
AdaptaMaterialEscolar



Trabajo de Fin de Grado

Pablo Miranda Torres
Natalia Rodríguez Peral-Valiente
Jorge Velasco Conde

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

Junio 2020

Documento maquetado con T_EX^S v.1.0+.

Este documento está preparado para ser imprimido a doble cara.

AdaptaMaterialEscolar

Informe técnico del departamento

Ingeniería del Software e Inteligencia Artificial
IT/2009/3

Versión 1.0+

Departamento de Ingeniería del Software e Inteligencia
Artificial

Facultad de Informática
Universidad Complutense de Madrid

Junio 2020

Copyright © Pablo Miranda Torres
Natalia Rodríguez Peral-Valiente
Jorge Velasco Conde

ISBN 978-84-692-7109-4

Índice

1. Introducción	1
1.1. Motivación	1
1.2. Objetivos	2
2. Estado del Arte	3
2.1. Adaptación Curricular	3
2.1.1. Lectura fácil	4
2.1.2. Pictogramas	5
2.1.3. Herramientas que facilitan adaptaciones curriculares .	7
2.2. Diseño Centrado en el Usuario	7
2.2.1. Conocer al usuario	8
2.2.2. Prototipado	8
2.2.3. Evaluación	10
3. Metodología de desarrollo	11
3.1. Metodología tradicional vs. Metodología ágil	11
3.2. Kanban	12
3.2.1. Tablero Kanban	13
3.3. Tipos de pruebas	15
3.3.1. Pruebas unitarias	15
4. Herramientas empleadas	19
4.1. Moqups	19
4.2. React	20
Bibliografía	23

Índice de figuras

2.1. Ejemplo de pictograma de una rosa.	5
2.2. Ejemplo de pictograma que representa la acción comer.	6
2.3. Ejemplo del buscador de pictogramas de ARASAAC y las opciones que ofrece	6
2.4. Interfaz de ARAWORD.	7
2.5. Ejemplo de guión gráfico.	9
2.6. Ejemplo de prototipo en papel interactivo.	10
3.1. Fases de una metodología tradicional	12
3.2. Tablero <i>Kanban</i> al inicio del proyecto	14
3.3. Tarea asignada a varios usuarios	16
4.1. Interfaz de Moqups	20
4.2. Menú lateral de interacciones	21
4.3. Funcionalidad sin usar JSX	22
4.4. Funcionalidad empleando JSX	22

Índice de Tablas

Capítulo 1

Introducción

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

1.1. Motivación

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 (E.P.) y Educación Secundaria Obligatoria (E.S.O.).

"La inclusión se ve como el proceso de identificar y responder a la diversidad de las necesidades de todos los estudiantes a través de la mayor participación en el aprendizaje, las culturas y las comunidades, y reduciendo la exclusión en la educación. Involucra cambios y modificaciones en contenidos, aproximaciones, estructuras y estrategias, con una visión común que incluye a todos los niño/as del rango de edad apropiado y la convicción de que es la responsabilidad del sistema regular, educar a todos los niño/as."

En este ámbito nos encontramos alumnos con Trastorno del Espectro Autista, que pueden necesitar adaptaciones curriculares individualizadas, significativas (o muy significativas) o no significativas.

"Los trastornos del espectro autista (TEA) son una discapacidad del desarrollo que puede provocar problemas sociales, comunicacionales y conductuales significativos.[...] Es posible que quienes tienen un TEA se comuniquen, interactúen, se comporten y aprendan de maneras distintas a otras personas. Las destrezas de aprendizaje, pensamiento y resolución de problemas de las personas con TEA pueden variar; hay desde personas con muy altos niveles de capacidad y personas que tienen muchas dificultades."

En un centro educativo, los alumnos TEA, además de asistir al aula base con el resto del alumnado, cuentan con aulas especializadas, llamadas Aulas TEA, que les ofrece un espacio propio, donde se favorece su aprendizaje, desarrollo e integración social, ajustándose a su manera de relacionarse con el mundo. En ellas se trabaja y refuerza el contenido escolar del currículo y se fomentan aspectos como la organización y autonomía.

Debido a que cada alumno supone un caso único por sus características particulares, la motivación que ha llevado a la creación de AdaptaMaterial-Escolar es la de desarrollar una herramienta de trabajo inclusiva dedicada al profesorado que imparte clase a estos alumnos, para facilitarles la creación de temario, actividades e instrumentos de evaluación personalizados y acordes al nivel educativo de cada alumno.

1.2. Objetivos

El objetivo de nuestro proyecto es desarrollar una aplicación web que consista en un editor de texto para el profesorado y que permita adaptar un documento original en cualquier formato de manera intuitiva, fácil y rápida, y por tanto se puedan crear unidades didácticas personalizadas que se ajusten a cada alumno.

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

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

Dicha aplicación incluirá herramientas como generador de resúmenes, generador de ejercicios y buscador de pictogramas. Para el generador de resúmenes y buscador de pictogramas utilizaremos unas APIs ya existentes.

También aprovecharemos los conocimientos adquiridos durante la carrera sobre proyectos. Dado que es un proyecto grande, tenemos que garantizar la modularidad y calidad del código, aplicaremos patrones software que nos ayuden a ello. Además, respecto a la gestión del proyecto usaremos la metodología ágil de Kanban.

Capítulo 2

Estado del Arte

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

2.1. Adaptación Curricular

En la educación es necesario conseguir la participación y aprendizaje de todos los alumnos. Para ello, hay que garantizar que tanto los alumnos como la propia escuela sepan adaptarse a las necesidades de todos. Por ello surgen las adaptaciones curriculares.

Se define adaptación curricular (o adecuación curricular) como la medida de modificación de los elementos del currículo a fin de dar respuesta a las necesidades del alumnado. Las adecuaciones curriculares pueden ser necesarias en el caso de que un niño o niña tenga dificultades para adquirir habilidades o conocimientos a la velocidad que se demanda en una escuela mediante la currícula oficial. Las adaptaciones curriculares pueden clasificarse según dos tipos:

- **Adaptaciones Curriculares de Acceso al Currículo.** Responden a las necesidades de un grupo de personas. Estos pueden ser:
 - **De Acceso Físico.** Representa a los recursos materiales y espaciales. Ejemplos de estos son mobiliario adaptado, iluminación y sonoridad adaptada o profesorado especializado.
 - **De Acceso a la Comunicación.** Incluye materiales específicos de enseñanza tales como: aprendizaje, ayudas tecnológicas o sistemas de computación complementarios.

- **Adaptaciones Curriculares Individualizadas.** Se realizan para un único alumno con el fin de responder a necesidades educativas especiales y que no pueden ser compartidas por el resto de los compañeros. Pueden ser:
 - **No Significativas.** Modifican elementos básicos del currículo. Son adaptaciones en cuanto a los tiempos, las actividades, la metodología, las técnicas e instrumentos de evaluación. En un momento determinado, cualquier alumno tenga o no necesidades educativas especiales puede precisarlas. Es la estrategia fundamental 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 objetivos generales de la etapa, contenidos básicos y nucleares de las diferentes áreas curriculares y criterios de evaluación.

2.1.1. Lectura fácil

Uno de los problemas que plantea resolver nuestro proyecto es el de mostrar una lectura más sencilla y visual para las personas con necesidades especiales. Las adaptaciones curriculares para estos casos requieren documentos con una estructura más simple y esquemática. Algunos ejemplos de las adaptaciones más comunes son las siguientes:

- El formato del texto. El texto suele mostrarse con un espaciado entre líneas bastante mayor, para que diferenciar las líneas resulte más sencillo. Además, la fuente del texto suele ser de tipo informal y colorido. El ejemplo más común es Comic Sans.
- Textos más cortos y resumidos. La excesiva longitud del texto puede producir agobio y desmotivación. Por ello, es necesario ofrecer la información más básica.
- Área de escritura definida. El espacio destinado al alumno para escribir tiene que estar claramente señalado, de forma similar a un cuaderno de rayas.
- Ejercicios de resolución sencillos. Incluir un gran abanico de ejercicios que no impliquen mucho esfuerzo de escritura para el alumno, como por ejemplo sopas de letras, crucigramas, ejercicios de verdadero/falso?
- Sistemas Aumentativos y Alternativos de Comunicación (SAAC). Formas de expresión diferentes del lenguaje hablado cuyo objetivo es aumentar y/o compensar (alternativo) las dificultades de comunicación

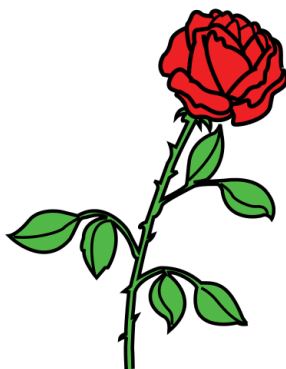


Figura 2.1: Ejemplo de pictograma de una rosa.

y el lenguaje de personas con dificultades. Uno de los SAACs más comunes son los pictogramas, que son dibujos que representan objetos o acciones. Los pictogramas se explicarán más detalladamente en el siguiente apartado.

2.1.2. Pictogramas

Las personas con Trastorno del Espectro Autista presentan dificultades en el desarrollo del lenguaje y en la intención comunicativa. Sin embargo, entienden y procesan muy bien las imágenes. Por ello, los Sistemas Aumentativos y Alternativos de Comunicación (SAACs) 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. Sintetiza un mensaje que puede señalar o informar sobrepasando la barrera de las lenguas.

Es un recurso comunicativo de carácter visual que podemos encontrar en diversos contextos de nuestra vida diaria y nos aporta información útil por todos conocida. Sus principales características 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.

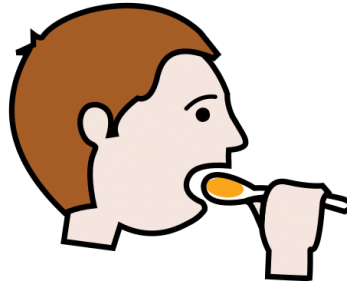


Figura 2.2: Ejemplo de pictograma que representa la acción comer.

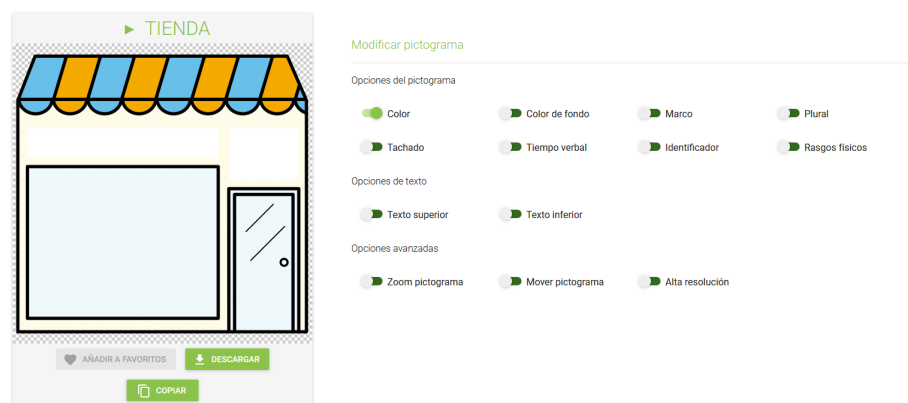


Figura 2.3: Ejemplo del buscador de pictogramas de ARASAAC y las opciones que ofrece

2.1.2.1. ARASAAC

El Portal Aragonés de Comunicación Aumentativa y Alternativa (ARASAAC), creado en 2007, ofrece un amplio catálogo de pictogramas de libre acceso. Su principal herramienta es un buscador de pictogramas que ofrece varios resultados para una palabra introducida. Además, incluye opciones que permiten elegir el estilo del pictograma al gusto del usuario, como por ejemplo elegir si se desea en color o blanco y negro, incluir el título del pictograma, tacharlo, incluir señales que identifiquen símbolo plural, un identificador, etc. En la figura 2.3 se muestra un ejemplo del buscador con todas las opciones de personalización del pictograma.

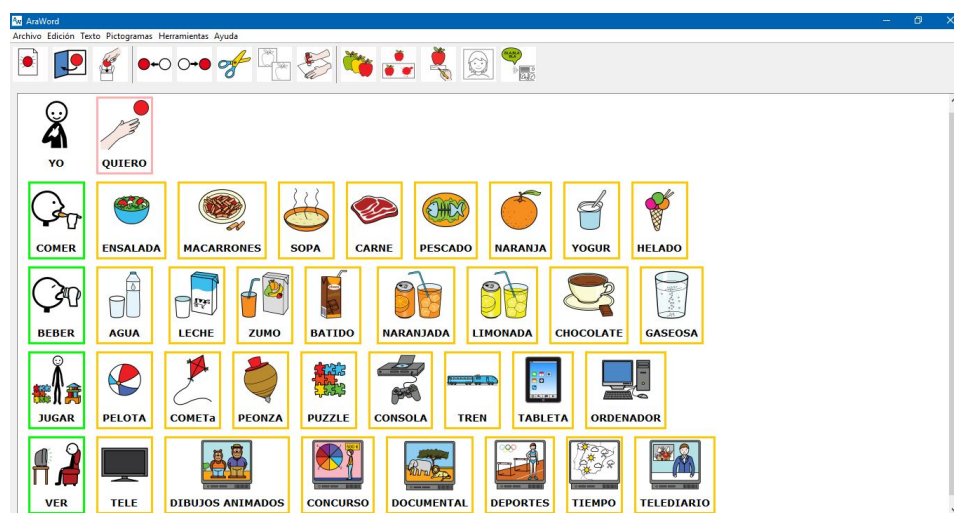


Figura 2.4: Interfaz de ARAWORD.

2.1.3. Herramientas que facilitan adaptaciones curriculares

A continuación se mostrarán algunas herramientas útiles que permiten realizar adaptaciones curriculares especializadas para alumnos con TEA, especialmente centrada en los pictogramas.

2.1.3.1. ARAWORD

Procesador de textos creado por ARASAAC que permite la escritura simultánea de textos y pictogramas. Facilita la elaboración de materiales de comunicación aumentativa, la elaboración de documentos accesibles, y la adaptación de documentos para personas que presentan dificultades en estos ámbitos.

ARAWORD resulta también una herramienta muy útil para ser usada por usuarios que están adquiriendo el proceso de la lectoescritura, ya que la aparición del pictograma, a la vez que se escribe, es un refuerzo muy positivo para reconocer y evaluar que la palabra o la frase escrita es correcta.

2.2. Diseño Centrado en el Usuario

Para que nuestra aplicación responda a las necesidades de nuestros usuarios finales, realizaremos un diseño centrado en el usuario.

El Diseño Centrado en el Usuario (DSU) consiste en una forma de diseño cuyo objetivo es crear un producto que resuelva las necesidades de los clientes, consiguiendo así una mayor calidad de producto y por tanto una

mayor satisfacción de los clientes. Con dicha metodología se pretende entender previamente a los usuarios para poder encontrar soluciones acordes a sus necesidades. Este proceso se basa en tres pilares: conocer al usuario, prototipado del producto y evaluación.

2.2.1. Conocer al usuario

Entender a la gente, el entorno y el uso que puedan darle al producto ayuda a diseñar productos que puedan encajar de mejor forma en dicho contexto y mejorar la forma en la que trabajan, comunican e interactúan.

Por ello, se realizan estudios en los que se planea obtener información para conocer quiénes van a ser los usuarios reales del sistema, cómo se comportan y cuál es el contexto en el que se desenvuelven.

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

- **Entrevistas.** Se realizan preguntas al sujeto de estudio. Responden a un conjunto de preguntas con el fin de aportar la información necesaria para poder comenzar con el diseño. Se suelen realizar diferentes tipos de preguntas, como las orientadas a objetivos, que pretenden descubrir las prioridades del usuario, orientadas al sistema, que identifican las funcionalidades más usadas, orientadas a flujos de trabajo, que identifican procesos y recurrencia de actividades, y las orientadas a actitudes.
- **Observación del usuario.** Analizar el comportamiento del usuario para poder detectar si las necesidades reales coinciden con las necesidades escritas.
- **Diarios de uso.** Dar a los usuarios un diario en el que escriban cosas que servirán para extraer información de ellos. Normalmente en estos diarios se suelen describir interacciones con un sistema, cuándo es necesario el producto o un registro de comportamientos que engloban acciones de los usuarios.
- **Analizar a la competencia.** Estudiar cómo los usuarios usan un sistema ya existente para poder descubrir qué características del producto son útiles y cuáles son los problemas que los usuarios encuentran en estos.

Sea cual sea la técnica utilizada, esta deberá ir seguida de una planificación basada en el tiempo que se dedicará (tanto usuarios como desarrolladores) a la tarea, cuál es el objetivo deseado y quiénes lo van a realizar.

2.2.2. Prototipado

Es importante tener clara cuál va a ser la interacción del usuario con el producto. Por ello, se hacen prototipos que simulan el diseño y comporta-



Figura 2.5: Ejemplo de guión gráfico.

miento del sistema para que los clientes finales puedan dar su feedback y confirmen si es lo que realmente buscan.

Dichos prototipos pueden ser realizados en papel o digital. Los prototipos en papel pueden ser:

- **Guiones gráficos.** conjunto de ilustraciones pintadas a mano a forma de comic, en el que se muestra un escenario y una secuencia de acciones que simulan casos de uso de la aplicación. En la figura 2.5 se muestra un 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.
- **Prototipos en papel interactivos.** Es una maqueta o mockup de la interfaz hecha en papel y compuesta de las piezas que representan a la aplicación. Estos prototipos son interactivos, es decir, muestran una simulación del comportamiento del sistema ante acciones del usuario. Algunas formas de representar estas interacciones son mostrar cambios en la interfaz a raíz de acciones realizadas (como clicks o pulsaciones de botones del teclado).

Por otra parte, los prototipos pueden ser digitales. Un prototipado a computador suele ser mucho más fiel al diseño final del sistema. Además, permiten

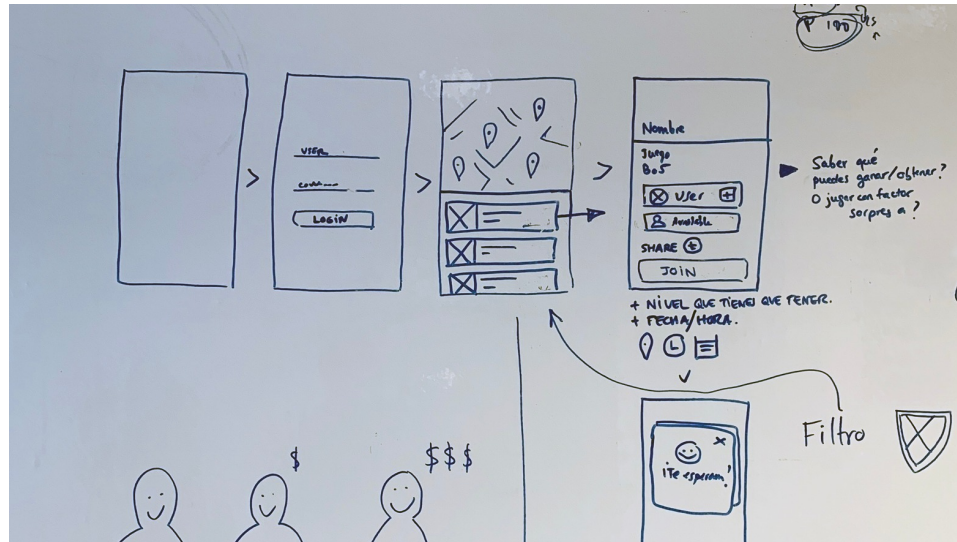


Figura 2.6: Ejemplo de prototipo en papel interactivo.

diseñar interacciones de manera más real. Algunas herramientas para realizar prototipos digitales son Photoshop, Moqups, o incluso HTML y CSS.

2.2.3. Evaluación

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

- **Evaluación heurística.** Identifican 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.** Simulan 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.

Capítulo 3

Metodología de desarrollo

RESUMEN:

En este capítulo se describe la metodología de desarrollo que se usa en el proyecto. En la sección 3.1 se hace una comparativa entre las metodologías tradicionales y las ágiles. En la sección 3.2 se explica la metodología que se ha elegido. En el apartado 3.2.1 se especifica cómo se organizan las tareas a través del tablero *Kanban*. Por último, en la sección 3.3 se describen los tipos de pruebas que se realizarán.

En este proyecto, dado que no se conoce el alcance de la implementación de las distintas partes (componentes) de la aplicación, es decir, no se conoce con certeza si algún requisito es viable, se va a seguir una metodología de desarrollo ágil. En el siguiente apartado se explica el por qué se usa una metodología ágil en vez de una tradicional.

3.1. Metodología tradicional vs. Metodología ágil

En las metodologías tradicionales, se sigue un proceso secuencial unidireccional, es decir, una vez que se alcanza una fase no se puede volver a hacer una fase anterior, por lo que cuando se han capturado los requisitos no se pueden hacer cambios (no se añaden o eliminan requisitos). Cabe señalar que donde más comunicación hay con el cliente es en la primera etapa de captura de requisitos, en las demás fases apenas hay. En la Figura 3.1, se puede ver un ejemplo de las distintas fases más comunes en una metodología tradicional; cuando se alcanza la fase $N+1$, no puedes volver a la fase N .

En cambio, en las metodologías ágiles hay una lista de tareas, que se

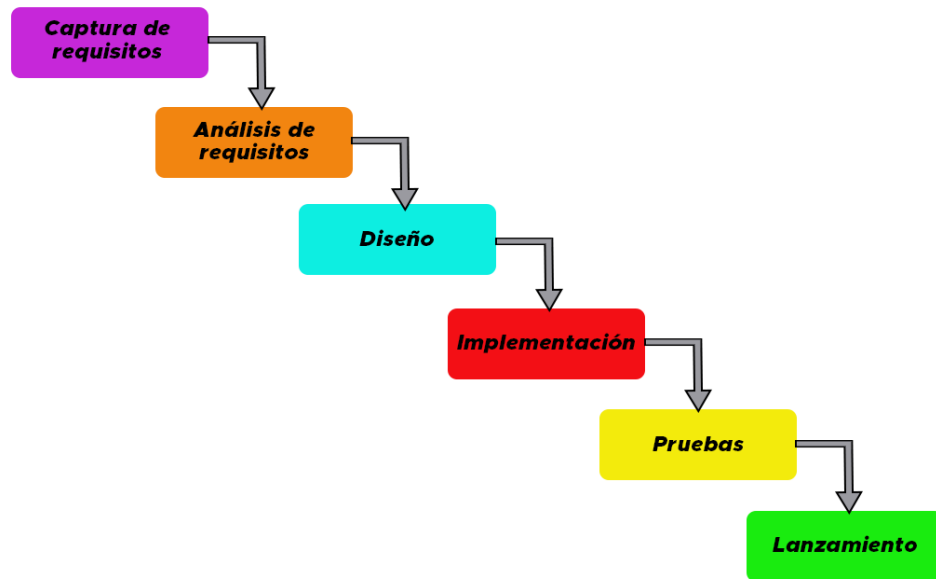


Figura 3.1: Fases de una metodología tradicional

reparten entre los distintos grupos/subgrupos. éstos suelen ser pequeños, de un tamaño máximo de 10 personas. Cada tarea se trata de forma independiente, y son propensas a tener cambios (por ejemplo, es posible que una tarea inicial X, durante el desarrollo del proyecto, no se pueda realizar ya que no se ve viable), por lo que una tarea que estaba en una fase posterior sí puede volver a una fase previa. Todos estos cambios son los que hacen que genere valor para el cliente, por lo que la comunicación es fundamental en este tipo de metodologías.

Hay diferentes metodologías ágiles: *XP*, *Scrum*, *Kanban*, *ScrumBan*, etc. Este proyecto seguirá la metodología *Kanban*.

3.2. Kanban

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

Con *Kanban* puedes visualizar el trabajo hecho y por hacer, así como las distintas fases por las que puede pasar una tarea, con el fin de que no se acumule el trabajo pendiente. Todo esto es posible ya que se utiliza una

pizarra o tablero, “tablero *Kanban*” (*Kanban Board*). La calidad del trabajo y la productividad se ven aumentadas por la mejora del flujo de trabajo en equipo.

Como uno de los objetivos de *Kanban* es saber el estado del proyecto en cada momento, y los grupos son reducidos, hay una limitación de tareas que podrá tener cada miembro del equipo en cada fase. Esto es el WIP (*Work In Progress*).

3.2.1. Tablero Kanban

Para representar las fases y las tareas se usa un tablero *Kanban*. Dicho tablero se divide en varias columnas que representan las distintas fases por las que puede pasar una tarea. Este tablero deberá tener, como mínimo, tres columnas:

- *To Do*: En esta lista se tienen las tareas pendientes por realizar.
- *Doing*: Cuando un grupo empieza a trabajar en una tarea, deberá moverla de “*To Do*” a “*Doing*”.
- *Done*: Tras saber que se ha realizado correctamente la tarea, y se ha dado por validada y aprobada, se podrá dar por terminada, por lo que se moverá a “*Done*”.

En la Figura 3.2 se puede ver nuestro tablero *Kanban* en el inicio del proyecto. Las tres columnas que se requieren como mínimo están presentes en nuestro tablero, pero, además, se han añadido cuatro columnas extra:

- *Backlog*: Corresponden con las tareas que no se pueden realizar en un presente, y que se podrán en un futuro.
- *Document Review*: En esta columna se encuentran las tareas que requieren revisión de memoria por parte del equipo. Estas tareas las revisarán quienes no hayan redactado el punto de memoria que dice la tarea.
- *Needs Testing*: Al terminar de hacer una tarea de implementación, habrá que moverla de “*Doing*” a “*Needs Testing*” y habrá que pasarla en una serie de pruebas para comprobar que se ha realizado correctamente, y no da lugar a errores.

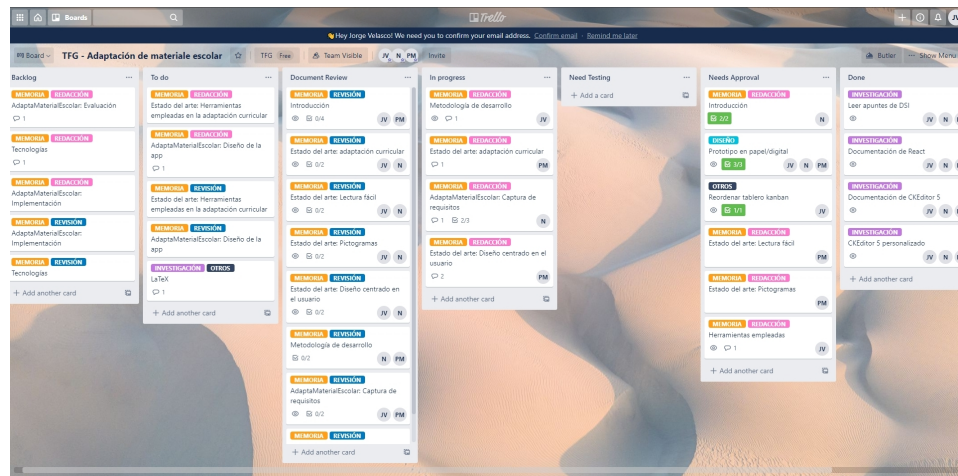




Figura 3.2: Tablero *Kanban* al inicio del proyecto

- *Needs approval*: En esta lista se encuentran las tareas que necesitan ser aprobadas por las tutoras antes de darlas por finalizadas. Normalmente se encuentran las tareas que son de “memoria”, aunque también pueden aparecer de otro tipo.






El WIP será de tres tareas en la columna “*In progress*”, y una en “*Needs testing*” para cada miembro del equipo, ya que lo formamos tres personas. Hay varias excepciones en las que no se contempla el WIP, que son en “*Document Review*” (cuando una persona realiza una tarea de redactar memoria, el resto deben revisarla en busca de incoherencias o errores en la redacción, con lo cual puede haber tantas tareas de revisión como de redacción), y en “*Needs approval*” (son tareas a la espera de aprobación o rechazo por parte de las tutoras).

Algunas tareas podrán ser asignadas a varios usuarios en el caso de que sea extensa, o requiera que alguna o todas las partes necesiten hacer lo mismo. En la Figura 3.3, se puede ver un ejemplo de este tipo de tarea, en el que todos los componentes del grupo han tenido que realizar un prototipo, por lo que solo se movería a “*Done*” en el caso de que la lista esté completa.

Así mismo, en la Figura 3.2 se observa que las tareas tienen asignadas unas etiquetas de colores:

-  Tareas relacionadas con la memoria.
-  Corresponden con tareas de redacción. Normalmente van en con-

junto con las tareas de memoria.

-  En este color se encuentran las tareas que requieren una revisión por parte del equipo. Normalmente van junto a las tareas de memoria.
-  Tareas que requieren un diseño, ya sea un prototipo en papel, una interfaz de la aplicación, etc.
-  Tareas relacionadas con la implementación, es decir, el desarrollo de código.
-  Aquí se encuentran las tareas que requiere investigación, por parte del equipo, antes de empezar a codificar o realizar cualquier otro tipo de tarea.
-  Tareas que no corresponden con ninguna de las anteriores.

3.3. Tipos de pruebas

Debido a que vamos a usar componentes, es decir, “piezas” que son independientes entre sí, haremos pruebas unitarias.

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

Ventajas de las pruebas unitarias:

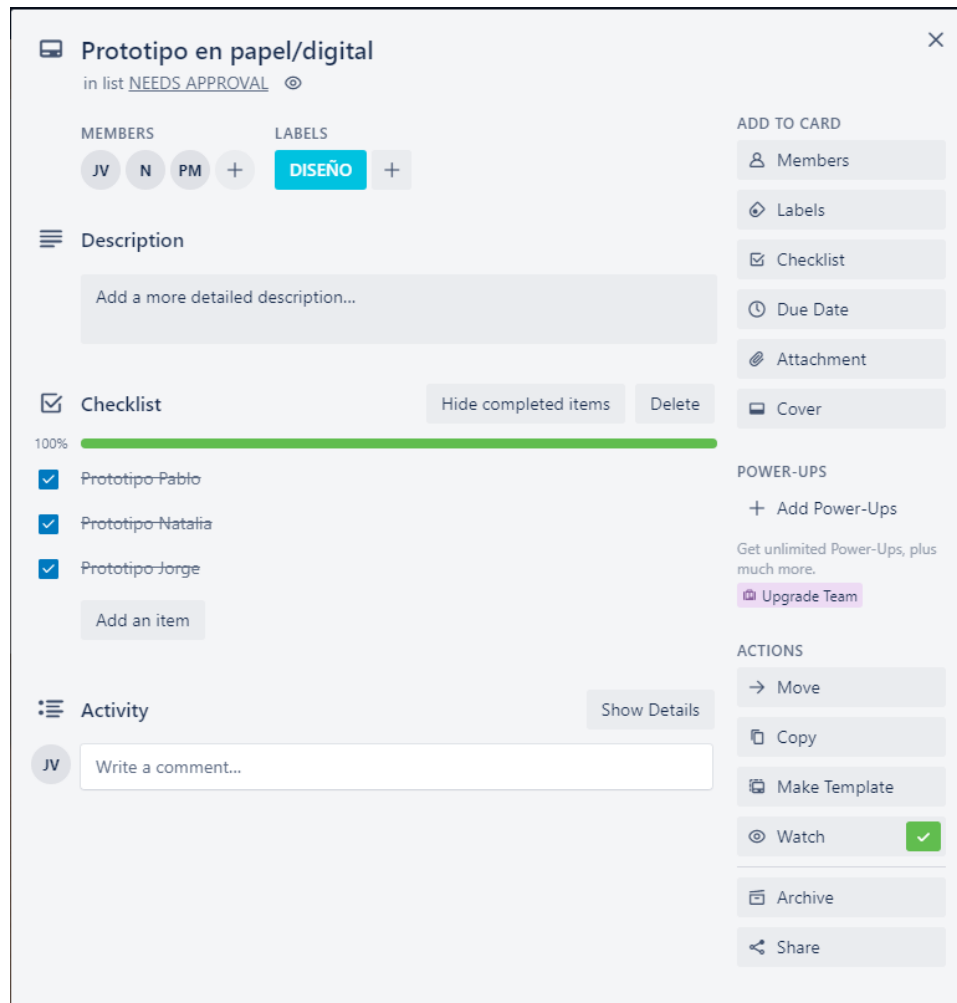


Figura 3.3: Tarea asignada a varios usuarios

- **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.

Para realizar las pruebas, hemos optado por usar *Jest*¹, una librería de testing para *Javascript*, que además es compatible con el *framework* que hemos elegido (*React*). *Jest* tiene una instalación muy sencilla, de pocos pasos, y su configuración es mínima. La documentación es completa, y contiene lo necesario para poder desarrollar estos tipos de pruebas, junto con una serie de ejemplos, realizados paso a paso.

Estas pruebas las desarrollará y realizará alguno de los miembros que no haya implementado esa parte del código, y las hará cuando:

1. La persona que haya desarrollado el código, haya completado dicha tarea.
2. No tenga tareas pendientes en la columna “*Needs Testing*” (el WIP en esa columna es de una tarea por persona).

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.

¹<https://jestjs.io/es-ES/>

Capítulo 4

Herramientas empleadas

RESUMEN:

En este capítulo se explican las herramientas y librerías que se han empleado en la creación de los prototipos. En la sección 4.1 se introduce la herramienta online Moqups. En la sección 4.2 se enseña el “framework” que se ha utilizado.

Para realizar el prototipado, algunos hemos optado por hacerlo en papel, y otros en formato digital. Para formato digital, se han usado diferentes herramientas.

4.1. Moqups

Moqups¹ es una página web enfocada en la creación de bocetos, prototipos, diagramas, etc. Es bastante completa: puedes diseñar una interfaz, o simplemente ver cómo es el flujo de un algoritmo, arrastrando los distintos elementos (o plantillas) que podemos encontrar en una aplicación (barras de progreso, etiquetas o *labels*, enlaces, diferentes tipos de ventanas, etc.) desde menú lateral al espacio de trabajo.

Existe la posibilidad de añadir comentarios e iconos, y poder crear diferentes páginas en un mismo proyecto (por ejemplo, crear varias vistas de una aplicación), así como añadir interacción a los elementos (como por ejemplo, cuando le des clic a un botón, éste realice una función específica). También

¹<https://moqups.com/>

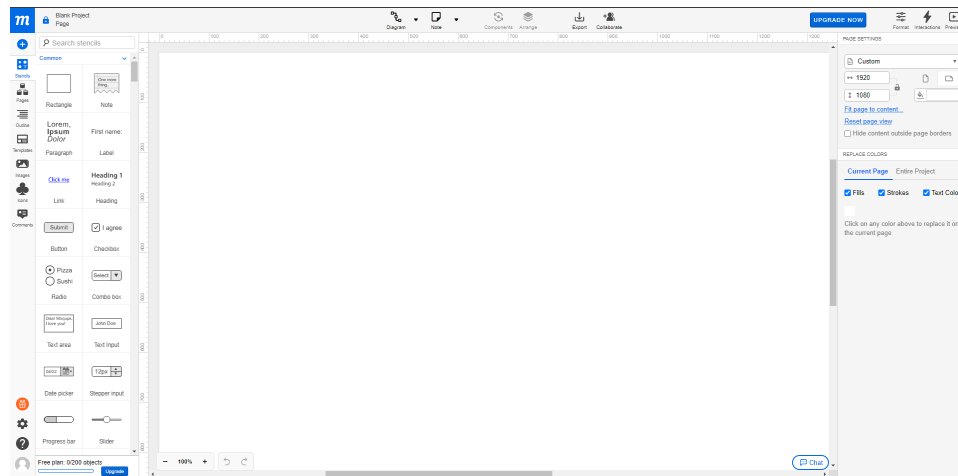


Figura 4.1: Interfaz de Moqups

es colaborativo, es decir, puedes invitar a más miembros para trabajar en equipo; y permite la gestión de roles.

En la Figura 4.1 se puede ver la interfaz de un proyecto en blanco. Se observa que en el menú lateral de la izquierda, se encuentran los apartados para la creación del prototipo (plantillas, páginas que tiene el proyecto, comentarios, imágenes, iconos, etc). En la parte superior de la página, se pueden crear figuras geométricas y añadir notas, así como poder agregar a otros usuarios, y la posibilidad de exportar el proyecto como una serie de imágenes en formato PNG o PDF; y a formato HTML. Por último, en el menú lateral de la derecha, podemos encontrar el formato de las diferentes páginas (o componentes), y las interacciones disponibles (Figura 4.2).

4.2. React

*React*² es una librería de *JavaScript*, creada por Facebook y de código abierto, que permite crear interfaces de usuario interactivas de forma sencilla. Está basada en la programación orientada a componentes, donde cada componente se puede ver como una funcionalidad distinta, es decir, como una “pieza” de un puzle.

La ventaja de usar componentes es que, al ser independientes unos de otros, si en la carga de una página web falla uno en específico, no afectaría

²<https://es.reactjs.org/>

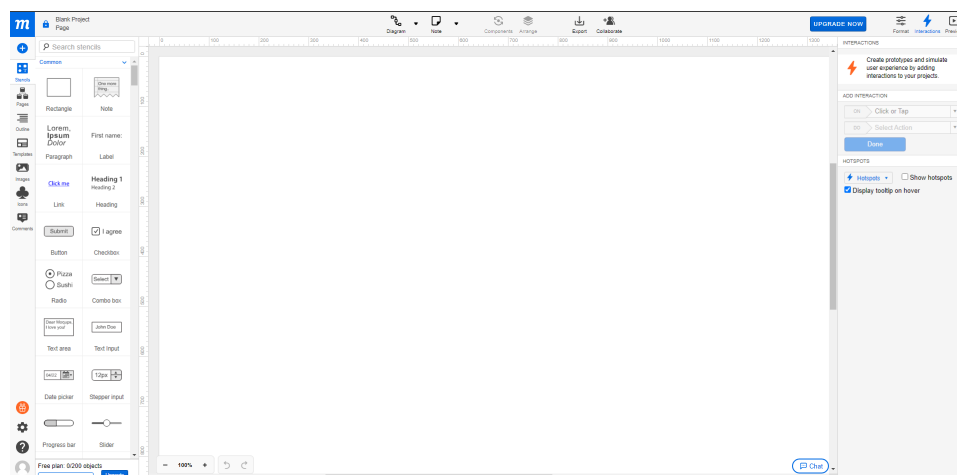


Figura 4.2: Menú lateral de interacciones

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 empleándolo facilita tanto la codificación como la lectura del código. En la Figura 4.3 se puede ver un ejemplo de una funcionalidad sin emplear el formato *JSX*; y en la Figura 4.4, la misma funcionalidad pero usando *JSX*.

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



```
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 4.3: 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 4.4: Funcionalidad empleando JSX

Bibliografía

*Y así, del mucho leer y del poco dormir,
se le secó el cerebro de manera que vino
a perder el juicio.*

Miguel de Cervantes Saavedra

—¿Qué te parece desto, Sancho? — Dijo Don Quijote —

Bien podrán los encantadores quitarme la ventura,

pero el esfuerzo y el ánimo, será imposible.

Segunda parte del Ingenioso Caballero

Don Quijote de la Mancha

Miguel de Cervantes

—Buena está — dijo Sancho —; fírmela vuestra merced.

—No es menester firmarla — dijo Don Quijote—,

sino solamente poner mi rúbrica.

Primera parte del Ingenioso Caballero

Don Quijote de la Mancha

Miguel de Cervantes

