras a buscar.
- Verdadero y falso. Permite crear una lista de oraciones y añade al final un recuadro para contestar con una V si es verdadero o con una F si es falso.

Todos los ejercicios tienen una opción para añadir un ejemplo de cómo resolver el ejercicio, y en todos ellos se añade un enunciado plantilla automáticamente, que posteriormente el usuario puede personalizar a su gusto. Al hacer clic en una de las funcionalidades, se abrirá, justo debajo de la barra de funcionalidades, una ventana móvil manteniendo el clic izquierdo pulsado en la parte superior de dicha ventana. En la Figura 5.17 se puede ver el mensaje que sale al pasar el ratón por encima de la parte superior de la ventana en el que informa al usuario de que puede moverla; y se ve que ésta se ha movido de sitio.

### 5.3.1. Arquitectura

En el proyecto de este TFG se ha decidido aplicar una arquitectura *Redux*<sup>7</sup>. Redux es un patrón de arquitectura de datos que pretende disminuir el número de relaciones entre los componentes de la aplicación. Se trata de un patrón que se adapta a todo tipo de librería o framework del lado del cliente.

---

<sup>7</sup><https://redux.js.org/>

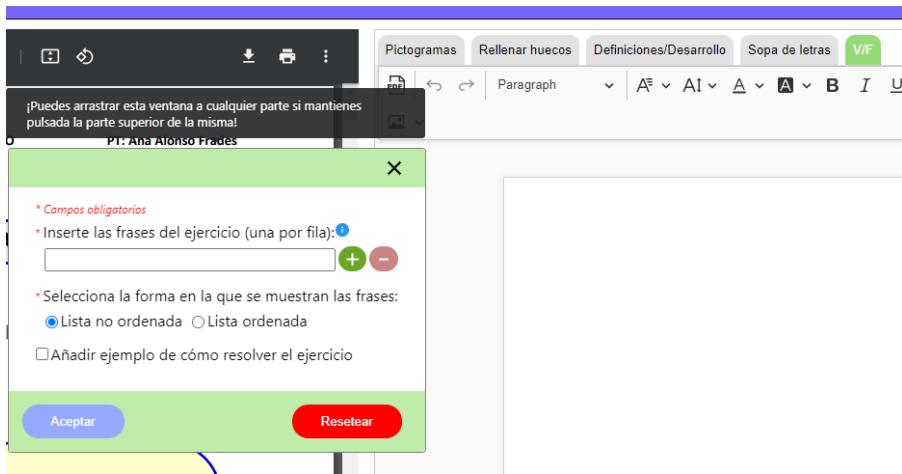


Figura 5.17: Ejemplo de ventana móvil

La figura 5.18 muestra un diagrama mostrando el flujo de una aplicación Redux.

Esta arquitectura se basa en tres principios:

- Única fuente de la verdad. El estado de la aplicación se guarda en un único store y cuando los componentes necesiten conocer el estado lo cogerán del store. Este store es un objeto Javascript y está estructurado en forma de árbol con todos los datos necesarios para la aplicación.
- El estado es de sólo lectura. No podemos modificar el estado directamente, solo podremos leer de él para mostrarlo en la vista y si queremos modificarlo tendremos que ejecutar acciones. Las acciones son un objeto Javascript que incluye al menos un atributo type que indica el tipo de acción que estamos emitiendo y en caso de que haya datos asociados al cambio o modificación, un atributo payload con esos datos. Por ejemplo, en el siguiente fragmento de código se tiene una acción para cargar un documento, donde el atributo type indica el tipo de acción a realizar y el payload llevará los datos del documento a cargar:

```
export const loadDocument = (document) => ({
  type: DocumentActionTypes.LOAD_FILE,
  payload: document
});
```

- Cambios con funciones puras. Ya que el estado no se puede modificar directamente y está almacenado en un store, tenemos que ejecutar

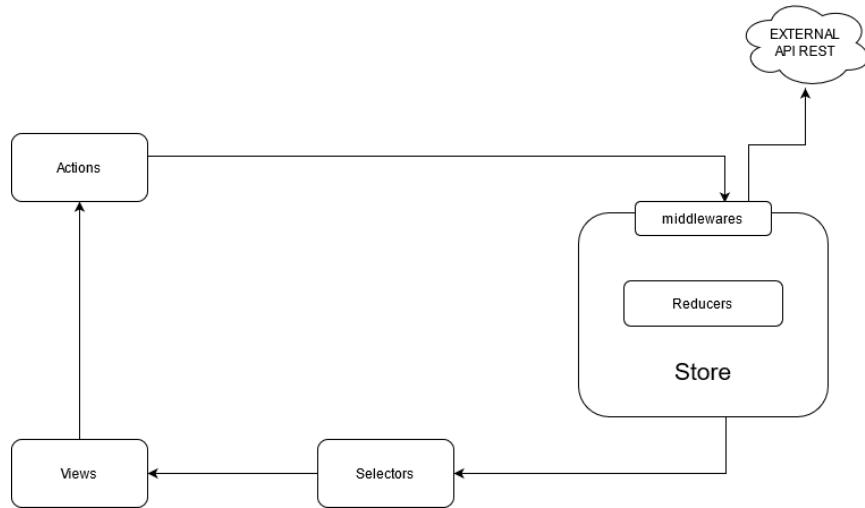


Figura 5.18: Muestra de flujo

acciones puras llamadas reducers para modificarlo. Un reducer es una función que recibe dos parámetros, el estado inicial y una acción. Dentro contendrá un switch-case que, en base a la acción que reciba, ejecutará los cambios necesarios en el store. El siguiente fragmento muestra un reducer del módulo documentos. En caso de recibir una acción del tipo LOAD\_FILE, almacenará el documento recibido del payload de la acción y lo guardará en el store:

```
const documentReducer = (state = INITIAL_STATE, action) => {
  switch (action.type){
    case DocumentActionTypes.LOAD_FILE:
      return {
        ...state,
        file: action.payload.file,
        fileIsLoaded: true
      };
    default:
      return state;
  }
};
```

Al externalizar toda la información en un store, facilitamos el acceso y modificación de datos en la aplicación. En la figura 5.19 se muestran las diferencias de flujo de información entre los componentes de una aplicación web sin re-

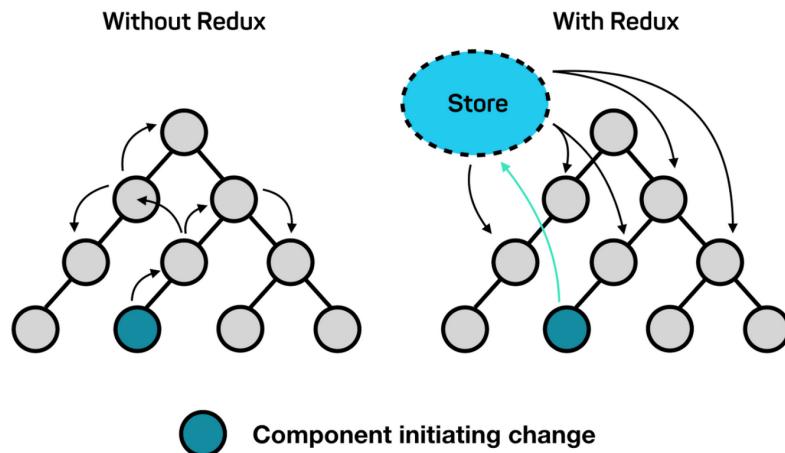


Figura 5.19: Comparación de flujo de información de componentes sin y con redux. Fuente: <https://blog.codecentric.de/en/2017/12/developing-modern-offline-apps-reactjs-redux-electron-part-3-reactjs-redux-basics/>

dux y una aplicación con redux.

En redux, el flujo de datos es unidireccional (ver figura 5.18). Se siguen los siguientes pasos:

- Los componentes se suscriben a los datos del store mediante selectores. Estos selectores actúan como getters, es decir, funciones de sólo lectura de datos. Cuando el store modifique uno de esos valores, los componentes a los que estén suscritos se actualizarán automáticamente. Se recurre a una librería de npm denominada *reselect*<sup>8</sup> que simplifica estas implementaciones. Mediante reselect ejecutamos una función denominada createSelector, que internamente realiza la suscripción al dato del store que queremos. En el siguiente fragmento se crea un selector para obtener el documento cargado en el store. Se selecciona el módulo document del store y creamos un selector que obtenga el fichero:

```
const selectDocumentStore = (store) => store.document;

export const selectDocumentFile = createSelector(
  [selectDocumentStore],
  documentStore => documentStore.file
);
```

<sup>8</sup><https://www.npmjs.com/package/reselect>

En los componentes se inyectarán los selectores necesarios mediante una función llamada `createStructuredSelector`. Por ejemplo, en el componente que muestra un documento:

```
const mapStateToProps = createStructuredSelector({
  document: selectDocumentFile
});
```

- Los componentes ejecutan acciones. Esto lo realizan ejecutando un método `dispatch` y añadiendo como parámetro la acción a ejecutar. Por ejemplo, si tenemos una acción de cargar un documento, haríamos lo siguiente:

```
export const closePictogramSearchModal = () => ({
  type: PictogramsActionTypes.CLOSE_PICTOGRAMS_MODAL
});
```

- Se ejecutan middlewares en caso de necesitar acciones asíncronas. Estos middlewares permiten ejecutar llamadas a API REST y posteriormente despachar otras acciones según el resultado. El siguiente ejemplo muestra cómo se ejecuta un middleware de buscar pictogramas llamando a la API de ARASAAC:

```
export const fetchPictograms = (searchInput) => {
  return dispatch => {
    fetch(`https://api.arasaac.org/api/pictograms/es/search/${searchInput}`)
      .then((response) => response.json())
      .then(data => {
        let items = [];
        for(let i = 0; i < data.length && i < 20; i++){
          items.push(`https://static.arasaac.org/pictograms/${data[i]._id}/${data[i]._id}.png`)
        }
        dispatch(fetchPictogramsSuccess(items));
      })
      .catch(error){
        dispatch(fetchPictogramsFailure(error))
      };
    }
  }
}
```

- El store de redux invoca a la función reductora solicitada por la acción. El store le pasará dos argumentos a la función: estado actual y acción. El switch dentro del reductor encontrará la acción solicitada y actualizará los datos en base a ella. Los selectores detectarán el cambio y actualizarán las vistas.

### 5.3.2. Funcionalidades

En esta sección se explicarán con detalle la implementación de cada una de las funcionalidades.

#### 5.3.2.1. Sopa de letras

En la Figura 5.20 se puede ver el aspecto de la aplicación al seleccionar sopa de letras. La creación del ejercicio de sopa de letras solicita los siguientes parámetros:

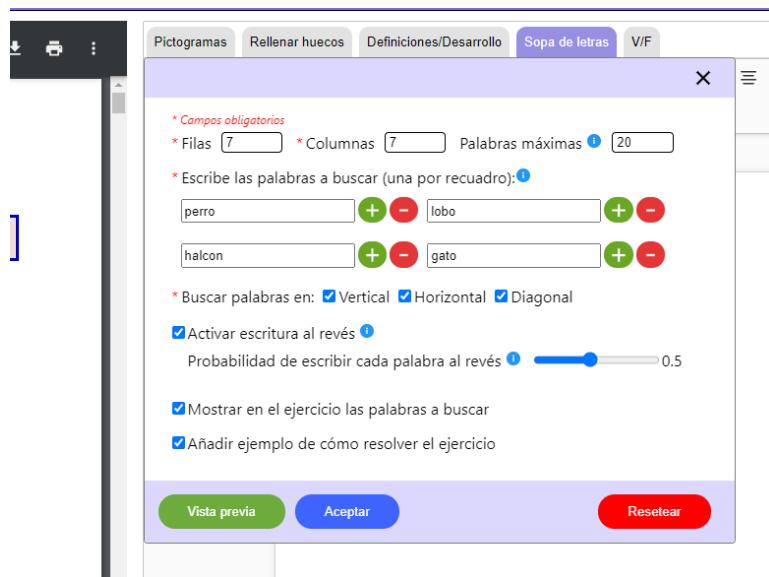


Figura 5.20: Generación de la sopa de letras en AdaptaMaterialEscolar

- Filas: El número de filas que tendrá la sopa de letras.
- Columnas: El número de columnas que tendrá la sopa de letras.
- Palabras: Las palabras que se insertarán en la sopa de letras.
- Dirección: Las direcciones en las que se quiere insertar la palabra. Como mínimo se debe activar una dirección para poder generar la sopa de letras.

También se ha añadido otra opción de carácter no obligatorio:

- Activar escritura al revés: Activa la probabilidad de escribir cada palabra al revés. En el caso de que esta opción no se active, la probabilidad será cero. Si se activa, aparecerá una barra deslizante para establecer dicha probabilidad.

Así mismo, existe otra opción de carácter opcional no relacionada con la creación de la sopa de letras y es la posibilidad de mostrar en el editor las palabras a buscar junto al enunciado.

Una vez que se han llenado los campos obligatorios (filas, columnas, palabras y dirección) se activarán los botones de “Vista previa” y “Aceptar”. El botón Vista previa genera primero la sopa de letras y la muestra al usuario antes de escribirla en el editor (en la Figura 5.21 se muestra la vista previa de la sopa de letras usando la configuración de la Figura 5.20). En cambio, el botón Aceptar genera la sopa de letras y la escribe directamente en el editor, sin mostrarla antes al usuario. En ambos casos, si ocurre algún error (por ejemplo, que el número de filas y columnas sea igual o menor que cero) se mostrará al usuario el mensaje de error correspondiente.

Añadir ejemplo de cómo resolver el ejercicio



Figura 5.21: Vista previa de la sopa de letras

En la Figura 5.22 se muestra el diagrama de secuencia de esta funcionalidad. Tal y como puede verse en dicho diagrama, al pulsar aceptar ocurre lo siguiente:

1. Se hace una llamada al despachador pasándole como acción `updateForceReadyToPreview(false)`, para ocultar la vista previa en caso de que esté visible.
2. Hay que distinguir dos casos fundamentales:

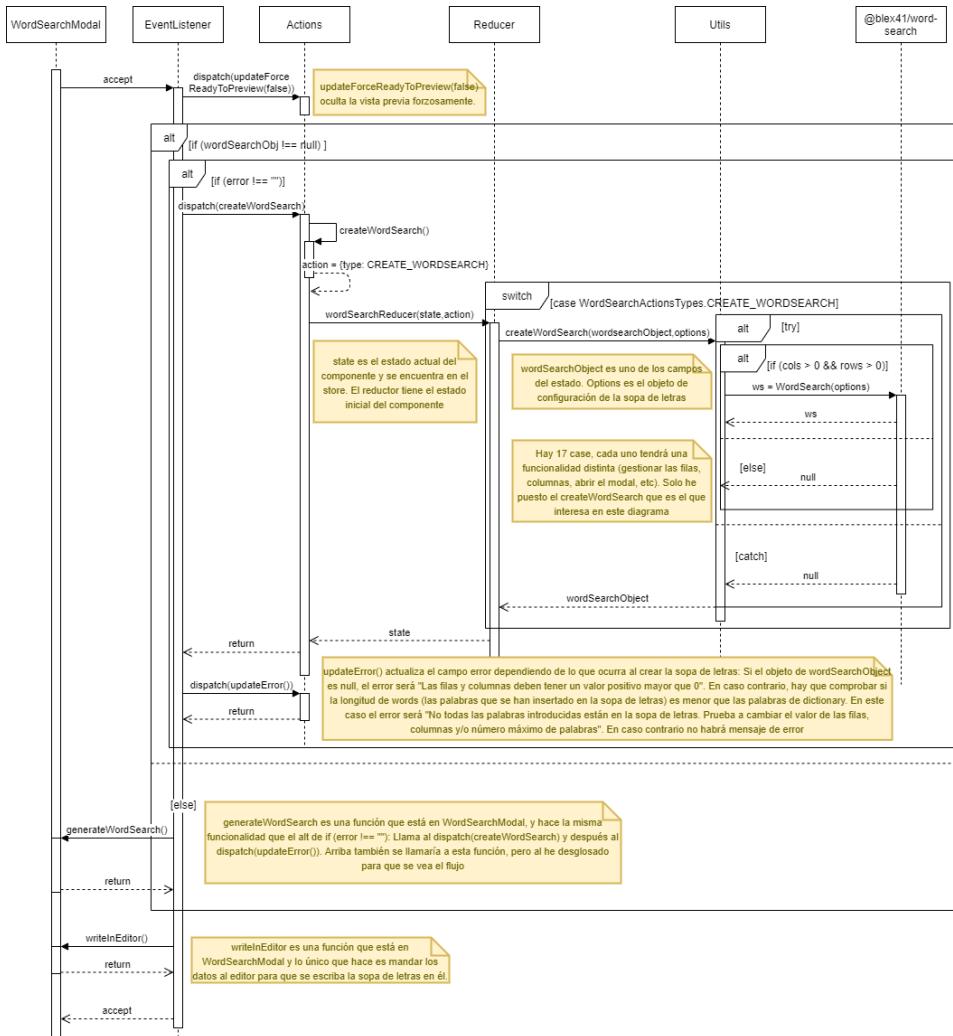


Figura 5.22: Diagrama de secuencia de la generación de la sopa de letras

- Si ya hay una sopa de letras creada, es decir, el objeto `WordSearchObject` es distinto de `null`, significa que anteriormente se ha pulsado el botón de Vista Previa, o el botón Aceptar pero hay un error. En este caso, si hay un mensaje de error se hace una llamada a `generateWordSearch()`.
- Si no hay una sopa de letras creada, es decir, el objeto `WordSearchObject` es `null`, llamaremos directamente a `generateWordSearch()`.

3. La función `generateWordSearch()` consta de dos fases:

- a) Se llama al despachador pasándole como acción `createWordSearch()`.

El despachador buscará en *Actions* el tipo de acción que es (en este caso, *CREATE\_WORDSEARCH*). Posteriormente llamará al reductor a través de *wordSearchReducer()*, pasándole como parámetros el estado (*state*) del componente y la acción buscada anteriormente. A continuación, buscará en el *switch* el tipo de acción y se procederá a crear la sopa de letras llamando a *createWordSearch()* pasándole como parámetros el objeto *wordSearchObject* y el objeto de configuración de la sopa de letras, *options*. Esta función devolverá una sopa de letras en caso de que las filas y columnas sean mayores que cero. En caso contrario, o si ha habido alguna excepción, devolverá null. En la Figura 5.23 se puede ver un diagrama con el flujo de acción, reductor, etc., concretos para este caso.

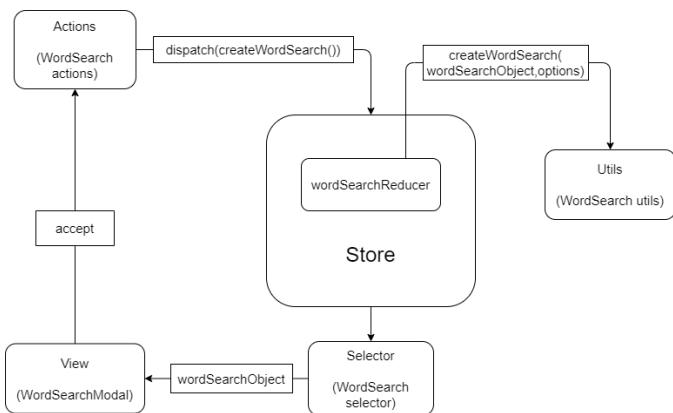


Figura 5.23: Diagrama de flujo de la generación de la sopa de letras

- Se llama al despachador pasándole como acción *updateError()*. El despachador buscará en *Actions* el tipo de acción que es (que en este caso es *UPDATE\_WORDSEARCH\_ERROR*). Después llamará al reductor donde buscará el tipo de acción en el *switch* y actualizará el mensaje de error llamando a la función *manageError()* pasándole como parámetros el error, el objeto *wordSearchObject* y la longitud del array *dictionary*. El error solo puede tomar tres estados:
  - Si el objeto *wordSearchObject* es null, el error será “Las filas y columnas deben tener un valor positivo mayor que 0”.
  - Si el objeto *wordSearchObject* es distinto a null:
    - Hay que ver si la longitud de *words* (una de las funciones de la sopa de letras, que se puede leer en la sección 3.6) es menor que la longitud del array *dictionary*. En este caso, el error será “No todas las palabras introducidas están en”.

la sopa de letras. Prueba a cambiar el valor de las filas, columnas y/o número máximo de palabras”.

- En caso contrario, no habrá mensaje de error.
4. Por último, se hará una llamada a *writeInEditor()*, una función local de *WordSearchModal*, cuyo objetivo es escribir en el editor la sopa de letras, además de resetear los datos del modal y cerrarlo.

En la Figura 5.24 se muestra el resultado de la generación de la sopa de letras si se usa como configuración los datos de la Figura 5.20.

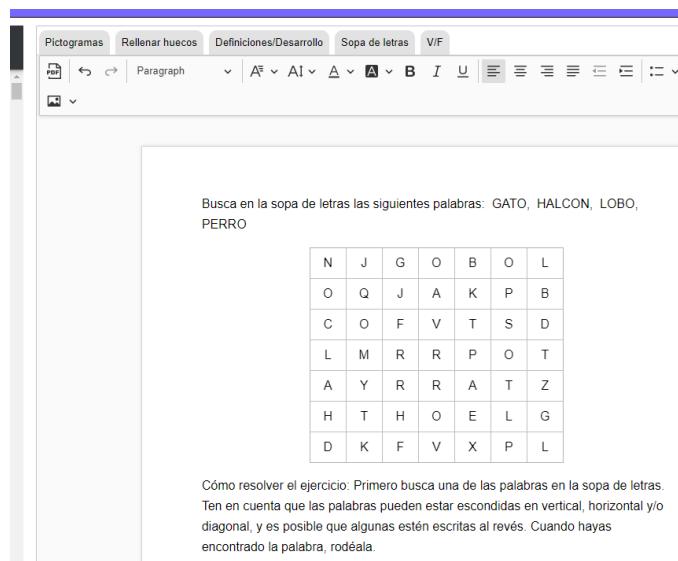


Figura 5.24: Resultado de la generación de la sopa de letras

### 5.3.2.2. Definiciones/Preguntas

Esta opción inicialmente fue pensada para permitir añadir ejercicios para que los estudiantes pudieran definir conceptos, pero durante la implementación vimos la opción de aumentar la funcionalidad planteando otros tipos de ejercicios, como, por ejemplo, preguntas cortas para que el alumnado tenga que responder argumentando.

Al seleccionar esta opción en nuestra aplicación se muestra la interfaz de la Figura 5.25. Para crear este tipo de ejercicio, hay que llenar los siguientes campos:

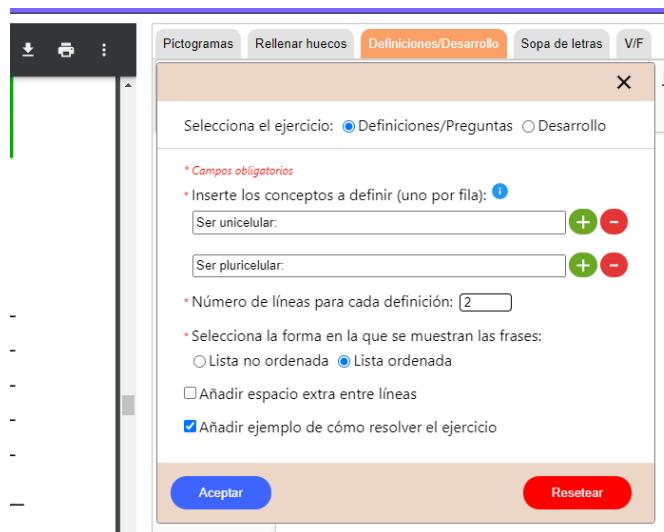


Figura 5.25: Interfaz del ejercicio de definiciones/preguntas

- Conceptos a definir o preguntas a contestar: Es un array de strings que inicialmente está vacío, y contiene los conceptos a definir y/o preguntas a contestar. Dentro de los inputs, si se pulsa a la tecla enter, se podrá introducir más conceptos o preguntas (también se puede hacer clic en el botón “más” para añadir. Para quitar alguna fila, se tendrá que hacer clic en el botón “menos”).
- Número de líneas: número máximo de líneas para definir el concepto o contestar la pregunta. Por defecto es 1.
- La forma en la que se muestra las frases: puede mostrarse como una lista no ordenada o una lista ordenada. Por defecto está establecido en mostrarse como una lista no ordenada.

Así mismo, se ha incluido una opción para añadir espacio extra entre líneas para los estudiantes que tengan una letra más grande.

Una vez que se han rellenado los campos obligatorios, el botón Aceptar se habilitará. Al hacer clic en él se procederá a escribir en el editor cada concepto y/o pregunta, seguido del número de líneas especificado por el usuario. Así mismo, se reseteará el modal y se cerrará. En la Figura 5.27 se muestra el resultado de este ejercicio al usar las opciones de la Figura 5.25 en la parte superior. La parte inferior es el resultado de usar las opciones de la Figura 5.26

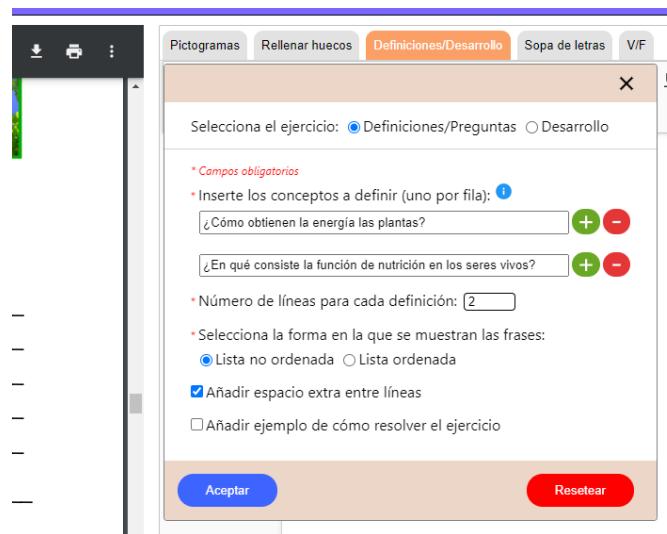


Figura 5.26: Interfaz del ejercicio de definiciones/preguntas con espacio extra activado

The screenshot shows the software editor displaying the results of the exercise configuration from Figure 5.26. The interface includes a toolbar with various text and paragraph styles. The main content area contains two sections for defining concepts:

- Section 1:** 'Define los siguientes conceptos.' with two numbered lines for answers:
  1. Ser unicelular: \_\_\_\_\_
  2. Ser pluricelular: \_\_\_\_\_
- Section 2:** 'Cómo resolver el ejercicio. Tienes que definir cada concepto usando como máximo 2 líneas para cada uno de ellos (no es necesario llenar todas las líneas)' followed by another 'Define los siguientes conceptos.' section with two numbered lines for answers:
  - ¿Cómo obtienen la energía las plantas? \_\_\_\_\_
  - ¿En qué consiste la función de nutrición en los seres vivos? \_\_\_\_\_

Figura 5.27: Resultado del ejercicio de definiciones/preguntas usando las opciones de las Figuras 5.25 y 5.26

### 5.3.2.3. Pregunta para desarrollar

El ejercicio pregunta para desarrollar tiene como finalidad permitir que se inserte en el editor de texto un enunciado que pregunte al alumno sobre

teoría, de forma que tenga que argumentar su respuesta de manera más extensa.

La interfaz de este ejercicio se puede ver en la Figura 5.28. Para crear el ejercicio, hace falta escribir el enunciado y elegir el número de líneas máximo para desarrollarlo. Así mismo, se ha incluido una opción no obligatoria para añadir espacio extra entre líneas en caso de que algún estudiante tenga un tamaño de letra mayor.

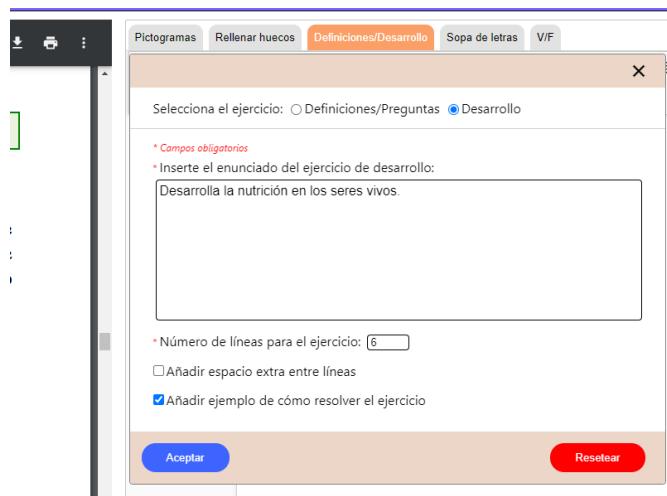


Figura 5.28: Interfaz del ejercicio pregunta para desarrollar

Una vez que se han llenado los campos obligatorios, el botón Aceptar se habilitará. Al hacer clic en él se procederá a escribir, en el editor, el enunciado del ejercicio y en los siguientes renglones el número de líneas especificado por el usuario. También se reseteará los datos del ejercicio y se cerrará el modal. En la Figura 5.29 se muestra el resultado de este ejercicio. En la parte superior se muestra dicho resultado usando las opciones de la Figura 5.28. En la parte inferior, el mismo ejercicio pero quitando la opción de añadir el ejemplo de cómo resolver el ejercicio, y añadiendo espacio extra entre líneas.

#### 5.3.2.4. Verdadero o Falso

La opción Verdadero o Falso sirve para plantear ejercicios en los que el alumnado tenga que indicar si un predicado es verdadero o falso, escribiendo en el recuadro una V si es verdadero o una F si es falso.

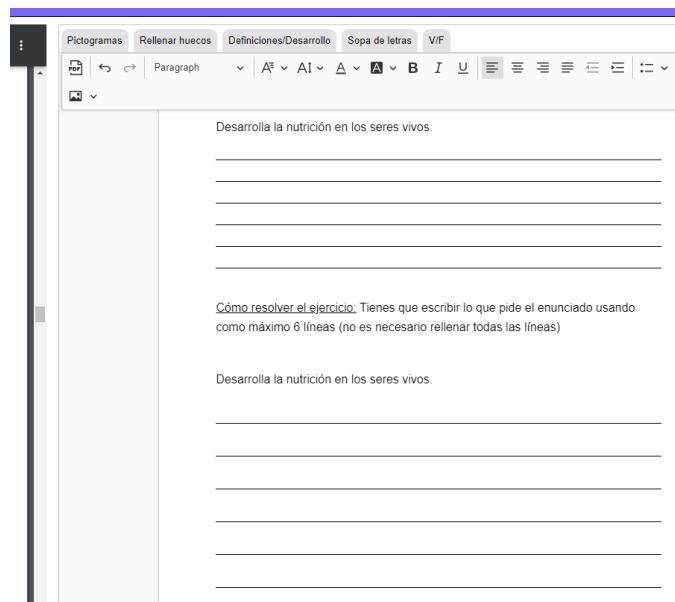


Figura 5.29: Resultado de ejercicio pregunta para desarrollar

Al seleccionar esta opción en nuestra aplicación, se muestra la interfaz de la Figura 5.30. Para crear un ejercicio de verdadero y falso se necesitan llenar los siguientes campos:

- **Frases:** es un array de strings que inicialmente está vacío, y contendrá las frases del ejercicio. Hay que escribir una frase en cada input de texto (para añadir más inputs, se puede pulsar la tecla enter dentro de uno de los inputs, o pulsar el botón “más” al lado de éste. Para quitarlos, se tendrá que hacer clic en el “menos”).
- **La forma para mostrarla:** puede ser como una lista no ordenada o una lista ordenada. Por defecto está seleccionada la lista no ordenada.

Una vez que se han llenado los campos obligatorios mencionados anteriormente, el botón **Aceptar** se habilitará. Al hacer clic en él se procederá a escribir en el editor cada frase, seguido de un recuadro para escribir V si es verdadero o F si es falso, y se procederá a resetear los datos del modal y a cerrarlo. En la Figura 5.31 se muestra el resultado del ejercicio usando las opciones de la Figura 5.30.

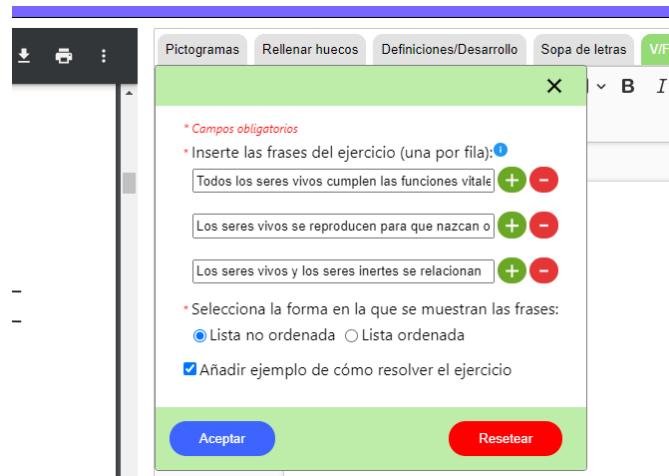


Figura 5.30: Interfaz del ejercicio Verdadero o Falso

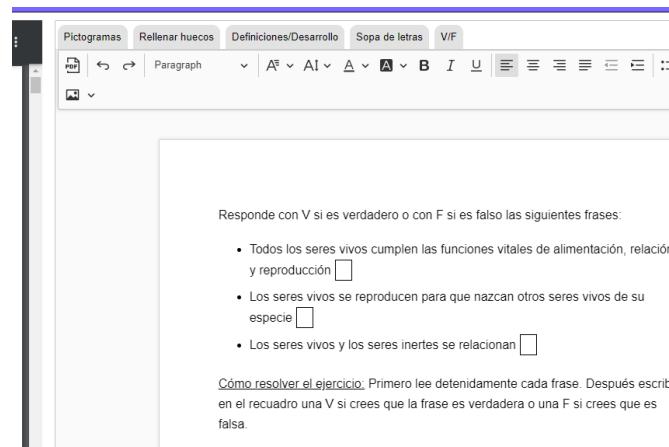


Figura 5.31: Resultado de ejercicio Verdadero o Falso

### 5.3.2.5. Rellenar huecos

El ejercicio de rellenar huecos tiene como finalidad escribir las palabras faltantes en los espacios en los que haya una línea. Al seleccionar esta opción en nuestra aplicación se muestra la interfaz de la Figura 5.32. Este ejercicio consta de dos fases:

- En la primera fase hay que insertar el texto en el que se quiera quitar palabras. Una vez que se ha insertado, el botón Seleccionar palabras se habilitará.



Figura 5.32: Interfaz de edición del ejercicio rellenar huecos

- La segunda fase se alcanza cuando se hace clic en el botón de Seleccionar palabras. Para quitar una palabra basta con hacer clic en ella. Si se quiere volver a ver la palabra, hay que volver a hacer clic en ella. Para volver a editar el texto (volver a la fase uno), simplemente hay que hacer clic en el botón Editar texto. En la Figura 5.33 se puede ver la interfaz de selección de las palabras. En ésta, se han seleccionado las palabras “vivos”, “plantas” y “orgánicas”.

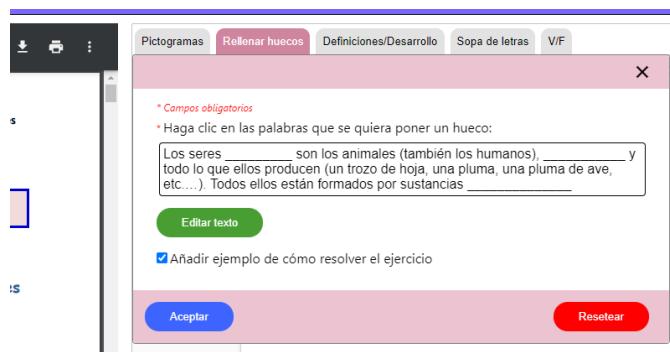


Figura 5.33: Interfaz de selección de palabras del ejercicio rellenar huecos

Una vez que se ha seleccionado al menos una palabra, el botón Aceptar se habilitará. Al hacer clic en éste, se escribirá, en el editor, el texto modificado. En el enunciado del ejercicio se mostrará las palabras que se han escondido. Así mismo, se reseteará el modal y se cerrará. En la Figura 5.34 se ve el resultado del ejercicio al usar las opciones de las Figuras 5.32 y 5.33.

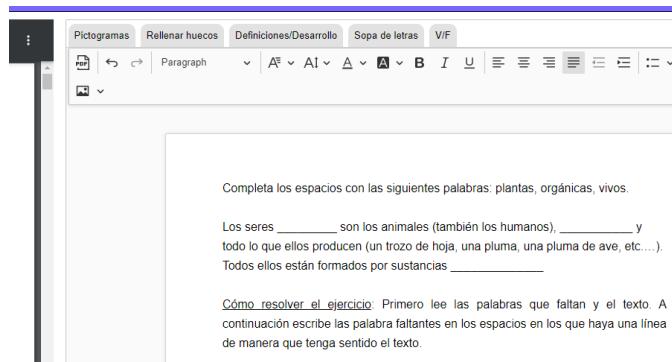


Figura 5.34: Resultado del ejercicio de rellenar huecos

### 5.3.3. Estructura del código

La organización de directorios del proyecto gira en torno al patrón de diseño Redux, como se ha explicado en la sección 5.3.1 de este mismo capítulo.

El código de la funcionalidad de la aplicación se estructura principalmente en cuatro carpetas: components, pages, ckeditor y redux; además, contamos con el fichero App, que supone el punto de inicio del programa.

El contenido de cada carpeta es el siguiente:

- **Components:** Representado en la Figura 5.35, contiene cada componente que representa una funcionalidad para la aplicación, o parte de ella, y todos cuentan, como mínimo, con los siguientes ficheros:

- Fichero modal de la funcionalidad del componente.
- Fichero que realiza las modificaciones pertinentes para ejecutar la funcionalidad.
- Fichero de estilo del componente.

Uno de los componentes más importantes es el de la barra de herramientas (*Toolbar*), que sirve como hilo conductor de la selección de los diferentes tipos de adaptaciones que pueden agregarse al documento, mediante el patrón de diseño del despachador (*dispatcher*).

- **Pages:** Representado en la Figura 5.36, contiene la página del editor, que permite cargar un fichero inicial para comenzar la adaptación, y la

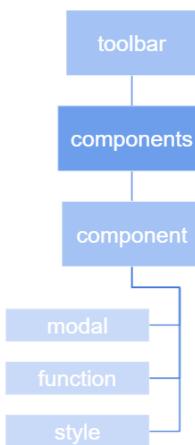


Figura 5.35: Ejemplo de organización de Components

página de ayuda, que muestra información relevante sobre la aplicación y su uso. En los dos casos se cuenta con:

- Fichero de funcionalidad de la página.
- Fichero de estilo de la página.



Figura 5.36: Ejemplo de organización de Pages

- **Redux:** Representado en la Figura 5.37, contiene la implementación del patrón de diseño que se ha aplicado a toda la aplicación, basado en el almacenamiento de los datos que necesita cada componente. La carpeta consta del store y el reducer, elementos necesarios para el desarrollo del mismo. Además, se incluye también cada componente principal con los siguientes ficheros:

- Fichero action: controla las diferentes acciones que puede ejecutar cada componente.
- Fichero reducer: controla el estado de los componentes según el tipo de acción que realicen, partiendo de un estado inicial.
- Fichero selectors: devuelve el valor de los atributos de cada componente.
- Fichero types: contiene un listado de acciones posibles para cada componente.

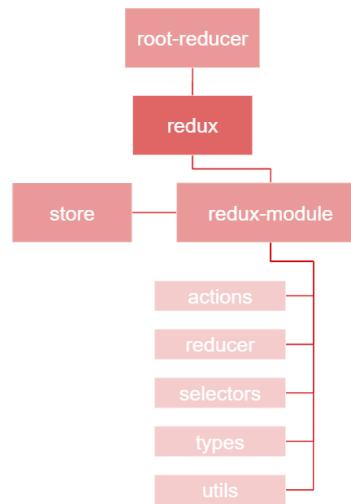


Figura 5.37: Ejemplo de organización de Redux

- **CKEditor:** Representado en la Figura 5.38, contiene todos los componentes software o plugins que se han desarrollado para la aplicación, permitiendo la interacción a partir de comandos que controlan las diferentes funcionalidades. Consta de un fichero que devuelve la instancia del editor CKeditor5 y una carpeta para cada plugin, con los siguientes archivos:

- Fichero de implementación del plugin.
- Fichero de inserción del plugin mediante el patrón de diseño de comando (*command*).

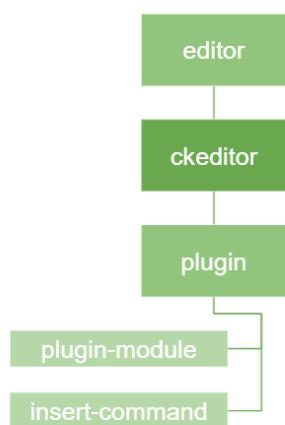


Figura 5.38: Ejemplo de organización de CKEditor



# Capítulo 6

## Evaluación de AdaptaMaterialEscolar

**RESUMEN:** En este capítulo se explicará cómo se realizó la evaluación del proyecto AdaptaMaterialEscolar con los usuarios finales y los resultados obtenidos.

### 6.1. Función de la evaluación

Una vez se tuvo una versión lo suficientemente completa de la aplicación web AdaptaMaterialEscolar para que fuera probada por los usuarios finales, es decir, las profesoras del Aula TEA de Ciudad Real, organizamos la evaluación que necesitábamos para cumplir la intención principal de ésta, la de confirmar si finalmente había resultado útil nuestro proyecto para los docentes que tuvieran que realizar algún tipo de adaptación curricular al alumnado que lo necesitara, durante cualquier etapa académica.

Con el fin de hacer una valoración detallada, los aspectos que queríamos analizar en nuestro proyecto eran: el diseño de la interfaz, la usabilidad del sistema al completo y cada funcionalidad independiente y las posibles mejoras que se valoraría implementar en el futuro.

Para facilitar la realización de la evaluación de nuestra aplicación web y análisis de los resultados, desarrollamos una encuesta en Google Forms para los usuarios finales, que finalmente fueron más de los esperados, tal y como se explicará a lo largo del capítulo.

La estructura de las preguntas del cuestionario que realizamos fue la siguiente:

- Preguntas generales, para obtener algunos datos estadísticos del usuario (Figura 6.1).

The screenshot shows a survey interface with the title "Preguntas generales". Below it, a sub-instruction says "A continuación, algunas preguntas generales:". There are two main sections separated by horizontal lines. The first section asks "¿Cuál es tu género? \*". It has four radio button options: "Mujer", "Hombre", "Prefiero no decírtlo", and "Otro". The second section asks "¿Eres docente? \*". It has two radio button options: "Sí" and "No". Both sections include an asterisk (\*) next to their respective question labels.

Figura 6.1: Preguntas generales

- Preguntas para docentes, para realizar ciertas preguntas únicamente a quienes pertenecieran a la profesión (Figura 6.2).
- Preguntas sobre usabilidad, para recopilar información sobre la facilidad o dificultad de uso del sistema completo (Figuras 6.3, 6.4 y 6.5).
- Preguntas sobre diseño, para obtener información sobre la estética de la aplicación (Figura 6.6).
- Preguntas sobre funcionalidades, para obtener información sobre la dificultad de uso de cada funcionalidad y recopilar nuevas adaptaciones o mejoras que se desean para la aplicación (Figuras 6.7 y 6.8).
- Preguntas sobre utilidad real, para comprobar si realmente la aplicación ha cumplido con su finalidad y evaluar si debería seguir en desarrollo en el futuro (Figura 6.9).

Todos los apartados de la encuesta, a excepción del de usabilidad, fueron planificados por nosotros específicamente para obtener información que

**Preguntas para docentes**

A continuación, unas preguntas para los docentes:

¿Podrías decírnos en qué nivel del sistema educativo eres docente? \*

Educación infantil  
 Educación primaria  
 Educación secundaria  
 Bachillerato  
 Formación profesional de grado medio o superior  
 Otra...

¿Podrías decírnos de qué centro educativo? No es obligatorio que respondas.

Texto de respuesta corta

Figura 6.2: Preguntas para docentes

**Preguntas sobre la usabilidad de la aplicación**

A continuación, algunas preguntas sobre la usabilidad de la aplicación:

Creo que me gustaría utilizar este sistema con frecuencia \*

1      2      3      4      5

Muy en desacuerdo                        Muy de acuerdo

El sistema me parece innecesariamente complejo \*

1      2      3      4      5

Muy en desacuerdo                        Muy de acuerdo

Me pareció que el sistema era fácil de utilizar \*

1      2      3      4      5

Muy en desacuerdo                        Muy de acuerdo

Figura 6.3: Preguntas sobre SUS I

Creo que necesitaría el apoyo de un técnico para ser capaz de utilizar este sistema \*

1      2      3      4      5

Muy en desacuerdo      Muy de acuerdo

Me parece que las distintas funciones de este sistema estaban bien integradas \*

1      2      3      4      5

Muy en desacuerdo      Muy de acuerdo

Me pareció que había demasiada inconsistencia en este sistema \*

1      2      3      4      5

Muy en desacuerdo      Muy de acuerdo

Me imagino que la mayoría de la gente aprendería a utilizar este sistema muy rápidamente \*

1      2      3      4      5

Muy en desacuerdo      Muy de acuerdo

Figura 6.4: Preguntas sobre SUS II

El sistema me pareció muy engorroso de usar \*

1      2      3      4      5

Muy en desacuerdo      Muy de acuerdo

Me sentí muy seguro al utilizar el sistema \*

1      2      3      4      5

Muy en desacuerdo      Muy de acuerdo

Tuve que aprender muchas cosas antes de poder ponerme en marcha con este sistema \*

1      2      3      4      5

Muy en desacuerdo      Muy de acuerdo

Figura 6.5: Preguntas sobre SUS III

**Preguntas sobre el diseño de la aplicación**

A continuación, algunas preguntas sobre el diseño de la aplicación:

¿Te parece que la aplicación es visualmente atractiva? \*

1    2    3    4    5

Muy en desacuerdo                     Muy de acuerdo

¿Te gustaría que se realizaran cambios en la estética de la aplicación? No es obligatorio que respondas

Texto de respuesta larga

Figura 6.6: Preguntas sobre diseño

**Preguntas sobre las funcionalidades**

A continuación, algunas preguntas sobre las diferentes funcionalidades de la aplicación:

Selecciona la dificultad de uso de cada funcionalidad de la aplicación (1 es Muy difícil y 5 Muy fácil): \*

1	2	3	4	5	
Generador de s...	<input type="radio"/>				
Ejercicio de rell...	<input type="radio"/>				
Ejercicio de des...	<input type="radio"/>				
Buscador de pi...	<input type="radio"/>				
Ejercicio de def...	<input type="radio"/>				
Ejercicio de ver...	<input type="radio"/>				

Figura 6.7: Preguntas sobre funcionalidades I

considerábamos relevante para el proyecto. Sin embargo, las preguntas sobre usabilidad de la evaluación siguen el modelo de cuestionario de la Escala de Usabilidad de un Sistema, o SUS (System Usability Scale) por sus siglas en inglés, uno de los más utilizados para la medición de usabilidad en Experiencia de Usuario. Se compone por diez preguntas puntuables del 1 al 5, donde la valoración más baja significa que el usuario está “Muy en desacuerdo” con la afirmación y, en la más alta, “Muy de acuerdo” y, a pesar de que es posible modificar las preguntas del cuestionario SUS, las mantuvimos tal y como aparecen en el modelo original.

¿El tamaño de la letra y el espacio es adecuado? \*

1      2      3      4      5

Muy en desacuerdo                               Muy de acuerdo

---

¿El tamaño de los pictogramas es adecuado? \*

1      2      3      4      5

Muy en desacuerdo                               Muy de acuerdo

---

¿Te gustaría que se añadiera alguna configuración de formato más? Por ejemplo, otros tipos de fuente, tamaño, etc. No es obligatorio que respondas.

Texto de respuesta corta

Figura 6.8: Preguntas sobre funcionalidades II

Para obtener un dato relevante sobre la usabilidad de nuestro proyecto, con el modelo SUS es necesario realizar una medición de la siguiente forma:

- Las preguntas pares deben restar a 5 el valor que asigne el usuario.
- Las preguntas impares deben restar 1 al valor que asigne el usuario.

Si el resultado que se obtiene es cercano a 100, que es la máxima puntuación que se podría obtener con las preguntas del cuestionario, significa que la usabilidad del proyecto es buena para el usuario. En caso contrario, habría que realizar cambios para mejorar y facilitar su uso.

## 6.2. Prueba con el usuario final

El viernes 28 de mayo planificamos una reunión presencial en el IES Maestro Juan de Ávila con las profesoras del Aula TEA, quienes han colaborado en el proyecto, como se ha comentado en anteriores capítulos, para establecer los requisitos necesarios para el desarrollo de AdaptaMaterialEscolar. Una vez allí, la orientadora del centro, María José Morales Rubio, se unió a la presentación por iniciativa e interés propio en nuestro proyecto. Se realizó una prueba de aproximadamente una hora de duración, en la que se les mostró todas las posibilidades de adaptación de un tema de naturales,

**Preguntas sobre utilidad real**

A continuación, algunas preguntas sobre la utilidad real que consideras que tiene la aplicación:

×

⋮

¿Te parece que la aplicación es útil? \*

1    2    3    4    5

Muy en desacuerdo                     Muy de acuerdo

¿Crees que ayuda a reducir el tiempo de trabajo del profesorado? \*

1    2    3    4    5

Muy en desacuerdo                     Muy de acuerdo

¿Te gustaría que la aplicación siguiera en desarrollo para mejorarla? \*

Sí  
 No  
 No sé

¿Qué mejorarias en la aplicación? No es obligatorio que respondas, pero valoraríamos mucha tu opinión.

Texto de respuesta larga

Figura 6.9: Preguntas sobre utilidad real

que nos facilitaron las propias profesoras del Aula TEA, con cada una de las funcionalidades que estaban desarrolladas para el momento de la prueba.

Ese mismo día se realizó una prueba de la aplicación con otra profesora del mismo instituto, Esther Valiente Moreno, que imparte las asignaturas de Emprendimiento Empresarial y Economía en el centro y cuenta con un alumno TEA en una de las clases. Además, aunque por falta de tiempo fue imposible mostrar en el día el proyecto a más profesores, contactaron con nosotros tres docentes más, debido a que se habían interesado en aprender a utilizar AdaptaMaterialEscolar para el curso siguiente, por los buenos comentarios que les llegaron por parte de quienes hicieron la prueba con nosotros. Uno de ellos imparte Biología y los otros dos, Historia y Geografía.

El día 1 de junio, una profesora del IES Pedro Álvarez de Sotomayor de Manzanares, María de las Cruces Valiente Moreno, que imparte la asignatura de Tecnología Industrial, también se interesó por el proyecto, debido

a que nos comentó que en su centro había un alumno con sordera total a quien debían adaptar el material, por lo que se organizó una reunión por videollamada para mostrarle el funcionamiento de AdaptaMaterialEscolar.

Durante las reuniones con los profesores de los distintos centros educativos, surgieron requisitos nuevos que se pretenden plantear como trabajo futuro en caso de que AdaptaMaterialEscolar pueda seguir en desarrollo en los próximos años; incluso se nos llegó a ofrecer la oportunidad de realizar un curso de formación a nuevos profesores y alumnos de máster el curso que viene, en el IES Maestro Juan de Ávila, para presentarlo como herramienta de adaptabilidad para el profesorado, independientemente de si llegábamos a cumplir con los nuevos requisitos o si manteníamos el proyecto tal y como lo vieron.

Los requisitos que se recopilaron entre todas las reuniones de forma oral, además de reiterar la necesidad de continuar desarrollando los de la captación inicial, abrieron nuevas posibilidades para asignaturas que no se tuvieron en cuenta al principio. Los requisitos fueron los siguientes:

- Añadir un pictotraductor como funcionalidad.
- Añadir cuadrícula para ejercicios de matemáticas, para que los alumnos puedan escribir los números en ella.
- Añadir la alternativa de añadir doble pauta, en vez de renglones de una única línea, para determinar el tamaño de la letra del alumno.
- Crear una herramienta de recorte de imágenes para el texto original.
- Añadir encabezado al texto con, al menos, nombre del centro educativo, nombre del alumno y asignatura.
- Añadir ejercicios con espacio para dibujar.
- Añadir ejercicios para ejercicios de cálculo con fórmulas con huecos a llenar por el alumno.
- Enumerar ejercicios de forma automática.

Como todos los profesores realizaron la encuesta mencionada anteriormente, anotaron también otros requisitos que no se comentaron de forma oral durante las pruebas de la aplicación.

# Capítulo 7

## Trabajo Individual

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

### 7.1. Natalia

Una de las tareas que he realizado durante todo el proyecto ha sido la de actuar como punto de comunicación entre nuestro equipo y las profesoras del Aula TEA del IES Maestro Juan de Ávila de Ciudad Real.

Contacté con ellas para realizar una entrevista presencial, donde se les pidió colaboración en el proyecto, de forma que pudieran aportar su punto de vista profesional y experiencia trabajando con alumnos con necesidades especiales.

A partir de la reunión, correspondiente a la primera iteración, realicé la captación de requisitos inicial del proyecto en base a las necesidades que para las profesoras del Aula TEA aún no estaban cubiertas por ninguna herramienta de trabajo que pudiera facilitar la adaptación de material escolar, aunque tampoco para ningún otro docente de ese centro.

Los requisitos finales obtenidos se dividían en listados de tareas, diferenciados en adaptaciones de formato, actividades y/o exámenes y temario.

Evalué los requisitos que se habían obtenido como resultado de la captación de requisitos de forma individual, según los conocimientos adquiridos

durante el grado, y calculé la prioridad final de cada tarea con la puntuación que habían asignado las profesoras y todos los componentes del equipo de desarrollo.

Cuando comenzamos a implementar el proyecto tuvimos que reajustar algunas puntuaciones y obtuve una nueva lista de funcionalidades prioritarias, distinta a la inicial, de manera que reorganizamos las tareas en el equipo.

Realicé en común con mis compañeros de equipo en el proyecto un diseño para la interfaz gráfica de la aplicación web para conseguir un prototipo inicial, de forma que vimos las diferentes herramientas que podíamos añadir al trabajo para aportar valor, como por ejemplo, un editor de código abierto de tipo WYSIWYG (What You See Is What You Get).

También realicé un diseño distinto al prototipo inicial del proyecto de forma individual, para que no hubiera influencia entre el equipo y se pudiera debatir luego sobre los cambios que cada uno consideraba importantes, de manera que fuera posible mejorar el diseño inicial añadiendo los cambios que a todos nos habían parecido necesarios y relevantes.

Investigué sobre cómo subir un fichero de texto que estuviera almacenado en una ubicación local del ordenador y sobre el editor de texto CKEditor 5, de tipo WYSIWYG, para poder añadirlo al prototipo individual y comprobar la dificultad inicial de implementar los requisitos necesarios para el proyecto.

Una vez que se consiguió añadir el editor de texto y escogimos la interfaz definitiva, comencé a desarrollar la adaptación de formato del generador de resúmenes y tres adaptaciones de actividades, ejercicio de verdadero o falso, de definiciones y de desarrollo.

La primera funcionalidad en la que trabajé fue el generador de resúmenes, con el que tuve que realizar la búsqueda de algún servicio web de código abierto que fuera capaz de sintetizar un texto, y llegué a añadir al proyecto, aunque no al editor, una primera versión de la funcionalidad, aunque al cambiar la lista de prioridades ésta tarea quedó en una posición intermedia y tuve que cambiar a otra.

Las siguientes tres funcionalidades que implementé sí llegaron a completarse para la entrega del proyecto y se trataban de adaptaciones de ejercicios. La primera fue el Ejercicio de verdadero o falso, la segunda, el Ejercicio de Definiciones y, la tercera, el Ejercicio de Desarrollo.

Una vez que tuvimos una versión suficientemente desarrollada de AdaptaMaterialEscolar, redacté las preguntas de la evaluación que haríamos a los

usuarios que probaran el sistema, para poder obtener información real sobre la utilidad de la aplicación y las posibles mejoras que podrían hacerse en un futuro cercano.

Contacté de nuevo con las profesoras del Aula TEA del IES Maestro Juan de Ávila de Ciudad Real para realizar una reunión presencial en la que pudiera mostrarles el avance del proyecto, cómo se utilizaba la aplicación y darles la opción de que evaluaran AdaptaMaterialEscolar. Además, una tercera profesora quiso probar la aplicación y también realizó la evaluación.

Analicé los resultados obtenidos después de la reunión presencial con las profesoras y realicé más demostraciones de AdaptaMaterialEscolar a otros profesores que se interesaron.

## 7.2. Jorge

Mi trabajo comenzó una vez que Natalia consiguió recabar los requisitos junto con las profesoras del Aula TEA. Lo primero de todo fue evaluar los requisitos por ambas partes; las profesoras debían puntuarlos según la importancia que tuviera ese requisito en una escala del uno al tres, siendo el uno poco importante, y el tres muy importante. Nosotros lo tuvimos que puntuar por la dificultad de implementación, siendo el uno una dificultad difícil, y el tres, fácil, de forma individual y sin influir los unos con los otros. Una vez realizado esto, se hizo la media de dificultad entre los tres integrantes.

A continuación, creamos en conjunto un prototipo inicial a través de la herramienta online *Moqups*, donde surgieron varias vistas de la aplicación, descritas en la sección 5.2.2. Así mismo, diseñé un prototipo individual, al igual que mis compañeros, para poder mejorar ese diseño en conjunto, el cual se puede ver en las figuras 5.7, 5.10, 5.11 y 5.12. Esta creación corresponde a una iteración competitiva, la cual se puede leer en la sección 5.2.3.

El 4 de diciembre de 2019 el grupo de investigación NIL organizó un *workshop* en la Facultad con docentes de otros centros y asociaciones, en el que se mostraban herramientas tecnológicas inclusivas que se habían desarrollado o estaban en desarrollo. Presenté, en ese *workshop*, AdaptaMaterialEscolar, donde expliqué el funcionamiento y finalidad de este proyecto, con ejemplos visuales sobre posibles adaptaciones que podrían llegar a realizarse.

Me encargué de gestionar y reorganizar el tablero *Kanban*, y de que estuviera lo más actualizado posible tras las reuniones con las tutoras.

Así mismo, y como la esencia de este proyecto es realizar adaptaciones, investigamos distintos editores de código abierto de tipo “WYSIWYG”<sup>1</sup>, aunque al final uno de mis compañeros localizó uno que cumplía las características que buscábamos, *CKEditor*. Una vez implementado este editor por parte de mi compañero Pablo, cada uno de nosotros se encargó de comprobar algún requisito de los datos. En mi caso investigué si había algún paquete NPM (*Node Package Manager*) sobre generación de sopas de letras, y ver si con nuestra tecnología era posible implementarlo.

Dediqué bastante tiempo a implementar la sopa de letras debido a que tenía mucha configuración por detrás en comparación con otros ejercicios de este proyecto. Así mismo, esta sopa de letras daba un único error al colocar un número de filas y columnas iguales o menores que cero, por lo que analicé los posibles errores que podría dar, y añadí uno extra para avisar a los usuarios de si todas las palabras estaban en ésta. Posteriormente rediseñé la mayoría de la interfaz de usuario de nuestra aplicación, en concreto, la barra de herramientas de los ejercicios (*toolbar*) y las ventanas modales de los éstos. Una vez hecho el rediseño, me puse a implementar el ejercicio de Rellenar huecos. En principio se planteó rodeando la palabra con algún carácter especial, por lo que busqué alguna API y librería que pudiera sustituir la palabra rodeada por algún carácter. Como no encontré nada que valiera la pena, le di otro enfoque y lo propuse haciendo clic en las palabras.

También investigué la API de Google Docs para ver si se podía implementar en el proyecto y así poder subir ficheros en formatos que no sean solo *.pdf* de forma sencilla.

### 7.3. Pablo

Tras la propuesta del tema y las entrevistas con las profesoras, mis compañeros me ofrecieron incorporarme al equipo.

Aporté mi puntuación de 1 a 3 de dificultad a los requisitos propuestos por las profesoras, siendo uno fácil y tres difícil, y se actualizaron los valores finales. Los resultados fueron similares a los previos.

Una vez decidido que el proyecto iba a ser web, había que decidir qué

---

<sup>1</sup> **WYSIWYG**, acrónimo de *What You See Is What You Get* (en español, "lo que ves es lo que obtienes"), es una frase aplicada a los procesadores de texto y otros editores con formato, que permiten escribir un documento mostrando directamente el resultado final, frecuentemente el resultado impreso. <https://es.wikipedia.org/wiki/WYSIWYG>

tecnologías FrontEnd usar. Por ello propuse usar React en el proyecto dada su comodidad, sencillez y adecuación para el proyecto. Además, mi previa experiencia usándolo serviría de ayuda para mis compañeros. La propuesta fue aceptada e implementada.

Investigué sobre librerías aptas para poder integrar un editor WYSIWYG (What You See Is What You Get) en la web. Finalmente propuse CKEditor por su gran cantidad de funciones, soporte y personalización, y fue aceptado por el equipo. Exploré la documentación de CKEditor en profundidad, procurando encontrar qué funcionalidades del framework eran útiles y necesarias para nuestro proyecto.

Posteriormente, realicé un prototipo web usando las tecnologías y librerías web propuestas. Este prototipo estaba realizado con React y CKEditor y mostraba un editor de texto que contenía un buscador de pictogramas integrado. De esta manera podía comprobar la viabilidad del proyecto con las tecnologías propuestas y así para servir como referencia en la estructura y avance del proyecto. Además, facilitaría el aprendizaje de su uso a mis compañeros.

Por último, inicialicé un repositorio en GitHub para poder almacenar nuestra plantilla Tesis de la memoria, gestionarlo de manera más ordenada y no recurrir a otras herramientas de edición de texto.



# Bibliografía

¿Qué es la educación inclusiva? <http://www.inclusioneducativa.org/ise.php?id=1>, 2006.

Información básica sobre el trastorno del espectro autista. <https://www.cdc.gov/ncbdd/spanish/autism/facts.html>, 2014.

BELLOCH, C. Sistemas de signos. <https://www.uv.es/bellochc/logopedia/NRTLogo8.wiki?7>, 2014.

BOE. Decreto 98/2005, de 18 de agosto, de ordenación de la atención a la diversidad en las enseñanzas escolares y la educación preescolar en Cantabria. 2005.

CADAH, F. Tipos de adaptaciones curriculares individualizadas (a.c.i.). <https://www.fundacioncadah.org/web/articulo/tipos-de-adaptaciones-curriculares-individualizadas.html>, 2012.

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, Disponible en <http://ojs.uac.edu.co/index.php/prospectiva/article/view/36>.

ÓSCAR GARCÍA MUÑOZ. Lectura fácil: Métodos de redacción y evaluación. vol. 1, Disponible en <https://www.plenainclusion.org/sites/default/files/lectura-facil-metodos.pdf>.

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

GONZÁLEZ, L. ¿Cómo se organiza el sistema educativo español? <https://www.emagister.com/blog/se-organiza-sistema-educativo-espanol/>, 2020.

ORJALES VILLA, I. *Déficit de atención con hiperactividad. Manual para padres y educadores*. Editorial CEPE, Madrid, 1999.

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.

RUIZ-GRANADOS, A. S., BALLESTEROS, R. H., MANJÓN, B. F. y LORENZ, A. V. Diseño de sistemas interactivos. 2019.