Escuela Politécnica Superior

Fase de diseño en metodologías de desarrollo web

Grado en Ingeniería Multimedia

Trabajo Fin de Grado

Universitat d'Alacant
Universidad de Alicante

Autor:
Claudia Casanova Soler

Tutor/es:
José Vicente Berná Martínez

Junio 2019

Resumen

Actualmente, el uso de metodologías ágiles en el desarrollo de proyectos web está en auge y cada vez nacen nuevas metodologías o incluso adaptaciones de las ya existentes dependiendo de las necesidades que se desea cubrir. Esto se debe a que, ya sean empresas o particulares, quieren sacar la mayor productividad a su negocio y trabajar de la forma más eficaz posible.

En este trabajo se ha realizado un estudio de las metodologías más usadas hoy en día y se ha analizado en qué lugar no incluye la fase de diseño en su proceso. A partir de aquí se ha propuesto una nueva metodología como adaptación a lo que ya existe y funciona, pero con la mejora adicional de la agilidad en diseño, para el alcance de cualquier desarrollador del campo que desee aplicarlo.

Abstract

Currently, the use of agile methodologies in the development of web projects is booming and each time new methodologies are born or even adaptations of existing ones depending on the needs to be covered. This is because, whether companies or individuals, want to get the most productivity out of their business and work as efficiently as possible.

In this project a study of the methodologies most used today has been carried out and an analysis has been made of where the design phase is not included in the process. From here a new methodology has been proposed as an adaptation to what already exists and works, but with the additional improvement of agility in design, for the scope of any field developer who wishes to apply it.

Resum

Actualment, l'ús de metodologies àgils en el desenrotllament de projectes web està en auge i cada vegada naixen noves metodologies o inclús adaptacions de les ja existents depenent de les necessitats que es desitja cobrir. Açò se deu al fet que, ja siguen empreses o particulars, volen traure la major productivitat al seu negoci i treballar de la manera més eficaç possible. En este treball s'ha realitzat un estudi de les metodologies més usades hui en dia i s'ha analitzat en quin lloc no inclou la fase de disseny en el seu procés. A partir d'ací s'ha proposat una nova metodologia com a adaptació al que ja existix i funciona, però amb la millora addicional de l'agilitat en disseny, per a l'abast de qualsevol desenvolupador del camp que desitge aplicar-ho.

Motivación, justificación y objetivo general

La idea surgió a raíz de coger experiencia en el ámbito creativo dentro del mundo del desarrollo web como perfil Front-End. A lo largo del grado ha destacado el rol de desarrollo web en gestión de contenido creando a su vez un rol derivado de diseñador gráfico. El momento en el cual más aprendimos a gestionar proyectos de gran envergadura, más similares a los del mundo laboral, fue con el último curso de Ingeniería Multimedia.

Mediante la metodología ABP, Aprendizaje Basado en Proyectos, desarrollamos una aplicación web desde cero con la suerte de tener clientes reales y consecuentemente, una exigencia real. Aisha fue la solución web que se dio a una asociación murciana llamada Demusa que necesitaba una herramienta en la nube para toda la gestión que realizaban en su día a día entre profesionales del campo de la sanidad y la educación, más concretamente aquellos que tratan la mutilación genital femenina (MGF).

En ese tiempo fue cuando empecé a valorar ciertos aspectos en los roles y en las fases del desarrollo, pues se realizaban cambios constantes en la parte de diseño. Entonces me di cuenta de que apenas se conoce la forma de trabajo en desarrollo web con código ayudándose de una previa planificación y gestión de la interfaz gráfica que se programará.

Hoy en día existen diversas herramientas y metodologías ágiles para gestión de proyectos software, pero considero que ninguna parece mostrar importancia a la fase de diseño. Como mejora, considero fundamental establecer un proceso intermedio dentro de las metodologías ya existentes, para poder establecer las bases del diseño del proyecto y desarrollarlo de forma paralela a la implementación.

Por lo tanto, estaría más enfocada a todo aquel perfil multimedia que puede dar una perspectiva gráfica con vistas a una posterior implementación, y así, ofrecer al desarrollo eficiencia y productividad, ya que el grado aporta unas competencias específicas acerca de conocer los fundamentos de la expresión gráfica y el diseño y proyectar elementos de comunicación visual.

Además de estar estudiando el grado, llevo dos años trabajando como perfil de diseñador gráfico y Front-End en una empresa y actualmente como perfil de Front-End, por lo que debido a que apenas tengo tiempo y ya paso todo el día frente al ordenador picando código, he decidido realizar el Trabajo de Fin de Grado de algo que me resultara más familiar y conociera el tema y así hacerlo más ameno.

En mi tiempo libre me dedico al diseño gráfico y al diseño de interfaces web, por lo tanto, sería interesante realizar esta idea para poder aprender mucho más investigando y buscando información, y así mejorar mis habilidades tanto para mi vida personal y tiempo libre como para el mundo laboral.

Agradecimientos

A mi padre, por ser mi inspiración para conseguir lo que te propongas y luchar por lo que más quieres. Y a mi familia y tutor por apoyarme en todo momento.

Citas

Siempre parece imposible hasta que se hace.

Nelson Mandela

Índice de contenidos

Índice de figuras

Índice de tablas

1. Introducción

A la hora de realizar un proyecto se necesita de una gestión para buscar una armonía entre función, tiempo y lugar. Para ello, el uso de las metodologías ágiles forma un papel importante en este tema. Se necesita de una planificación y reparto de tareas entre los componentes del equipo, una previsión de fechas de entrega y una evaluación permanente. Entre muchas otras metodologías, *Kanban*, *Scrum* o *Scrumban* son las tres metodologías ágiles más usadas.

Empecé a investigar más acerca de estas herramientas de trabajo gracias al proyecto AISHA, desarrollado en el último curso del grado de Ingeniería Multimedia, el cual seguía una metodología de Aprendizaje Basado en Proyectos (ABP). Esta metodología contaba con todas las fases, entregas y exigencias con las que te encuentras en el mundo laboral.

A partir de la realización de ese proyecto, me di cuenta de algo que nos perjudicó en el desarrollo, la fase del diseño no era ágil. Una vez analizado el proyecto, comenzó la especificación de funcionalidades según las preferencias del cliente, después nos planificamos y nos pusimos manos a la obra.

Realizamos algunos bocetos sin color y a línea para organizar el contenido de forma más visual. Una vez hechos todos los diseños comenzamos a programar y maquetar las interfaces. Conforme programábamos nos parábamos en cada elemento para discutir su color, su forma, su tipografía o su grosor. Perdimos muchísimo tiempo ya que el proceso de programación se estancaba o incluso, teníamos que rehacer componentes ya programados debido a cambios de última hora en el diseño del cual partíamos.

Entonces se me ocurrió algo. A raíz de investigar acerca de herramientas para diseño web, conocí una forma de realizar los bocetos de forma rápida y con opciones de exportación de código, formas, vector, etc. Utilicé un programa para representar una interfaz tal cual la queríamos. Realizamos diversos cambios y agrupamos en componentes algunos elementos para luego al programarlo poder reutilizarlo.

La diferencia fue sorprendente; tardé 15 minutos en maquetar una interfaz partiendo del diseño ya establecido. ¿Cómo se pudo reducir días en minutos? Estableciendo bien el diseño conforme el cliente y el equipo quiere realizar luego su implementación y agrupando en componentes por futuros cambios agiliza el desarrollo del proyecto.

Más tarde, comencé a trabajar en una empresa donde se utilizaba la metodología Scrum [1]. Ésta tiene como base el ciclo de vida iterativo e incremental, el cual va liberando el producto por partes, periódicamente. Así, cada entrega o Sprint es el incremento de funcionalidad respecto a la anterior. Mediante las revisiones en las reuniones se consigue transparencia y comunicación, creando así un equipo ágil.

Es un modelo caracterizado por ofrecer una calidad de resultado basado principalmente en el conocimiento innato de las personas en equipos auto organizados, más que en la calidad de los procesos llevados a cabo.

Además, destaca el seguimiento de los pasos del desarrollo ágil, desde el concepto general de la necesidad del cliente, a la construcción del producto.

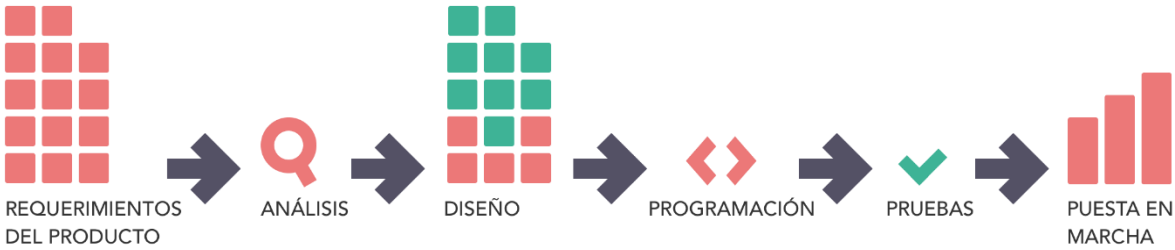


Figura 1. Gráfico de proceso de metodología tradicional (Fuente propia)

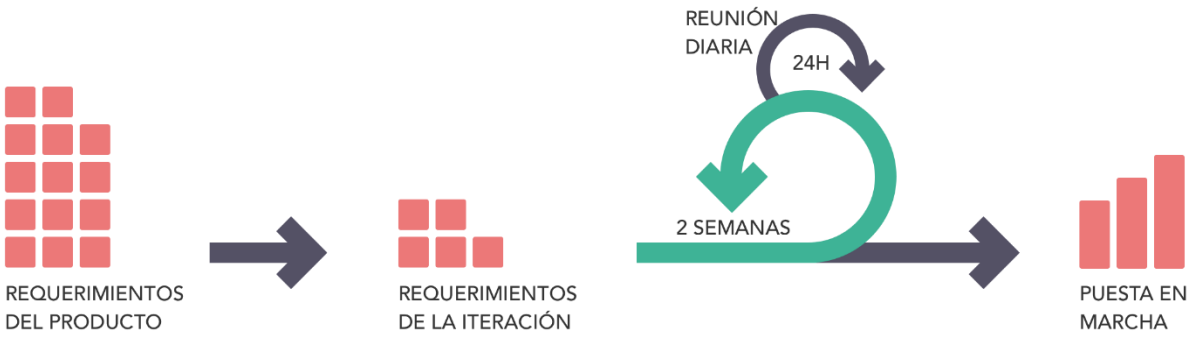


Figura 2. Gráfico de proceso de metodología Scrum (Fuente propia)

Empecé a investigar acerca de esta y otras metodologías y entonces me di cuenta de realmente no existe ninguna herramienta ágil que proporcione esa agilidad al proceso de diseño.

Mediante el TFG busco dar un lugar merecido a esa fase de diseño que queda colgando en estas metodologías. La idea es que el perfil de programador multimedia pueda aportar conocimientos gráficos enfocados a una posterior implementación, y de esta forma, agilizar el proceso y ganar en productividad y eficiencia.

Una página web se compone, a rasgos generales, de contenido y diseño. En este último, es donde más tiempo se invierte y más cambios se realizan a lo largo del desarrollo. Y muchas organizaciones deciden dejar el diseño en segundo plano y centrarse más en el contenido o la implementación. Pero al final, el usuario decide la elección de una página web y su uso a partir de lo atractivo que le sea éste. Veámoslo con un ejemplo real.

Figura 3. Página web de Ligscars (https://www.lingscars.com)

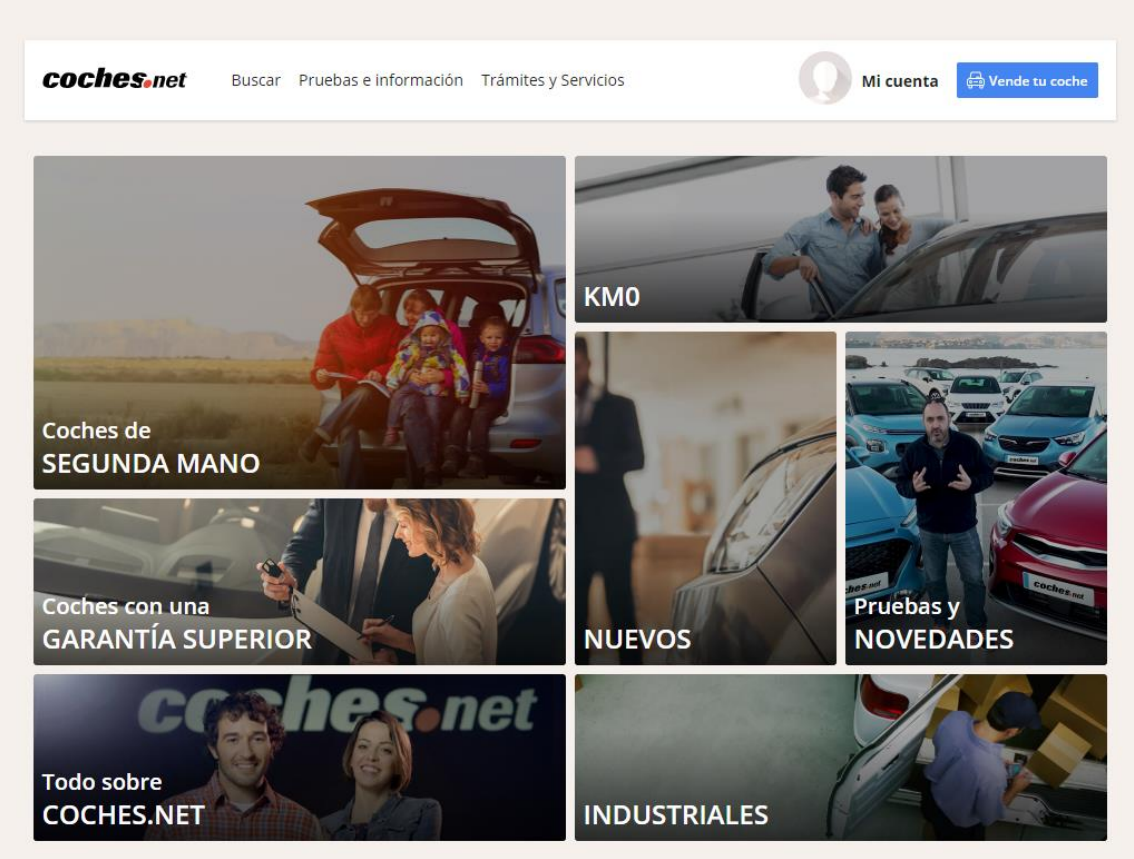


Figura 4. Página web de Coches.net (https://www.coches.net/)

Por un lado, tenemos una página web de venta de coches de segunda mano del Reino Unido. Y, por otro lado, encontramos una página web de compra venta de coches. A simple vista, se puede saber cuál es más atractiva en cuanto a diseño para entrar y quedarse navegando además de usable y accesible.

Pero ¿cómo podemos gestionar el contenido que se publica, la forma de publicarlo, el diseño de la web, la estructura, y demás?

El avance de la tecnología y la aparición de nuevas herramientas de trabajo han hecho cambiar la estructura de gestión de proyectos. Ante esta situación, resulta cada vez más complicado obtener resultados favorables. Por ello, las metodologías ágiles como modelo de gestión han sido bienvenidas en el desarrollo de proyectos en diversos campos.

Trabajar con este sistema permite adaptar la forma de trabajo al contexto y naturaleza del proyecto, focalizándose en la flexibilidad y en la inmediatez.

2. Planificación

Con el fin de llegar al plazo de tiempo establecido me he apoyado de una planificación temporal con las diferentes tareas a realizar y sus fechas límite en las que tenerlas terminadas.

Debido a que no quiero complementar trabajo y estudios, he querido realizar una estimación de fechas más ajustada y hacer el desarrollo del TFG en el curso académico entero, con fecha de finalización en junio. En el intervalo del segundo cuatrimestre he apartado el desarrollo para centrarme en finalizar con unas asignaturas. Es decir, el mes de inicio es la última semana de septiembre donde se arranca con los primeros apartados, luego más tarde comenzará el desarrollo más a fondo del proyecto. Haciendo una parada en febrero y continuando el mes de mayo. Teniendo el último mes para repasar detalles y estar planificado teniendo en cuenta cualquier imprevisto.

Contenidos	Tiempo total	Fecha límite fin
Motivación, justificación, objetivo general Agradecimientos, citas, índices Introducción Planificación Estado del arte	2 meses	24 noviembre
Objetivos Metodología Propuesta	3 meses	1 febrero
Pruebas y validación Resultados Conclusiones y trabajo futuro Referencias, bibliografía y apéndices	1 mes	26 junio

Tabla 1. Planificación temporal TFG

3. Estado del arte

En este apartado vamos a analizar y profundizar acerca de las distintas metodologías existentes y filosofías de trabajo sobre gestión de proyectos. Además, vamos a tratar de analizar en qué lugar queda específicamente contemplado el diseño, en caso de que esté contemplado, para tratar de entender y atrapar los componentes esenciales en el desarrollo de un proyecto software.

De este modo, entenderemos mejor su evolución a través de los conocimientos adquiridos, estudiando por separado cada componente esencial en el desarrollo de un proyecto, y el trabajo agilizado.

Para contextualizar la idea del proyecto, en primer lugar, vamos a estudiar qué es la gestión de proyectos y el desarrollo de productos viendo las diferentes fases existentes en el periodo de desarrollo hasta su puesta en producción y entrega. Luego vamos a analizar las metodologías tanto tradicionales como ágiles, centrándonos en las segundas, y sus respectivos procesos de desarrollo, así como la fase de diseño existente en cada una.

3.1. Gestión de proyectos

Gestionar un proyecto no es una tarea fácil. Pues requiere de una visión global que permita ejecutar un producto que necesitamos sacar adelante. Esta gestión puede realizarse de manera predictiva o ágil y como sucede en cualquier disciplina, existen defensores tanto de una como de otra. La decisión de cuál escoger dependerá de varios factores. Por un lado, la metodología predictiva otorga más importancia a los procesos, por otro lado, la metodología ágil considera más importante el valor o utilidad final del resultado que se quiere obtener.

Tendremos en cuenta diferentes factores que forman parte del proceso como son el equipo, las herramientas, los tiempos, los costes, etc., independientemente del contexto donde se desarrolle. Para ello conviene realizar antes un previo estudio de la gestión de un proyecto general y analizar las diferentes fases.

El perfil *Project Manager* será el rol encargado de asegurar que ninguna idea pueda ser tenida en cuenta sin haber considerado antes todas las áreas descritas en la siguiente imagen.

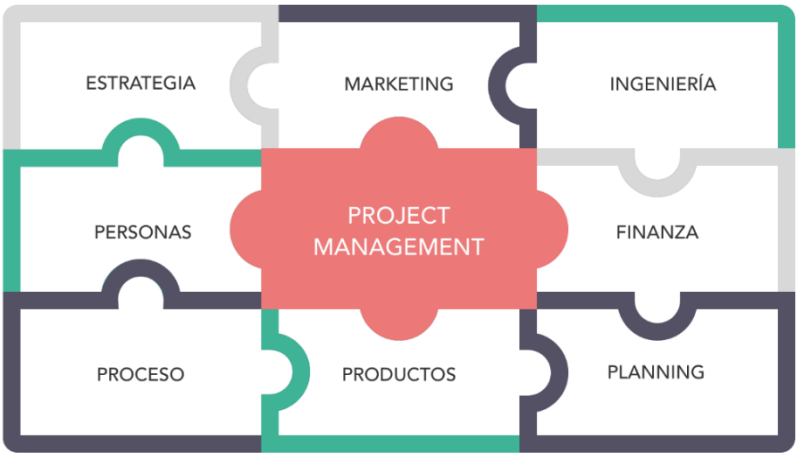


Figura 5. Perfil de Project Manager

La definición de proyecto es una secuencia de diversas tareas que previamente han sido planeadas desde el inicio hasta el fin siguiendo una línea temporal de avance, un presupuesto y el dimensionamiento de los recursos necesarios, teniendo en cuenta los requerimientos previos que establece el cliente.

Como diversos ingenieros han estado diciendo a los administradores de proyectos durante décadas: *"Puede tenerlo bueno, rápido o barato. Elija dos"*. Esto se traduce en que todo proyecto se equilibra en un triángulo de **tiempo**, **costo** y **alcance**.

Figura 6. Triángulo tiempo, costo, alcance

Estos son los tres parámetros básicos con los que tiene que lidiar el Project Manager y que determinarán en gran medida si el proyecto ha sido o no exitoso. Entre estas tres restricciones existe frecuentemente un grado de competitividad, de tal forma que el alcance normalmente aumenta el tiempo y el costo, o una restricción del tiempo puede aumentar en costos o alcances.

Si el trabajo se hace barato y rápido, la calidad sufrirá. Si se hace barato y bueno, se necesitará más tiempo para desarrollarlo. Y, si se hace rápido y bueno, lo que podrá conllevar es aumentar el número de personas en el trabajo, por lo que no será barato.

El triángulo está compuesto por puntos muy interdependientes. Se puede arreglar un parámetro, se puede arreglar un segundo parámetro, pero nunca se puede arreglar un tercer parámetro. Esto significa que al tercero se le debe permitir cambiar.

Vamos a ver un ejemplo con un trabajador y un cliente. Han llegado a una expectativa compartida sobre la calidad y el tiempo. También están de acuerdo en realizar el trabajo con una base de precios fijos. ¿Esto significa que el coste es fijo? Bueno, más bien significa que el coste del cliente es fijo, pero ¿el coste del trabajador es fijo? De una forma u otra, el trabajador ha estimado el coste del trabajo, probablemente basándose en el tiempo que tardará en realizar el trabajo. Pongamos que tardará 5 días, y en consecuencia le cuesta el trabajo. Suponiendo que es un comerciante único, hay tres resultados posibles:

1. Hace el trabajo en 5 días como se esperaba
2. Lo hace en 4 días porque le resulta más fácil
3. Encuentra dificultades y lo hace en 7 días

En cada caso, los costes son diferentes. Esto significa que, desde el punto del trabajador, la calidad y el tiempo es fijo. Pero en cada uno de los tres casos comentados anteriormente, el coste y el tiempo son diferentes. Pero desde el punto de vista del cliente se fijaron dos parámetros: calidad, tiempo y coste: Un proyecto, dos personas, dos puntos de vista diferentes del triángulo. Es por ello que se necesita establecer una jerarquía y una buena gestión de proyectos de la manera más equilibrada posible.

3.1.1. Desarrollo de un proyecto y sus fases

El producto o servicio para desarrollar surge de una idea tangible o intangible que tiene ciertas características que lo forman y la naturaleza de satisfacer una necesidad. Para ello, podemos diferenciar las siguientes fases:

1. **Toma de requisitos**. Se necesita de una especificación de requisitos funcionales para darle forma a la idea, ya sea nuestra o del cliente, y de este modo definir lo que vamos a construir y que utilidad tiene.

2. **Definición de la estrategia**. Una vez detectado el problema, se da paso al análisis y proceso de la información que disponemos, en su respectivo contexto y su orientación estratégica.

   Los objetivos de esta fase son principalmente trazar la dirección estratégica del proyecto a través de la definición del problema a solucionar con el fin de garantizar la afinidad entre el proyecto y la estrategia a seguir.

   Es importante evaluar las capacidades existentes y cuáles deberían ser adquiridas, identificar posibles usuarios y canales de distribución con punto final en su venta. Por último, pero no menos importante, cabe destacar determinar los factores relacionados con la sostenibilidad del proceso y la disposición final del producto.

3. **Diseño del concepto.** En esta fase se generan alternativas para el diseño del producto en base a los requisitos a desarrollar, incluyendo criterios de orientación al usuario asegurándose de que lo siga en las fases posteriores. Tiene como objetivo hacer la idea entendible por terceros mediante el análisis y la creatividad.

   Es fundamental asignar tiempos, recursos y fondos, además de definir la tecnología y los materiales que se utilizarán. Mediante la realización de una selección y estudio de factibilidad de aplicación de las ideas generadas se persigue refinar a su vez, los estudios sobre legislación y propiedad industrial relacionados.

4. **Desarrollo del producto: prototipo.** En esta fase se evalúa la forma en la que se va a hacer, partiendo de los requisitos iniciales. Para ello es importante experimentar con distintas configuraciones y evaluarlas para validar su nivel de adaptación en los requisitos funcionales establecidos.

   Se busca, por un lado, un producto sencillo, barato y algo funcional mediante la construcción mínima del producto que cumpla con los requisitos del cliente, ya que esto nos permitirá corregir rápido y poder iterar sobre un producto claramente definido. Y, por otro lado, facilitar el paso de la fase de diseño a la industrial y de producción, convirtiendo progresivamente la solución técnica en una solución fabricable.

Esta fase es fundamental realizarla previa a la fase de producción, por si se tuviera que realizar cualquier modificación asegurando el cumplimiento de los objetivos establecidos. Además, es aquí donde podremos comprobar la eficacia de las estrategias y definiciones planteadas, y en caso de error poder trasladarlas de forma correcta.

5. **Producción.** Esta fase se realiza fabricando una serie corta o prueba piloto, utilizando los medios productivos necesarios. El objetivo es organizar y documentar las necesidades técnicas específicas para una producción y distribución del producto en condiciones, así como definir cómo y con qué medios se realizará, definiendo los tiempos de cada uno de los procesos y los responsables involucrados en la producción.

   Una vez el producto se ha producido correctamente al plan es el momento de entregar al cliente para su lanzamiento.

Ante la problemática de resultados inesperados con retrasos en los plazos estimados y poca rentabilidad vamos a analizar los errores cometidos con más frecuencia en la gestión de proyectos del artículo *10 errores comunes en la gestión de proyectos [2]*, escrito por *Elena Gordillo Polo, Project Manager* y *Responsable Analística Web:*

- **Comienzo del proyecto sin objetivos claros.** Al comienzo del proyecto los objetivos tienen que establecerse de manera clara, alcanzable y medibles con el fin de poder orientar todos los esfuerzos de las partes implicadas de manera adecuada.

- **Equipo como actor secundario.** La metodología utilizada y la planificación no son elementos suficientes para sacar un proyecto adelante. Es necesario de la cualificación de los miembros del equipo para poder realizar las tareas. Así como, adaptar una actitud proactiva, identificando los riesgos y tomando las decisiones adecuadas, para poder hacer frente a los tiempos y cambios internos o externos que surjan en el desarrollo.

- **Problemas de comunicación.** Para evitar malentendidos, retrasos o costes adicionales causados por la mala gestión de las comunicaciones es importante tener identificadas a las personas involucradas y que muestran interés e informarles del progreso del proyecto y cualquier cambio respecto a planificación inicial establecida.

- **Análisis de resultados al final.** Para comprobar si se cumple con la planificación establecida es importante realizar un seguimiento del coste, del avance, del presupuesto y de la calidad de forma periódica. Para ello, se puede hacer uso de diversas herramientas que permiten trabajar en tiempo real y así poder tomar las decisiones más convenientes.
- **Mala planificación.** En este punto lo más importante es la comunicación entre los miembros del equipo, a través de la experiencia y confianza, estableciendo plazos de tiempos realistas. Es decir, manteniendo la visión global y con buena planificación, ayudará al equipo a conocer los pasos a seguir con la suficiente antelación como para evitar retrasos.
- **Ausencia de metodología**. Cada proyecto requiere de una organización que determinará la forma de trabajo y comunicación entre los perfiles del equipo, un seguimiento, etc.

3.1.2. Metodologías tradicionales vs ágiles

Al inicio del desarrollo de software todo el proceso se realizaba de manera artesanal, centradas en generar una documentación exhaustiva de todo y finalmente, cumplir con un plan de proyecto ya definido al comienzo mediante la división y especialización del trabajo por departamentos o funciones diferenciadas.

Este tipo de metodologías imponen una disciplina de trabajo sobre el proceso de desarrollo con el objetivo de conseguir eficiencia en el resultado. Está centrado principalmente en el control del proceso mediante una previa planificación total sobre el trabajo a realizar. Una vez todo detallado, se comienza con el ciclo de desarrollo del producto.

Esta forma de trabajo resultaba inviable debido a los altos costes de implementación a la hora de realizar cambios y la incapacidad de ofrecer buenas soluciones a proyectos donde el entorno fuese volátil.

Los errores más frecuentes eran que los objetivos del proyecto no estaban claros y no eran alcanzables y medibles para poder orientar todos los esfuerzos de las partes implicadas de forma adecuada. La mala planificación y la ausencia de una metodología que determine la forma de trabajo y comunicación perjudicaban al desarrollo.

Además, otro error es que no se realizaba un seguimiento del avance, del coste, del presupuesto y la calidad de forma periódica con el fin de comprobar si se cumple la planificación prevista.

Por último, no es recomendable pensar en el equipo como un actor secundario, es decir, no considerar que la metodología y la planificación son elementos suficientes para el éxito, independientemente de quién lo ejecute, es decir, se necesita de miembros con cualificación necesaria.

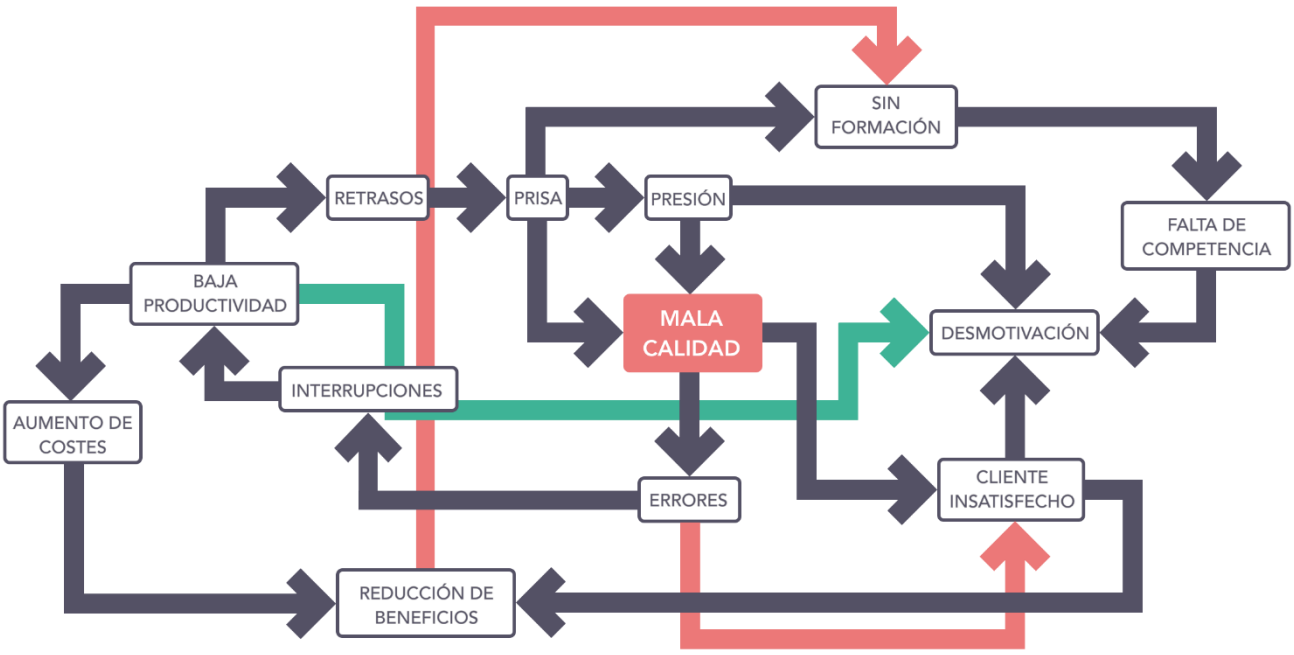


Figura 7. Círculos viciosos en desarrollo de proyectos

Como podemos observar en la Figura 7, hay numeroso factores que pueden darse en el desarrollo de un proyecto derivando todos en un producto de mala calidad. Es imprescindible hacer uso de metodologías para poder gestionar todos estos factores y buscar el equilibrio entre ellos para sacar un producto de calidad.

Debido al avance tecnológico y de software, cada vez las metodologías tradicionales utilizadas para el desarrollo de proyectos se quedaban más obsoletas en cuanto a cobertura de necesidades en el mercado, ya que se necesita un desarrollo más rápido y eficaz. Para quitar rigidez a los métodos más tradicionales surge la alternativa “agile”, la cual beneficia tanto a la organización realizadora como al cliente.

Es una herramienta enfocada mayormente al desarrollo de proyectos informáticos, ya que las metodologías tradicionales no cumplen las expectativas de agilidad de las empresas de hoy en día. Sin embargo, he decidido orientar el proyecto a desarrollo de proyectos web ya que cada tipo proyecto requiere de un método y organización específicos y este último es donde más experiencia y conocimientos he adquirido.

Stone [3] afirma que las fases de desarrollo de un proyecto se solapan y de este modo, proporcionan información entre ellas. Durante la fase de diseño, se pueden identificar problemas en los requerimientos especificados y a su vez, durante la codificación se pueden identificar problemas en la fase de diseño.

Es por ello que el proceso no sigue un modelo lineal simple, sino que emplea una secuencia de iteraciones entre las distintas etapas de desarrollo. La necesidad continua de realizar cambios a petición del cliente precisaba de un dinamismo inexistente hasta entonces.

La comunicación y la planificación eran otras cualidades que necesitaban ser transformadas, así como la autonomía de los equipos, pues dotan de adaptabilidad al proyecto. Es por ello que la gestión ágil de proyectos comienza a formar parte de los desarrollos cada vez con más frecuencia. Para su implantación los siguientes puntos serían de especial importancia:

- Cultura de empresa basada en la delegación de tareas, el trabajo en equipo, el proceso creativo y la mejora continua.
- Facilidad en la realización de cambios en el proyecto.
- Compromiso del cliente con disponibilidad para poder colaborar con el equipo.
- Uso de un espacio común donde el equipo pueda comunicarse con facilidad.
- Dedicación a tiempo completo por parte del equipo y estabilidad.
- Compromiso de la dirección de la organización formando equipos multidisciplinares y autogestionados con un líder al servicio del equipo dispuestos a resolver problemas y realizar cambios.
- Compromiso por parte de los miembros del equipo basados en la colaboración conjunta.
- Tamaño de cada equipo de entre 5 y 9 personas.
- Relación entre proveedor y cliente basada en colaboración y transparencia.

Dentro de la metodología ágil podemos encontrar distintas opciones, entre las más comunes, estudiaremos a continuación Scrum, Kanban y, por último, Scrumban.

3.1.3. Waterfall

La metodología *Waterfall* o también conocida como metodología en cascada, es el método tradicional más utilizado en los últimos 30 años. Se basa en un desarrollo secuencial estricto que comienza con las fases de análisis donde se captura y documenta los requisitos necesarios, seguidamente la fase de diseño y desarrollo, testeo con corrección de errores y ajustes finales, y por último la puesta en producción.

En el modelo *Waterfall* o *en cascada*, el progreso fluye de arriba a abajo, como su nombre indica, simulando una cascada. A continuación, vamos a analizar cada una de las fases que lo componen:

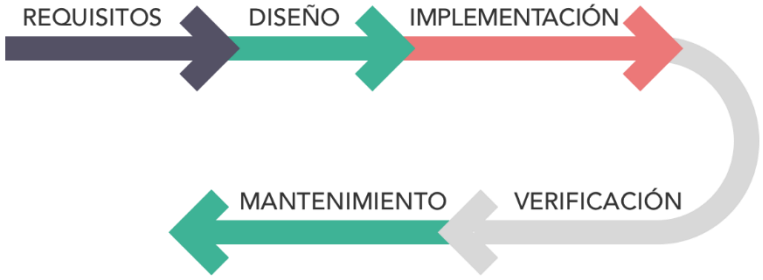


Figura 8. Fases del modelo Waterfall o en cascada

1. **Análisis de requisitos del software.** En la primera fase de análisis donde se recogen las especificaciones que el producto final debe tener, el cliente debe validar previamente la información antes de comenzar con la siguiente fase de diseño. Por ello, es la parte más importante y crítica del proyecto. Además, es sólo en ese momento cuando el cliente participa en el proyecto, el resto de las fases del desarrollo está en manos del equipo. Mediante la elaboración de una memoria o Documento de Especificación de Requisitos (en inglés llamada SRD, *System Requirements Document*), se especifica lo que debe hacer el sistema sin entrar en detalles internos. Esta etapa es crucial ya que no se podrán realizar modificaciones posteriormente para las siguientes etapas.

Para la elección de elementos de diseño, tener que decidir todos los colores. formas y disposición de elementos sólo al comienzo del proyecto, puede resultar una tarea excesiva para el cliente, es por ello que sólo se establecen los principios que regirán el diseño, como son los colores corporativos, estilos a utilizar, etc.

2. **Diseño del sistema y del programa.** El diseño del sistema atomiza y organiza el sistema en subelementos que se recogen en un documento llamado Documento de Diseño de Software (o en inglés SDD, *Software Design Descriptions*), el cual contiene la estructura relacional global y la conexión entre las distintas partes. En el diseño del programa se realizan los algoritmos y se determinan las herramientas a usar en la implementación. Además, se diseñan las interfaces del proyecto e interacción entre ellas.

3. **Implementación**. Es la etapa donde se codifica haciendo uso de prototipos y pruebas para corregir errores. Se puede hacer uso de bibliotecas y componentes reutilizables dentro del proyecto para mejorar el proceso. En esta fase se realizan los diseños establecidos en la fase anterior. Al haber dado por finalizada la anterior etapa ya no se podrán realizar modificaciones en el diseño.

4. **Verificación.** Proceso en el que se unifican las partes ya programadas para componer el sistema y se comprueba su correcto funcionamiento. Antes de la entrega final se realizan las correcciones necesarias. Una vez finalizada la comprobación se entrega al cliente con el fin de ejecutar y comprobar que cumple con sus exigencias. En caso de cambios en el diseño, tendrían que ser sencillos y pequeños, pues realizar cambios en gran escala puede afectar a la funcionalidad del proyecto y esto causará un aumento inviable en coste y/o tiempo. Una modificación o cambio mediante el desarrollo de cualquiera de las fases, implicaría reiniciar el ciclo completo.

5. **Mantenimiento**. Es la última y la más crítica etapa, pues el mantenimiento del software puede que no cumpla con las expectativas. Es por ello que definir el diseño y la funcionalidad desde un comienzo puede arrastrar trabajo hasta la última etapa, ya que es donde el cliente vuelve a tener opinión y puede descartar o modificar cualquier cosa.

A diferencia de *agile*, en este sistema, cada una de las etapas que forman parte del desarrollo, deben darse por concluidas antes de comenzar la siguiente etapa. Es más, cada una de ellas tiene un hito bloqueante que no permite avanzar a la siguiente sin antes resolverlo de forma adecuada.

Ventajas	Desventajas
No se requiere estrictamente la presencia del cliente	Eficacia de los requisitos
Presupuesto cerrado	Los cambios pueden ser difíciles de implementar o costosos
Planificación y diseño más sencillos y directos	Posibilidad de que el cliente no esté satisfecho con el producto
Es más fácil de medir y seguir	Solucionar problemas requiere más tiempo, esfuerzo y dinero

Tabla 2. Ventajas y desventajas de Waterfall

Analizando la fase de diseño en este tipo de metodología, podemos comprobar diferentes factores que no favorecen esta etapa. Debido a que no es frecuente que el cliente explicite clara y completamente los requisitos, sería necesario establecer una guía estricta de estilos al comienzo del desarrollo.

Esto supone un grave problema, como indicaba en la fase inicial anteriormente, ya que el cliente debería decidir al comienzo del proyecto cómo quiere diseñar gráficamente toda su aplicación, pues no se podrán realizar cambios en el estilo en todo el desarrollo.

Para visualizar la aplicación tal cual se desea, en la fase de diseño no sería suficiente el uso de mockups con wireframe, es decir, sería necesario acordar colores, tipografías, estructuras y contenido de absolutamente todas las interfaces. Esto haría de la fase de diseño una etapa demasiado extensa e inviable en la planificación.

A continuación, vamos a ver dos ejemplos de mockup de una de las interfaces del proyecto realizado en el ABP, como se menciona anteriormente, llamado AISHA. En concreto es una interfaz compleja donde necesitábamos mostrar una enorme cantidad de datos de la manera más sencilla y organizada posible. Es por ello que hicimos uso de los mockups en la herramienta de prototipado Sketch [4] para determinar tal cual la disposición de los elementos, los colores, las tipografías, las sombras, los elementos adicionales, etc.

Mediante Sketch, una herramienta para diseño de interfaces con posibilidad de interacción y exportación en distintos formatos y código, realizamos todos los mockups previos a la implementación.

Una vez realizados tal cual los queríamos, los revisó el profesor y después procedimos con la implementación. Como ya establecimos desde un principio los elementos básicos del diseño y la estructura visual, la implementación fue mucho más productiva y los cambios a realizar más sencillos, pues partíamos de la estructura base que se quería.

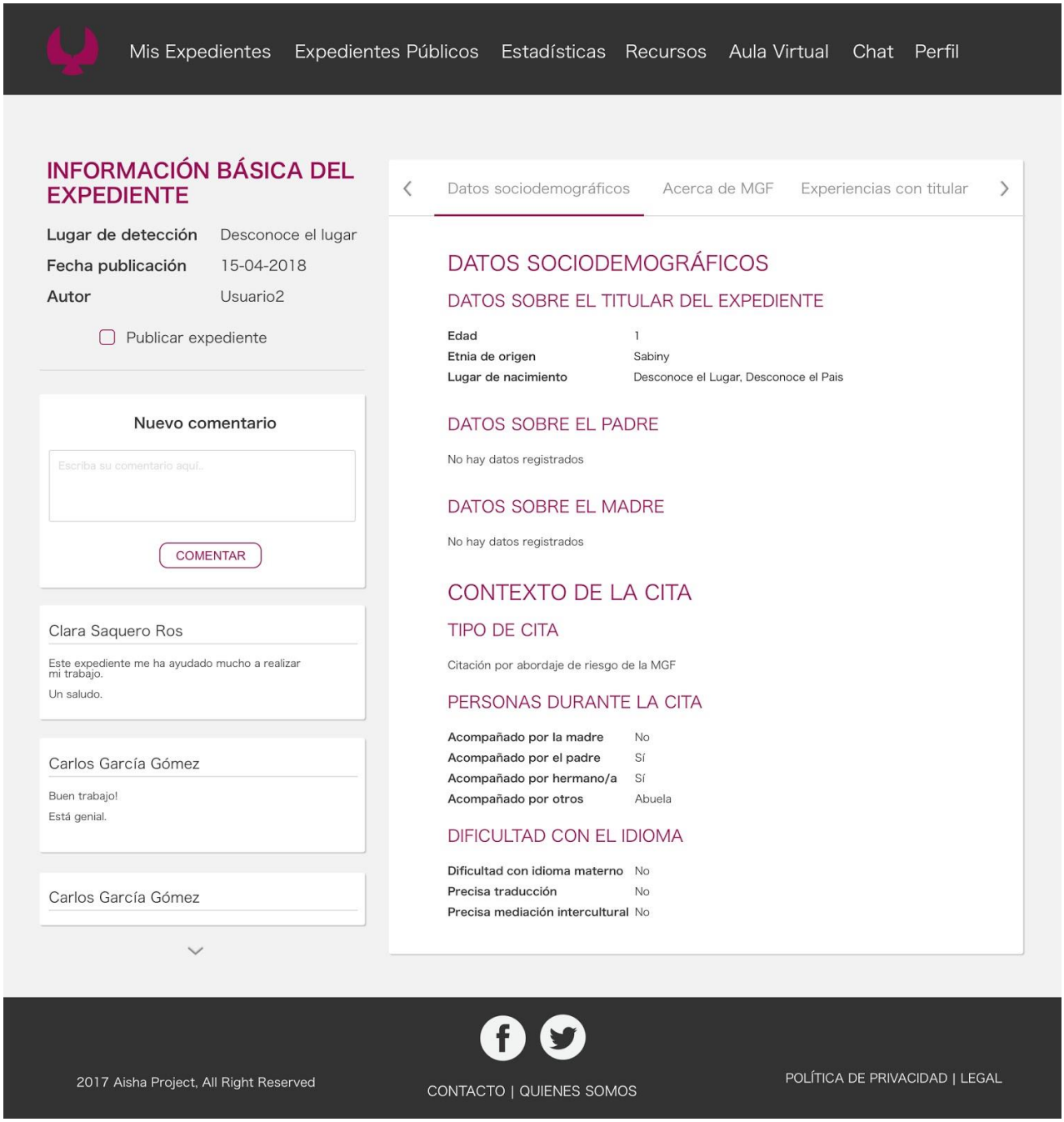


Figura 9. Mockup con Sketch del proyecto AISHA

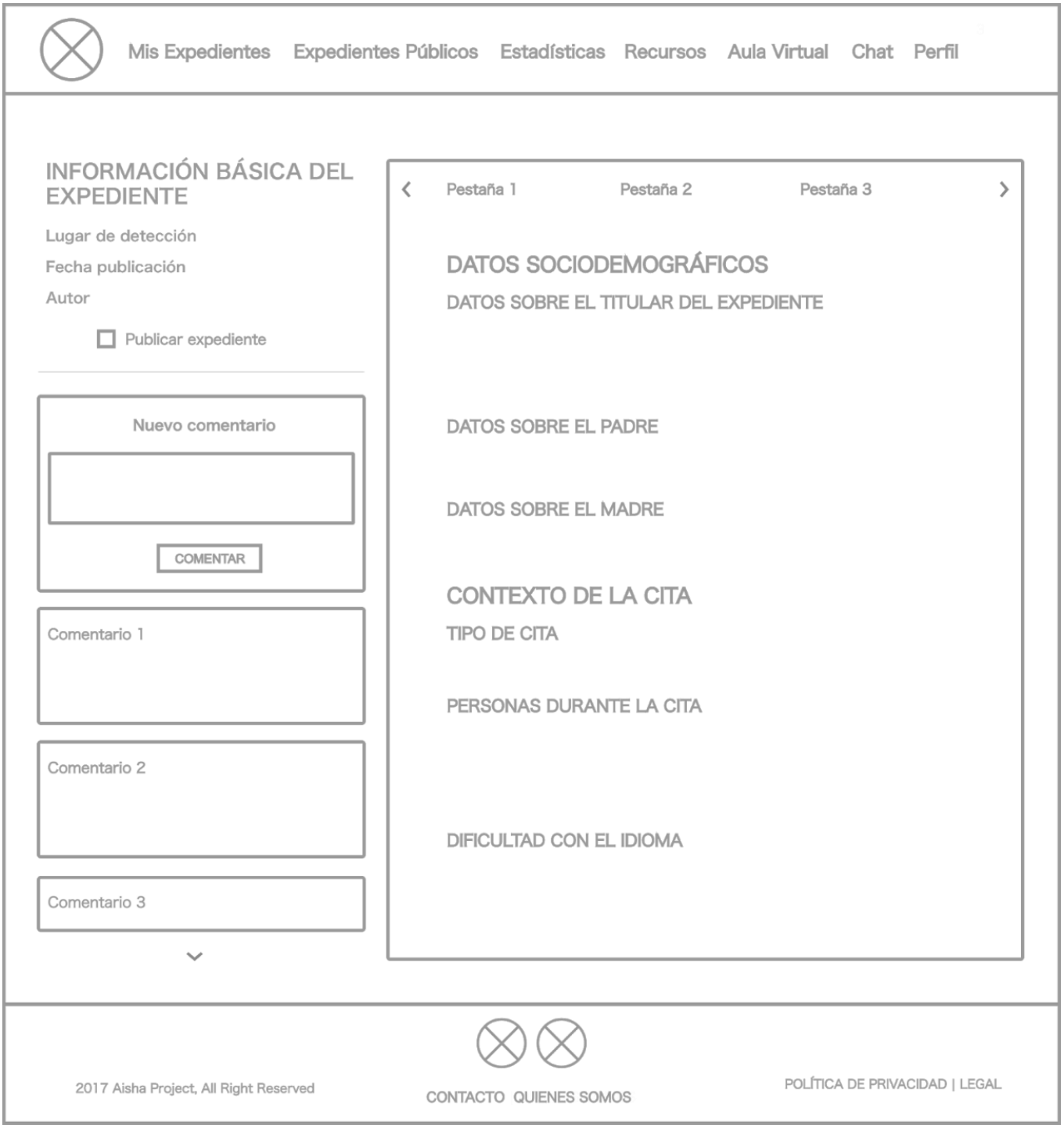


Figura 10. Mockup con wireframe del proyecto AISHA

Como se puede observar en la Figura 10, el uso de mockups con wireframe también puede ayudar a visualizar la estructura del contenido, pero a diferencia de hacerlos con Sketch, retrasa la fase de implementación, ya que no establece los colores y los elementos básicos de diseño desde un principio.

La aplicación de la metodología en cascada, como podemos encontrar en *El Trabajo de Investigación de la Metodología en Cascada de la Universidad Estatal de Milagro* [5], se orienta mejor al desarrollo de proyectos de corto plazo y de poca innovación. Para proyectos más grandes, se suele hacer uso de algunas variantes del modelo en cascada debido a su simplicidad y eficacia produciendo retroalimentación entre etapas para que sea más fácil realizar cambios. Pero si nos centramos en la metodología Waterfall como tal, la fase de diseño no encajaría de la mejor forma y no se podría agilizar en este tipo de metodología.

Aunque a diferencia de agile, el diseño ya está establecido desde un primer momento, por lo que, aunque sea demasiado rígido, hace el esfuerzo al principio e introduce menos posibilidades de errores en el diseño. Este último punto puede ser un punto a favor interesante para el desarrollo del proyecto.

3.1.4. Scrum

El proceso Scrum, aplicando de manera regular un conjunto de buenas prácticas para trabajar en equipo, proporciona orden en situaciones caóticas aceptando la naturaleza cambiante de los proyectos. Mediante el uso de directrices, es la metodología ágil que da soluciones específicas a los problemas que se van encontrando de la forma más eficaz posible.

Esta información la podemos encontrar en la página web enfocada únicamente a proyectos ágiles llamada *proyectosagiles.org* [6] donde explica en este caso, qué es Scrum y sus características.

Se realizan entregas parciales del producto final, dando prioridad a aquellas que aportan valor al cliente. Es por ello que está enfocado a proyectos desarrollados en entornos complejos donde se necesita de resultados tempranos, donde los requisitos son cambiantes y donde prima la innovación, la flexibilidad y la productividad.

El proceso que sigue es mediante ciclos temporales cortos y de duración fija, llamadas iteraciones, que pueden ser de 2, 3 y hasta 4 semanas. En cada iteración se tiene que generar un resultado, el cual será parte del incremento del producto final.

El punto de partida es una lista donde se recogen los requisitos priorizados por el cliente o *Product Owner* respecto a su coste. De este modo, quedan repartidos en iteraciones y fechas de entregas.

Suponiendo que las iteraciones sean de dos semanas, vamos a analizar las principales actividades que se llevan a cabo:

1. **Planificación de la iteración**. Podemos dividir este paso en dos partes: por un lado, la selección de requisitos con duración de dos horas donde el cliente presenta la lista mencionada anteriormente y el equipo pregunta al cliente las dudas que le surgen. Además, sitúa cada requisito en el plazo de entrega correspondiente dependiendo de su prioridad.

   La segunda parte sería la planificación de la iteración, con duración de dos horas también. En esta parte el equipo realiza una lista de tareas de la iteración necesarias para el desarrollo de las funcionalidades seleccionadas. De manera conjunta, el equipo estima el esfuerzo y se asignan tareas entre los miembros.

2. **Ejecución de la iteración**. Para esta parte se hace uso de un tablero o pizarra llamado *Scrum Taskboard*, en el cual cada día el equipo realiza una reunión de 15 minutos. En esta reunión, el equipo revisa el trabajo que cada miembro ha realizado, el trabajo que realizará a continuación y, por último, se analizan posibles obstáculos que puedan impedir llegar a los objetivos establecidos. Esto permite realizar cualquier adaptación necesaria con el fin de cumplir plazos de entrega.

   El perfil de *Scrum Master*, el organizador del equipo es quien se encarga de eliminar aquellos obstáculos que el equipo sea incapaz de resolver y lo protege de posibles interrupciones externas.

   A su vez, el cliente refina la lista de los requisitos junto al equipo, para las próximas iteraciones. De este modo, si hubiera cualquier modificación, ya sea de implementación o diseño, se pueden planificar de nuevo los objetivos del proyecto.

   Estar en contacto con el cliente fortalece la guía de estilos, ya que en el momento que desee puede comunicar modificaciones en ésta, dando prioridad a aquellas interfaces donde se considere por parte del cliente y del equipo y el presupuesto establecido. Sería entonces interesante dividir el estilo del proyecto en componentes, donde se validen por iteraciones y no afecte a las fechas de entregas y al flujo de trabajo establecido.

3. **Inspección y adaptación**. Por último, al finalizar cada iteración, el último día se realiza una reunión, dividida en dos partes. En primer lugar, tenemos la revisión o demostración, con duración de hora y media, donde el equipo presenta al cliente las funcionalidades logradas en la iteración. Si es necesario, el cliente realiza las modificaciones convenientes de manera objetiva respecto a los resultados presentados, con el fin de ir replanificando el proyecto de la forma más eficiente posible.

   Después se realiza la retrospectiva, con la misma duración, donde el equipo analiza si han tenido algún problema que les haya impedido progresar adecuadamente, o cómo mejorar, con el fin de aumentar en productividad. Las decisiones son tomadas por el *Scrum Master* para que no genere obstáculos al equipo.

Ventajas	**Desventajas**
Gestión regular de las expectativas del cliente con resultados tangibles	El cliente siempre espera informes con fecha exacta
Flexibilidad y adaptación a las necesidades del cliente	Demasiadas reuniones
Productividad y calidad	En caso de renuncia es complicado el reemplazo del rol
Motivación del equipo	Estimación de costes inexacta en tareas mal definidas

Tabla 3. Ventajas y desventajas de Scrum

En resumen, se basa en el desarrollo incremental de los requisitos del proyecto en bloques temporales de corta duración y fijos en el tiempo, dando prioridad a aquellos requisitos que el cliente considere y el coste que conlleva.

Gracias a la sincronización del equipo en las reuniones diarias en las que se realizan las adaptaciones necesarias y a que el cliente pueda tomar las decisiones en función de lo que observa en las entregas, el proyecto adquiere un control empírico, a diferencia de la metodología tradicional donde el control es predictivo.

Como explica el artículo de *Control Empírico y Predictivo* [7], el control predictivo se refiere a firmar requisitos iniciales detallados en el comienzo del proyecto con vistas a prepararlo para la entrega final. Por otro lado, el control empírico asume la predicción como variable del proyecto mediante adaptaciones y bajo inspecciones en función de los resultados, siguiendo un ciclo de mejora continua.

Además, sería más fácil establecer la conexión entre programador y diseñador, como puede ocurrir con el perfil de Ingeniero Multimedia, para acordar las decisiones de estilo del proyecto.

Los equipos deben estar formados por miembros con habilidades complementarias, de este modo se genera una sinergia mediante la que cada miembro aporta lo mejor y minimiza sus debilidades. Esto dota al equipo de lo necesario para realizar tareas complejas y compuestas de subtareas independientes. En Scrum lo recomendable es crear equipos multidisciplinares, en el que siempre se disponga de un experto para realizar cada tarea compleja que surja, y también autoorganizativos con el fin de generar un ambiente de trabajo beneficioso tanto para el propio equipo como para el trabajo a realizar.

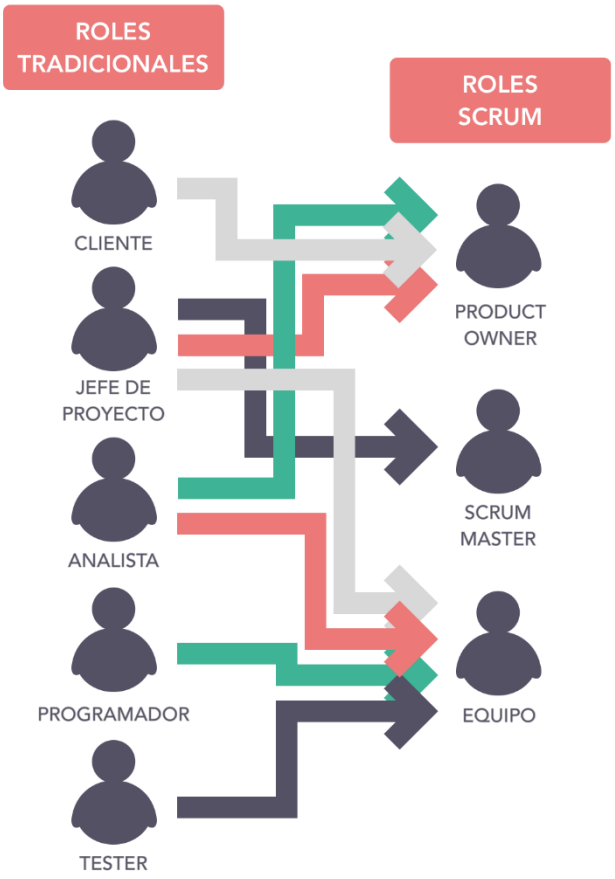


Figura 11. Diferencia entre roles tradicionales y Scrum

A continuación, vamos a analizar cada uno de los perfiles de **Scrum**:

1. **Product Owner (PO).** Es el responsable del producto, es decir, su función es asegurarse de que el producto sea el esperado por el cliente y los entregables ofrezcan las funcionalidades correspondientes que más valor ofrezcan al cliente.
   Al ser el único medio de comunicación con el cliente, debe extraer toda la información necesaria, así como el feedback de lo realizado y crear un plan de entregas con fechas de entrega parciales del producto.

2. **Scrum Master.** Como comentábamos en la ejecución de la iteración en el desarrollo del proyecto, es el perfil encargado de asegurar la autoorganización del equipo, mediante el apoyo y ofreciendo todo aquello que necesiten, así como la protección de presiones externas que puedan afectar a la organización.

   Su responsabilidad es ayudar al equipo a ser productivo, ya sea ofreciendo formación como herramientas o comunicación. El manejo del entorno del proyecto es también tarea de este rol, pues deberá asegurarse de que se hace uso correcto de todos y cada uno de los artefactos utilizados, que explicamos en el siguiente apartado, y sus respectivas actualizaciones.

3. **Equipo**. Es el grupo de profesionales que desarrollan el proyecto gracias a sus conocimientos técnicos, mediante la realización de las *historias* o tareas a las que se comprometen en el inicio de cada *sprint*. Además, sería recomendable que los programadores conozcan de forma superficial tanto la parte de servidor, como la de interfaz. Así como el diseñador, que tenga conocimientos básicos de programación web y viceversa, ya que Scrum apuesta por la profesionalidad y el enriquecimiento entre los miembros del equipo.

Además de garantizar la disciplina existente, aportar motivación al equipo y ofrecerles lo que necesite, el Scrum Master hace uso también de una serie de artefactos mediante los cuales los miembros del equipo se gestionará el trabajo de manera adecuada:

- **Product Backlog.** Al igual que en cualquier otro proyecto, al comienzo del desarrollo se realiza una toma de requisitos del producto por el cliente susceptibles al cambio tanto en volumen como prioridad.

Para facilitar la tarea, el *Product Owner* realiza una pila del producto donde puede ordenar fichas divididas en hitos que componen parcialmente el proyecto. Estos hitos se denominan *historias de usuario* y suelen representarse como se observa en la Figura 12:

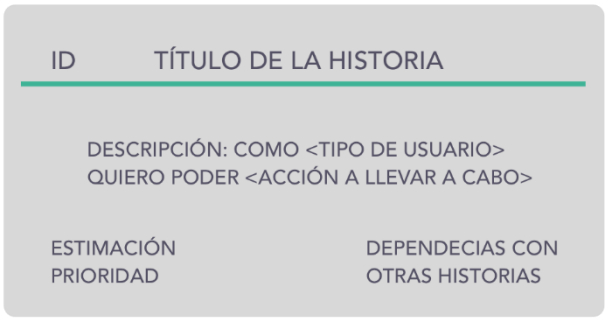


Figura 12. Historias de usuario

Una vez realizadas todas las tarjetas, se ordenan por prioridad estando las más prioritarias arriba del todo. Esto facilita la reorganización en caso de modificación. Si la tarea resulta ser inalcanzable por su tamaño, se puede dividir en subtareas para realizarlas más fácilmente.

- **Sprint Backlog**. Su función es ofrecer al equipo una visión centrada en la iteración en curso para evitar distracciones del objetivo parcial establecido. En resumidas palabras, es donde se encuentran las historias a realizar en el *sprint*.

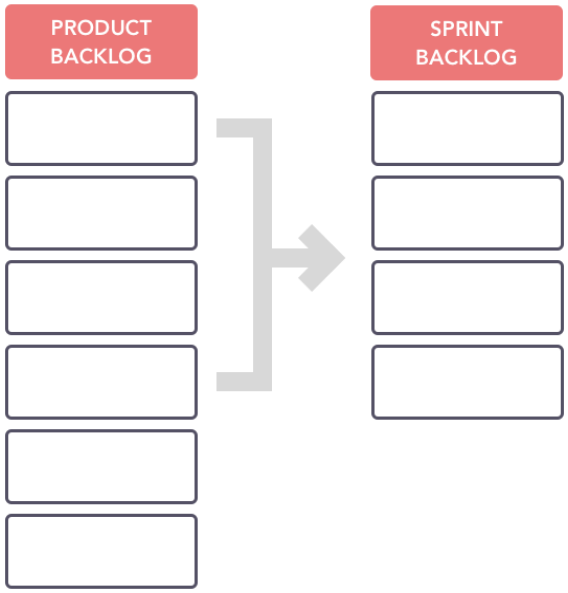


Figura 13. Product Backlog vs Sprint Backlog

- **Scrum Board**. Es el medio que recoge el estado concreto de un *Sprint* en cada momento. Esta pizarra suele contener el trabajo pendiente, el que se está realizando y el realizando. De forma opcional se puede poner una zona de impedimentos o puntos negativos o positivos que el equipo observa en el *sprint*.

  El tablero se rellena al comienzo de la reunión del *Sprint Planning,* que explicaremos más adelante, donde se colocan a la izquierda las tareas del *backlog* que el equipo se ha comprometido a realizar. Al finalizar cada *Daily*, este debe ser actualizado y las tareas no realizadas vuelven al *Backlog*.

Figura 14. Tablero Scrum

1. **Lista de bloqueos**. Se trata de una lista que contiene los bloqueos que impiden al equipo continuar y los que han sido solucionados, representado como una columna más de la Scrum Board para verlos de un vistazo. Los miembros del equipo son los encargados de avisar de los bloqueos en las historias que están realizando, pero es tarea del *Scrum Master* dar solución a esos bloqueos.

2. **Burn-down chart**. Para ofrecer una visión global del desarrollo de cada sprint, se utiliza una gráfica que contiene en su eje X los días que dura el Sprint y en el eje Y los puntos que el equipo se ha comprometido a realizar a lo largo de éste, es decir, la estimación que realizan al crear las historias medida en puntos de complejidad.

Mediante la unión de los puntos que cada día se ponen en su día correspondiente y uniendo los puntos en Y que quedan pendientes, se obtiene una gráfica que muestra el progreso que ha habido a lo largo del sprint.

Gracias al gráfico que se puede observar en la Figura 15, se pueden localizar problemas que ha podido tener el equipo o malas prácticas realizadas. Por ejemplo, una línea recta indicaría un periodo de bloqueo del equipo en un determinado tiempo.

Figura 15. Burn-Down chart

Para establecer todos y cada uno de los pasos que forman parte del ciclo, es responsabilidad del *Product Owner* de obtener la información necesaria para establecer un nivel de priorización de las *historias de usuario*.

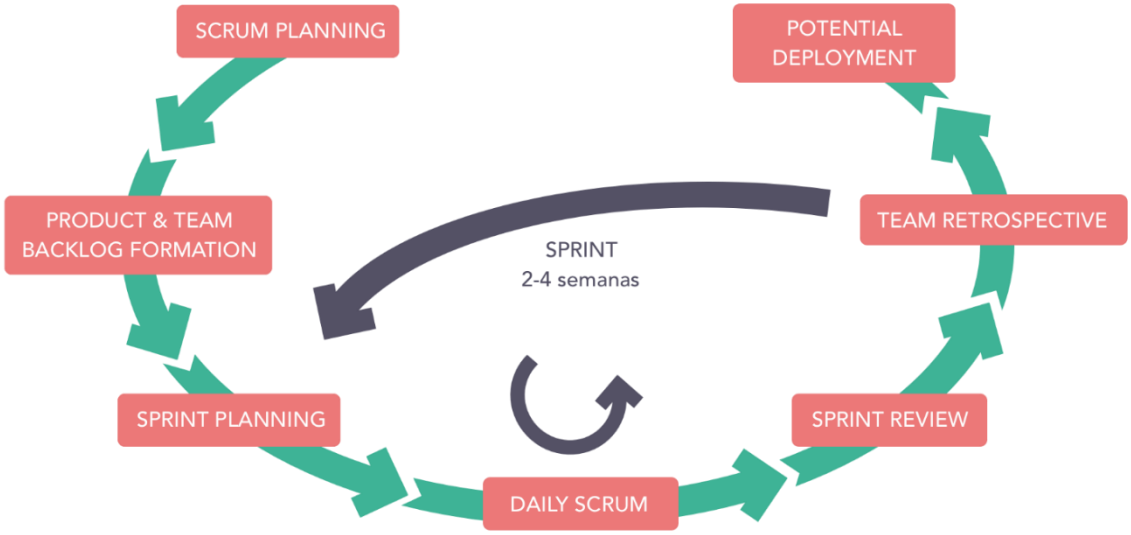


Figura 16. Ciclo de vida Scrum

Como comentamos anteriormente, uno de los objetivos de Scrum es eliminar en la medida de lo posible la documentación y las reuniones constantes. Es por ello que introduce unas determinadas reuniones que representan las distintas fases del ciclo, las cuales estudiaremos a continuación:

1. **Sprint Planning**. Es la fase inicial donde se realiza la planificación, es decir, el *Product Owner* selecciona de su *Sprint Backlog* las historias, previamente analizadas, en el próximo entregable. Cuantas más pequeñas sean, más funcionalidades independientes tendrán el equipo y menor posibilidad de historias sin acabar al final del Sprint.

   Esto beneficia a la parte de diseño, ya que atomizando en componentes cada vez más pequeños y unitarios, conseguimos más independencia entre elementos de diseño. Gracias a esto, a la hora de realizar cambios es más sencillo localizar el elemento a cambiar y no retrasar el desarrollo del proyecto.

   Primero se expone a historia al equipo, para poder exponer y resolver dudas. Después, el equipo vota la estimación del peso de la historia, en base a la complejidad. Luego, se apunta en una tarjeta y se traslada la historia al *Sprint Backlog*.

   Una vez realizados todo esto, el *Scrum Master* traslada lo artefactos a la *Scrum Board* y crea la *Burndown Chart* del Sprint para comenzar a trabajar.

2. **Daily Scrum**. Es el método de control por parte del equipo, en vez de terceros, el cual consiste en una reunión breve diaria donde se comparte el progreso de las actividades del *sprint*.

   Es decir, se comunica las tareas a las que se comprometieron el anterior día a completar, las que se comprometen el día de la reunión y, además, si existe algún bloqueo en el desarrollo. En caso de bloqueo sería responsabilidad del *Scrum Master* de ayudarle a dar una solución, como hemos comentado anteriormente.

3. **Sprint Review**. Cuando finaliza el tiempo estimado del sprint, se realiza una demostración con las funcionalidades terminadas. De esta forma, los miembros del proyecto pueden valorar el progreso y el cliente, tiene la oportunidad de ver su evolución y detectar posibles cambios, mejoras o diferencias de interpretación de requisitos.

Es en esta reunión donde el diseño puede ser cambiado, por lo que sólo habría que modificar los elementos ya componetizados previamente, sin afectar al proyecto en conjunto. De esta forma, el trabajo se puede realizar de forma paralela entre miembros del equipo.

4. **Team Retrospective.** Es la reunión previa que se realiza antes de dar por finalizado el sprint. Es decir, es un análisis que realiza el equipo y el Scrum Master acerca de los puntos fuertes y los puntos débiles del equipo, así como la metodología de trabajo llevaba a cabo hasta el momento. Además, para reflejar el avance se hace uso del Burndown chart y así poder mostrar al equipo fallos a la hora de emplear la metodología.

3.1.5. Kanban

Kanban es un conjunto de técnicas que ayudan al desarrollo de proyectos mediante gestión visual y uso de un tablero donde se muestran los requisitos en curso de cada estado del ciclo de vida. Es por ello que es recomendable utilizarlo como herramienta auxiliar en el uso de la metodología ágil Scrum.

Es una metodología más orientada al proyecto en el que prima la entrega continua y el tiempo de resolución de cada petición. Suelen ser entornos donde no es sencillo planificar, pues el alcance no está definido con anticipación, o donde las prioridades cambian con frecuencia. Son mayormente, entornos donde es necesario resolver peticiones lo antes posible, por lo que es recomendable combinarlo con Scrum.

Abordar más trabajo del que se puede afrontar, produce colapsos en el desarrollo del proyecto que afecta a la velocidad del trabajo que se saca y dificulta el cambio de tareas, la recuperación de información, la colocación del espacio del trabajo, etc.

Esta problemática la resuelve centrándose en un determinado número de tareas para conseguir efectividad en la finalización de éstas y mejorar el flujo. A continuación, vamos a analizar cada una de las distintas propiedades que conforman Kanban:

1. **Gestión visual.** Mediante un panel físico o electrónico, se representa el flujo de trabajo donde se organiza en columnas cada parte del proceso. El poder visualizar el flujo, los requisitos y el estado de cada uno de ellos ayuda al equipo a visualizar el trabajo de forma global y facilita identificar bloqueos.

2. **Limitar el trabajo en curso.** Para fomentar la finalización de requisitos sin acumularlos, Kanban ofrece limitar el número máximo de elementos de trabajo en curso de cada fase. Gracias a la visualización de las colas, se identifican bloqueos, por lo que se reduce el tiempo de entrega y se priorizan las tareas.

3. **Mejora continua**. Los miembros del equipo identifican impedimentos que puedan perjudicar al flujo e intentan aplicar nuevos enfoques para solventarlo.

4. **Just In Time (del ingles, JIT).** Kanban está orientado a la entrega continua de valor mediante este sistema de organización para optimizar el proceso productivo mediante la supresión regular de desperdicios.

Como hemos visto en el apartado anterior, la metodología Scrum se basa en procesos muy centrados en la planificación donde el equipo trabaja una lista de tareas ordenados por prioridad. Una vez se asignan un determinado número de tareas en cada sprint, cualquiera de las tareas que no se hayan priorizado deberá introducirse en la siguiente iteración. En Kanban, esta gestión de iteraciones se basa en un método de mejora continua en el que los desarrollos se priorizan en el momento. Es decir, se pueden introducir tareas en cualquier momento e ir priorizándolas en el desarrollo.

Además, Scrum trabaja iteraciones fijas en tiempo y cantidad de funcionalidades. Kanban no tiene iteraciones fijas, pues planifica y desarrolla en base a las prioridades que se van introduciendo en el desarrollo del producto. Es por ello, que el analista puede tener dificultades para gestionar la carga de trabajo, pues no tiene una planificaciones suficientemente detallada.

Ventajas	Desventajas
Facilita el entendimiento gracias a la exposición de información a los miembros del proyecto	Sólo funciona bien cuando el sistema es de justo a tiempo (JIT, en inglés, Just In Time)
Facilita la integración con metodologías ágiles	Sólo es aplicable a partes de consumo en el mismo día que se produce
Acepta el ingreso de cambios de última hora con facilidad	Si se utiliza en unidades muy de alto coste el almacenamiento y manejo es de alto coste también
Se adecúa a los proyectos en mejora continua	No dispone de anticipación en caso de situaciones imprevistas muy grandes.

Tabla 4. Ventajas y desventajas de Kanban

En cuanto a los roles, Kanban es un método evolutivo en el que se van introduciendo constantemente cambios, por lo que no hay roles a introducir que adopten este método. A diferencia de Scrum, no prescribe roles o reuniones, es decir, han sido creados mediante la evolución de la metodología en proyectos donde se ha adoptado. Podríamos encontrar los siguientes extraídos de *Metodología Kanban [8]*:

1. **Service Request Manager (Product Owner)**. Es el encargado de gestionar la demanda y los requisitos. Su papel es fomentar la transparencia en el sistema para poder dar la prioridad adecuada a las tareas.

2. **Service Delivery Manager (Flow Manager)**. Es el rol encargado de la gestión del flujo de trabajo.

Las tarjetas son indicadores visuales de un trabajo en concreto que tiene que ser producido. Se mueven en dirección de izquierda a derecha, no viajan a procesos anteriores. En ellas se debe identificar de manera visual y con suficiente información las distintas tareas. Estas tarjetas son prácticamente iguales a las de Scrum, pudiendo añadir fechas límite en caso de tener fechas cerradas importantes.

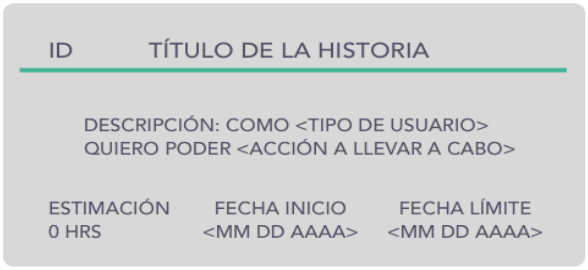


Figura 17. Tarjeta Kanban

Como se mencionaba en el anterior apartado de la metodología Scrum, el tablero de Scrum también se puede llamar Kanban, por lo que es el mismo que el de Scrum, pero el de Kanban tiene unas pocas diferencias que veremos a continuación extraídas del artículo *Kanban vs Scrum boards: 11 major differences* de *RealtimeBoard[9]*.

En el tablero de Scrum, todas las historias se añaden al principio y después de una reunión de retrospectiva al final de cada *sprint* se retiran. Cuando sucede esto al finalizar cada sprint, el tablero se limpia y se comienza de nuevo. En Kanban lo elementos se etiquetan para poder visualizar el proceso que sufren, pues siempre van entrando nuevos en mitad del desarrollo y hay que reorganizar la prioridad continuamente, sin llegar a retirarlos del tablero pues es un proceso continuo.

Cómo incorporación al tablero Kanban, encontramos un límite de cantidad de historias de usuario o tareas a realizar. Esto se debe a que en Scrum es el trabajo conjunto con todos los miembros del equipo quien realiza todas las tareas de cada iteración o sprint, en cambio, en Kanban, cada tarea es asignada a un miembro del equipo.

Además, el tablero no pertenece a un equipo en concreto como Scrum, que el tablero es sólo del equipo que tenga el sprint correspondiente. En Kanban el tablero es accesible por cualquiera, no es necesario que pertenezca a un equipo en concreto. Para entenderlo mejor, a partir de la Figura 18 podemos observar que, si el equipo está compuesto de dos miembros, sólo podrán existir dos tareas en curso, no más, porque cada tarea es asignada a un miembro.

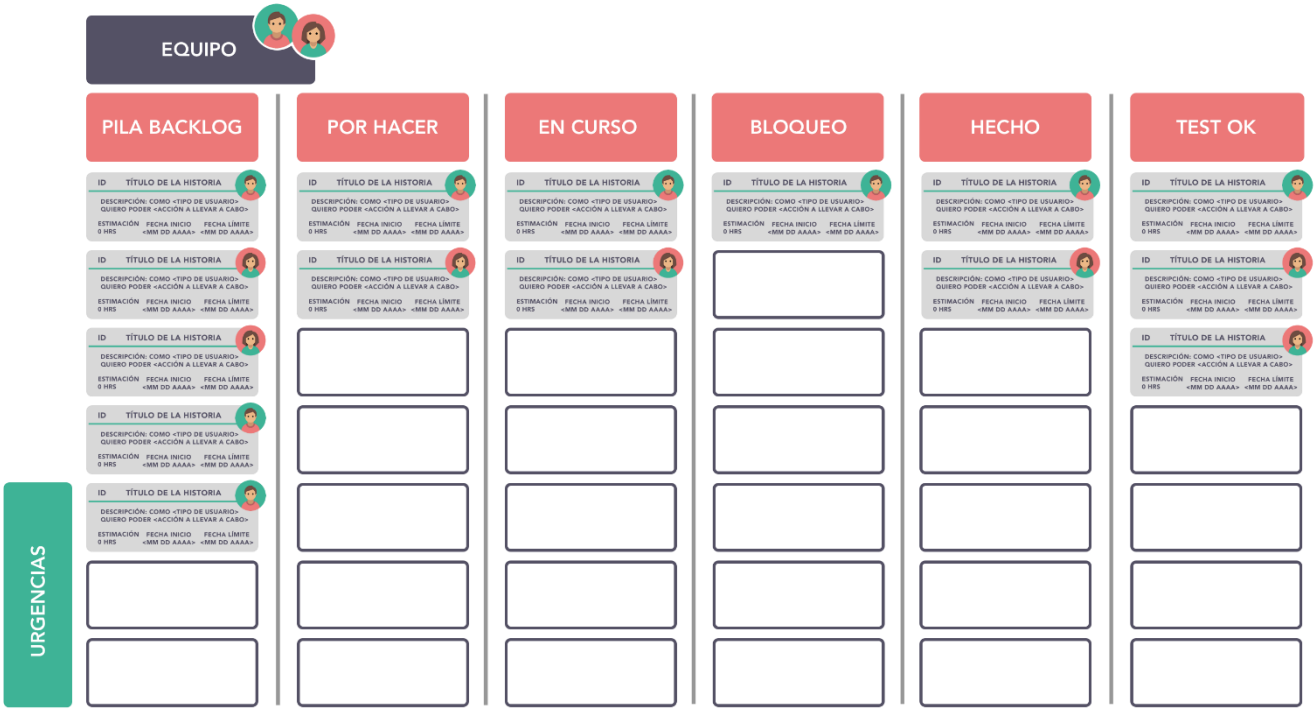


Figura 18. Tablero Kanban

Es importante que el proceso esté bien definido para no cometer errores a la hora de mover un elemento, realizando el movimiento de tareas de izquierda a derecha. Esta definición se realiza en una reunión de planificación. Además, los elementos del tablero deben ser pequeños pues tienen que poder ser desarrollados dentro de su previsión. La sección de "urgencias" del tablero Kanban ayuda a representar la prioridad entre tareas. En Scrum no es necesario ya que se establecen ya en orden de prioridad.

En el tablero se debe indicar la tarea en la que cada miembro del grupo está trabajando, por lo que deberá ser actualizado de forma regular a medida que los elementos progresan. Además, el tablero no sólo puede ser modificado por el equipo como sucede en Scrum, en Kanban el Product Owner también puede realizar modificaciones.

A diferencia de Scrum, en Kanban cada miembro del equipo no está asignado específicamente a un tipo de tarea, es decir, si una persona ha terminado su tarea puede escoger entre ayudar al tester a probar que funciona o escoger otra tarea de la cola.

Al igual que los roles, Kanban también ha adaptado reuniones específicas a su ciclo de vida que no existían en un principio.

- **Reunión diaria.** A diferencia de Scrum, en estas reuniones no se detiene en el análisis profundo, es decir, mediante el tablero se revisa el flujo y posibles bloqueos.

- **Replenishment (o rellenado)**. Para poder alimentar el tablero con las distintas columnas, el rellenado se suele hacer conforme el desarrollo. Estas reuniones, en caso de ser necesarias, ayudarán a gestionar las tareas si hay un flujo de trabajo mayor.

- **Retrospectivas.** La finalidad de esta reunión es para y observar cuando algo no va bien. Es necesario responder preguntas acerca del posible trabajo que se esté haciendo mal o cómo mejorar, pero mediante la acción, es decir, convertir las respuestas en planes de acciones reflejados en el tablero.

3.1.6. Scrumban

Cualquier metodología puede ser una herramienta sometida a adaptaciones y cambios según el contexto en el que se lleven a cabo y las necesidades que se deseen cubrir. Como explicaba en los apartados anteriores, Kanban es una herramienta que se suele utilizar para complementar Scrum, por lo tanto, cogiendo lo mejor de Scrum y Kanban, nace Scrumban. Ésta afirmación la podemos encontrar en el libro *Essays on Kanban System for Lean Software Development [10]* de Corey Ladas publicado en 2009 donde explica, en pocas palabras, cómo Scrumban nace a raíz de encontrar fallos en Scrum que se pueden mejorar aplicando los principios de Kanban.

La entrega de valor de manera constante de Kanban genera mayor confianza en los clientes, pero también es más difícil de planificar. Por otro lado, Scrum focaliza mejor el trabajo mediante la planificación, pero se adapta mucho peor a los cambios de planificación que puedan surgir.

Esta metodología ofrece numerosos beneficios para proyectos mayormente complejos, que actualmente sigan una metodología Scrum:

- **Proyectos de mantenimiento.** En este tipo de proyectos es de real importancia entregar resultados parciales para poder seguir avanzando en el desarrollo.

- **Proyectos con posibles errores de ejecución.** Son proyectos donde surgen errores que llevan a replantear al equipo si los métodos usados son correctos y, por consiguiente, un análisis de retrospectiva de la evolución de las tareas.

- **Proyectos en los que los requisitos sufran variaciones.** En esta última clase de proyectos, el cliente no especifica desde un principio las condiciones de forma fija, es decir, las expectativas y funcionalidades se van introduciendo en el desarrollo del proyecto y sus etapas.

Para beneficiar este tipo de proyectos, Scrumban ofrece una serie de características que solventan determinados problemas que puedan surgir:

- Ayuda a mantener un seguimiento constante del desarrollo en **tiempo real** del proceso de ejecución que se lleva a cabo.

- Gracias al conocimiento del estado real del proyecto, es posible **introducir soluciones** en los momentos oportunos ante errores que puedan surgir.

- Mediante las reuniones, el equipo **mejora la interacción** entre los diferentes miembros lo que beneficia muchísimos aspectos como son la comunicación y el trabajo conjunto.

- Ya que se tiene una visión segmentada del trabajo global, es más fácil **realizar análisis** de las tareas que se llevan a cabo.

- Todos estos aspectos ayudan a **aumentar la productividad**, sobre todo en proyectos más complejos.

- Por último, favorece **mayor adaptabilidad** de las herramientas según las exigencias del proyecto.

Pero, al igual que sucede en Scrum y Kanban, en Scrumban no se refleja una fase de diseño ágil tampoco. A continuación, vamos a desarrollar las principales adaptaciones que sufre Scrumban respecto a Scrum y Kanban:

1. **Visualizar el trabajo.** Scrum no requiere obligatoriamente de un tablero, es opcional. En cambio, Kanban está definido bajo la utilización obligatoria de un tablero. Por lo que utilizar Scrum con tablero es un primer paso hacia Scrumban.

2. **Imponer límites del trabajo**. Si adaptamos Kanban a Scrum, es esencial establecer un límite en el flujo de trabajo, pues si el equipo de desarrollo está compuesto por dos personas, sería recomendable que cada miembro del equipo disponga de una tarea, aunque en caso de bloqueo, se podría añadir una tarea más por miembro para poder continuar el trabajo. Por el contrario, Scrum trabaja con tantas tareas como requiera la iteración o sprint en el que se esté trabajando, independientemente de los miembros del equipo.

   Para trasladarlo a Scrumban, este límite se aplicaría al proceso, en el que la columna del tablero sólo albergue una número determinado de tareas, es decir, un límite ya que en no se trabaja con iteraciones o sprints. En caso de bloqueo, como no se pueden añadir más tareas, el equipo deberá trabajar conjuntamente para resolver el problema, mejorando el trabajo en equipo y la comunicación. Si el elemento está totalmente bloqueado y no se tiene control, sólo queda eliminarlo del tablero.

3. **Agregar más columnas**. Agregando una columna para “En progreso (Doing)” y para “Testing” con sus propios límites de trabajo en progreso. Para que los desarrolladores puedan poner sus tarjetas en “preparado para testear” se puede añadir una columna de “Hecho (Done)”.

4. **Empezar a ordenar.** En las reuniones de planificación o Sprint Planning, se asigna una tarea cualquiera a una persona cualquiera y una puntuación. Pero esto puede tener un impacto en el rendimiento del equipo a largo plazo ya que puede tener más prioridad otra tarjeta o incluso, otro miembro del equipo se puede quedar sin saber qué hacer.

En Scrumban se cambia la asignación temprana y se establece un orden de prioridad. Para ello es recomendable utilizar una columna para “Hacer (To-Do)” que se diferencie del “Backlog” donde se haga una selección de determinadas tareas a realizar con orden de prioridad. El orden se establece de más a menos prioritario. Si un miembro del equipo escoge una tarea que no se ve capaz de hacer, puede consultarlo con el equipo para que se le permita escoger una segunda o incluso tercera.

5. **Dejar de estimar**. En base a que cada acción que se considera que no tiene valor se considera un desperdicio, se traslada a que cada elemento del Backlog que no tiene valor se considera también, un desperdicio. En Scrumban no hay estimación, ni puntos de historia. Las sesiones de planificación de Sprint se convierten en una sesión exclusivamente para priorizar el trabajo, y no sobre dimensionar el trabajo.

   Es por ello que la velocidad no se puede medir con los puntos de historia como Scrum, si no por número de tareas realizadas en un determinado tiempo. Como se basa en la velocidad en que se realiza el trabajo, y no necesariamente tiene que estar alineado por una estimación, se puede obtener una previsibilidad sin estimar contando las historias en vez de contar los puntos de las historias.

   Aunque no se realicen estimaciones en forma de puntos de historias, es muy importante mantener la discusión que precede esta reunión, la comunicación del equipo es importante para hacer una estimación y una previsión del trabajo.

6. **Planificación activada**. Partiendo del principio de no probar lo que no se puede desarrollar y no desarrollar lo que no se puede probar. Esto se establece con el orden de las columnas, es decir, “En progreso (Doing)” sólo se mueven las tareas que estén en “Hacer (To-Do)” y sólo se mueven las tareas para empezar a hacer pruebas de “Hecho (Done)” a “Testing”. De esta forma organizamos con un orden ya establecido las tareas, para evitar bloqueos y problemas en el desarrollo.

   En vez de establecer un Sprint, donde las tareas se seleccionen de un Product Backlog a un Sprint Backlog con una sesión de planificación de dos semanas, en Scrumban iniciamos una sesión de planificación a demanda.

Para esto se hace uso de otra línea, como la de limitador de número de tareas, en el que cuando el número de tareas cae por debajo de cierto punto, es el punto de activación donde se requiere de una sesión de planificación necesaria. Estas sesiones son más ligeras, sin puntos de historia y sólo focalizándose en la priorización de tareas.

Apoyando la mejora continua de los procesos, Scrumban se puede considerar una evolución de Scrum junto Kanban. Es decir, tanto los equipos como las funcionalidades pueden beneficiarse de la visión general que proporciona el proceso Kanban, ayudando a supervisar y gestionar el trabajo, detectar cuellos de botella y mejorar la eficiencia en los procesos de desarrollo que se llevan a cabo. Para evitar límites en los procesos o procesos sin valor, es recomendable utilizar Scrumban para apoyar esta mejora continua.

3.2. Diseño

Pero además de las metodologías de gestión de proyectos, también es importante analizar otros puntos que forman parte de esta fase de diseño. Una vez tenemos las metodologías establecidas, toca analizar distintos aspectos que formarán parte de mi propuesta: Design Thinking, prototipado y el Atomic Design.

3.2.1. Design Thinking

La importancia que se le ha dado en los últimos años a utilizar el diseño como forma de pensar dentro del desarrollo de proyectos deriva en *Design Thinking*, una metodología de innovación enfocada al usuario. Partir de una idea y no saber en qué va a mejorar el producto en el cliente al que va dirigido no es una idea. Crear valor es lo importante, y esto se consigue si se pone al usuario en el mismo escenario.

Esta metodología combina la mentalidad de negocio con la creativa, mediante el trabajo al mismo tiempo de procesos racionales y estructurados, con procesos emocionales e intuitivos; con momentos de foco y convergencia y momentos de expansión y divergencia.

Al contrario que el resto de las disciplinas focalizadas en encontrar la solución a través de los fallos, de forma rápida y barata, *Design Thinking* utiliza el mismo esfuerzo en definir de forma correcta el problema con el objetivo de establecer un equilibrio entre el mundo de los negocios, la tecnología y las personas con los valores humanos, y así aportar valor a muchos niveles.

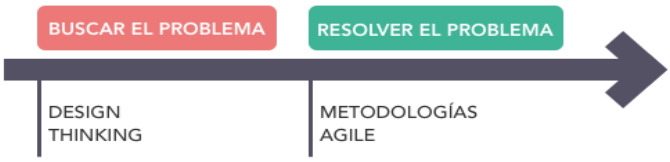


Figura 19. Objetivo de Design Thinking

Design Thinking se resume en mucho enfoque en lo práctico y experimental con procesos de colaboración entre personas y centrado en los objetivos para resolver el problema. Para ello establece una serie de valores:

- **Enseñar en vez de hablar**. Comunicar la visión del producto de la manera que cree impacto y experiencias a través de lo visual.

- **Focalizarse en los valores humanos**. Empatizar con las personas para las que se trabaja y tener en cuenta su feedback.

- **Claridad**. Producir una visión coherente de los problemas complicados transmitiendo a los demás inspiración para resolverlos.

- **Aceptar la experimentación**. El prototipado no es una parte simple para reflejar las ideas, por lo que requiere una parte de innovación donde se piensa y aprende.

- **Estar atento al proceso**. Saber en qué punto se sitúa cada miembro del equipo en el proceso de desarrollo, qué métodos usar y a qué nivel y por último, saber cuáles son los objetivos.

- **Predisposición a la acción**. *Thinking Design* es un término equivocado; se trata más bien de hacer que pensar. Con la predisposición de hacer y cambiar el pensamiento.

- **Colaboración radical**. Reunir a innovadores con habilidades diferentes y soluciones para emerger de la diversidad.

El proceso de esta metodología en rasgos generales se podría diferenciar en dos etapas: la exploración del problema y la exploración de la solución. El *Design Council de Reino Unido* definió cuatro grandes fases y las representó mediante rombos, haciendo referencia a la divergencia de los procesos, donde se busca información, y a la convergencia con la definición del problema y solución.

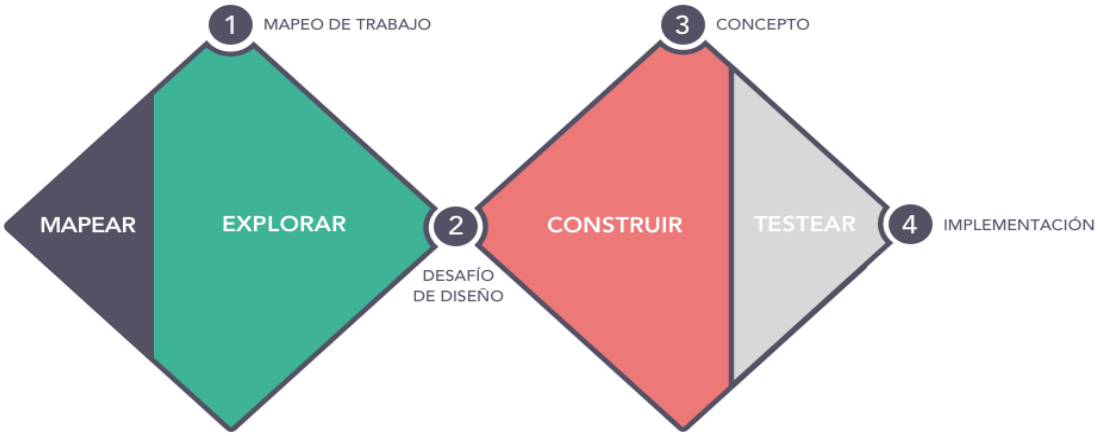


Figura 20. Fases de Design Thinking

1. **Mapeo**. En la primera etapa el proyecto surge de una idea inicial o problema definido con el objetivo de ser resuelto. Entonces comienza el mapeo, donde se delimita el contexto de trabajo en relación con lo conocido y desconocido, el alcance del equipo y los objetivos. Entra dentro del rombo de proceso divergente, donde cuanta más información se obtenga más beneficiará al proyecto, pues se descubren las necesidades del cliente para el que se diseña.

2. **Exploración**. La siguiente etapa después del mapeo es el proceso convergente donde se define e interpreta los objetivos respecto a las necesidades generando el problema a resolver como punto de anclaje.

3. **Construcción**. En esta etapa es donde se generan ideas y soluciones al problema establecido. Luego, se exploran las posibilidades y se empiezan a crear los primeros prototipos. Forma parte del proceso divergente donde se ejecutan las ideas mediante prototipos capacitados para ser usados como modelo de comunicación y discusión. Dependiendo de cómo sea el proyecto, se irán descartando aquellas ideas o prototipos que no encajen y dando prioridad a los que sí.

4. **Testing**. A partir del primer prototipo diseñado de la solución final, se testea y se pivota hasta la validación o el descarte. Es decir, se trata de un proceso iterativo donde el principal objetivo es tener feedback continuamente de las personas involucradas en el proyecto. Antes de comenzar con esta fase es necesario definir *KPIs* (del inglés, Key Performance Indicator), es decir, un conjunto de métricas que se utilizan para sintetizar información acerca de la productividad de las acciones que se llevan a cabo con el fin de cumplir los objetivos marcados.

Design Thinking más que una metodología, es una forma de trabajo, una actitud basada en cómo piensa el diseñador y cómo puede adaptarse para resolver un problema dado. El eje principal del proceso es el cliente que recibirá el producto. Gracias a los equipos multidisciplinares que forma, resulta ser una herramienta de co-creación donde prima la investigación cualitativa de datos.

Gracias a su flexibilidad, diversas entidades adaptan el método de la forma que más conviene en el proyecto, personalizando las herramientas y creando nuevos conceptos. Es por ello que mi propuesta apoya esta metodología y además, utilizarlo con las metodologías y los conceptos vistos previamente.

3.2.2. Prototipado

Tradicionalmente se realizaba el diseño de productos dirigidos por la funcionalidad o tecnología, llevado a cabo por expertos que se basan en sus conocimientos, dejando las necesidades del usuario en segundo plano. Hasta ahora, la definición de prototipo ha sufrido muchas modificaciones. En la actualidad, lo que se plantea es prestar real atención a los usuarios, entenderlos a ellos y lo que les motiva. A partir de esto, se puede crear un arquetipo hipotético para poder expresar los objetivos reales y que el usuario pretende lograr.

Para crear el prototipo es necesario decidir el contenido que le puede interesar al usuario, la información relevante, las motivaciones que tiene y en función de ello, llegaremos al contenido que hay que crear y diseñar. Como dijo Jakob Nielsen "*La usabilidad es un atributo de calidad que mide lo fácil que son de usar las interfaces de usuario. La palabra "usabilidad" también se refiere a métodos para mejorar la facilidad de uso durante el proceso de diseño.*" El prototipo consigue sustituir el documento tradicional que recoge la definición en cientos de páginas que nadie lee, consiguiendo una representación prácticamente exacta de la

funcionalidad que se quiere alcanzar y, además, poder utilizar a los usuarios como tester reales de usabilidad y estética antes de desarrollar. De este modo, se pueden detectar los problemas de uso que tendrá el producto de forma temprana, reduciendo costes y tiempo en el desarrollo.

Cuando se sitúa al usuario en el centro del proceso de definición, estamos hablando de *DCU*, es decir, *Diseño Centrado en el Usuario*, el cómo es el camino para alcanzar la usabilidad del producto, mientras que la usabilidad es el qué.

Antes de nada, cabe destacar una serie de términos que formarán parte del proceso de prototipado. En primer lugar, cabe destacar que el **diseño**, no es sólo una cuestión de opinión, es decir, conociendo los objetivos de un usuario el diseño se puede adaptar a la finalidad, independientemente de lo estético. También es importante saber que las **tareas** no son lo mismo que los objetivos, pues una tarea es un proceso intermedio que es necesario para cumplir con el objetivo. Por último, el **escenario** es la descripción narrativa donde una o más personas alcanzarán el objetivo propuesto. Para entenderlo mejor, vamos a ver un ejemplo visual en la Figura 21.

Figura 21. Elementos de prototipado

Una vez situada la persona con sus objetivos en un escenario con tareas a desempeñar, solo queda identificar los contenidos que podemos ofrecer con la ayuda de una estrategia de contenidos. La estrategia de contenidos es una serie de procesos que ayudan a estructurar, gestionar y organizar contenidos actuando en defensa del usuario y como vínculo de unión entre las necesidades del cliente y de la empresa.

A continuación, se realiza la modelización del contenido, pues como cita Zeldman *"El contenido precede al diseño. El diseño sin contenido, no es diseño, es decoración".* Es decir, para evitar diseñar algo que luego al incorporar el contenido, se aleja de la necesidad del cliente, es importante realizar el modelo previo del contenido.

En esta fase se tiene en cuenta:

- Tipo de contenido.
- Atributos que forman el tipo de contenido.
- Formatos (textos, imágenes, números, etc.).
- Extensión del contenido (caracteres).
- Frecuencia de actualización en caso de ser un elemento actualizable.
- Vocabulario controlado si se requiere.

Una vez definido todo esto, pasamos al prototipado, donde nos ayudará a representar de forma interactiva la funcionalidad conociendo el contenido que tendrá, las funcionalidades en función de las tareas del cliente dentro de un escenario específico y con unos objetivos establecidos. Una vez probado el prototipo por el usuario servirá de documento de definición y apoyo a las Historias de usuario y Product Backlog.

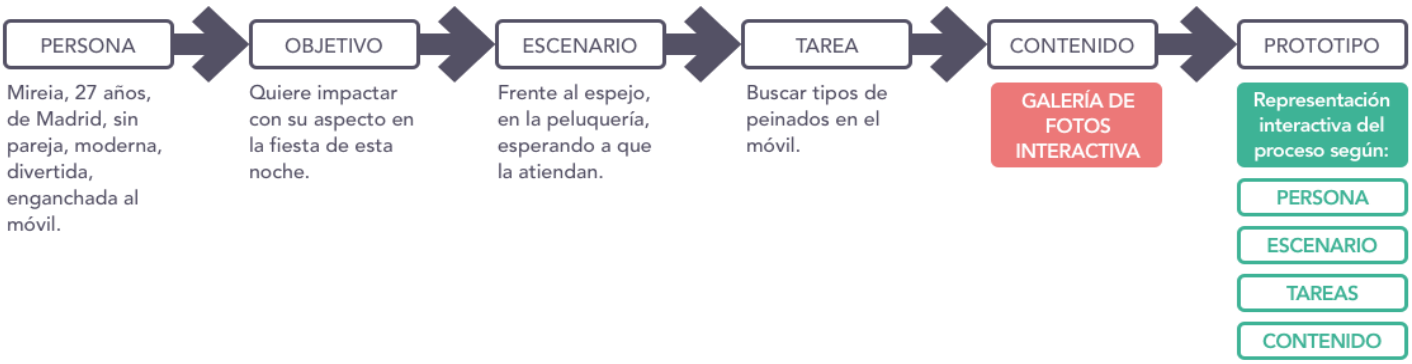


Figura 22. Fase de prototipado

La mejor forma de validar las soluciones de forma efectiva es mediante un prototipo interactivo, y así poder detectar en las pruebas de usabilidad los errores de la interfaz o comportamientos que no desea el cliente antes de comenzar con el desarrollo. Los prototipos se pueden clasificar en:

- **Sketch prototyping**. Este es el primer boceto del prototipo reflejado en papel y hecho a lápiz. Es la forma más rápida, económica y sencilla de esbozar las interfaces de la aplicación. El papel nos permite cometer errores, haciendo útil el proceso de exploración e ideación. En este proceso no se contempla la interacción como tal, pero es importante para discutir y conversar acerca de los bocetos e ir refinándolos en cada iteración.

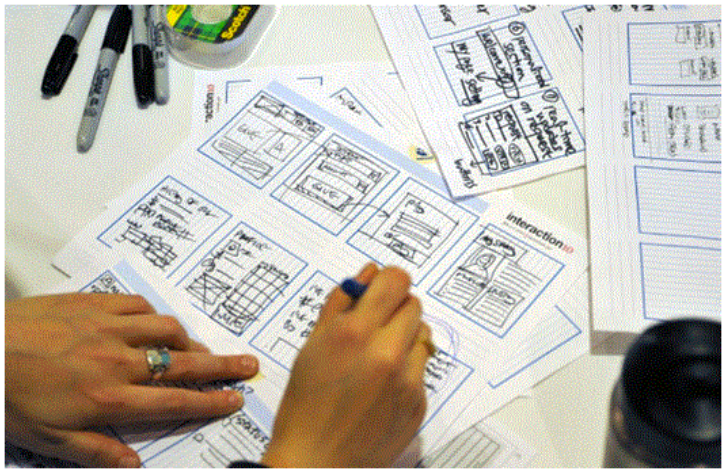


Figura 23. Sketch prototyping AISHA

- **Digital prototyping**. A continuación, se digitalizan los bocetos realizados. Para ello se puede hacer uso de herramientas específicas que permiten dotarlo de interactividad con efectos y movimientos. Aunque el diseño en papel sea más sencillo, estas herramientas digitales agilizan el proceso, ya que se puede copiar componentes ya diseñados o mantener un histórico de las versiones para poder retroceder en caso de equivocación.

  Para el proyecto que realizamos en la carrera, AISHA, realizamos los bocetos con la herramienta de prototipado y diseño de interfaces *Sketch*. La interacción de la que hablaba la realizamos con una extensión de la herramienta llamada *Marvel*, donde se podía crear movimiento y navegación entre los elementos del prototipo.

Figura 24. Sketch interfaz

3.2.3. Atomic Design

Desde el comienzo del desarrollo de proyectos software y web, se ha ido camuflando con distintos nombres, la componentización del diseño donde mediante la descomposición de problemas podemos dar con grandes soluciones. Este apartado estará nutrido del artículo de Brad Frost [11] acerca del Atomic Design, pues el concepto nace de aquí.

El concepto de componentización nace a finales de los 90 a medida que el diseño web iba evolucionando, pues surgió la necesidad de nuevos sistemas de diseño, como es Atomic Design, debido a la necesidad de adaptar la web a distintos tamaños de pantalla. Se trata de sistemas que van más allá del color o la tipografía, centrando su atención en tratar de entender en qué consisten las diferentes interfaces y cómo se pueden construir sistemas de diseño de una manera más metódica.

Al igual que la materia en su totalidad primeramente se compone de átomos, que se unen para formar moléculas, que a la vez se unen para formar organismos más complejos, etc. Del mismo modo, las interfaces están compuestas de elementos o componentes más pequeños. Podemos dividir esta estructura en cinco niveles:

1. **Átomos**. Es la unidad base, es decir, las mínimas unidades existentes que no se pueden dividir porque ya tienen sentido por sí mismas. Estos elementos podrían ser colores, tipografías, botones, imágenes, tablas, campos de formulario, etc.

   Para comprender mejor su utilidad vamos a contrastar la explicación con ejemplos ilustrados aplicados al proyecto AISHA donde se utilizó este concepto. En la Figura 25 podemos observar cómo nombrar cada variable con su respectivo código de color para poder utilizarlas en vez de reescribir código innecesario.

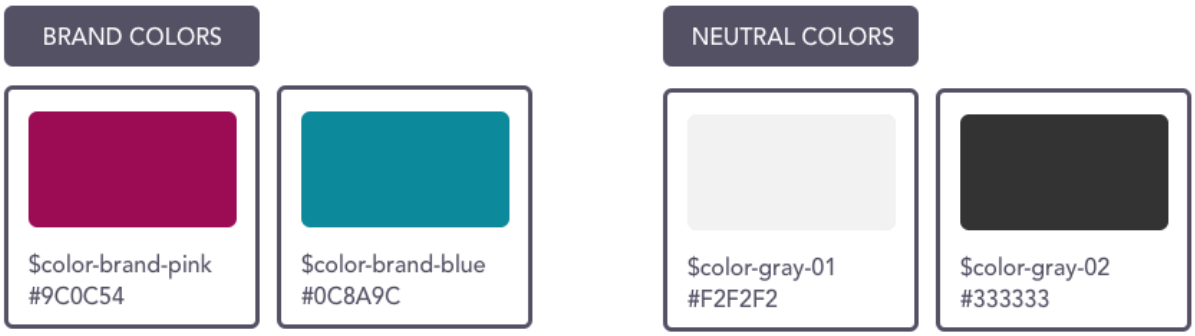


Figura 25. Ejemplo átomos con colores

También se puede aplicar a los botones, en los que se define un diseño y se crea un estilo predefinido para ese diseño de CSS y luego en el HTML basta con poner la clase correspondiente al botón que se desee como se puede observar en la Figura 26. De esta forma, conseguimos reducir y simplificar las líneas de código.

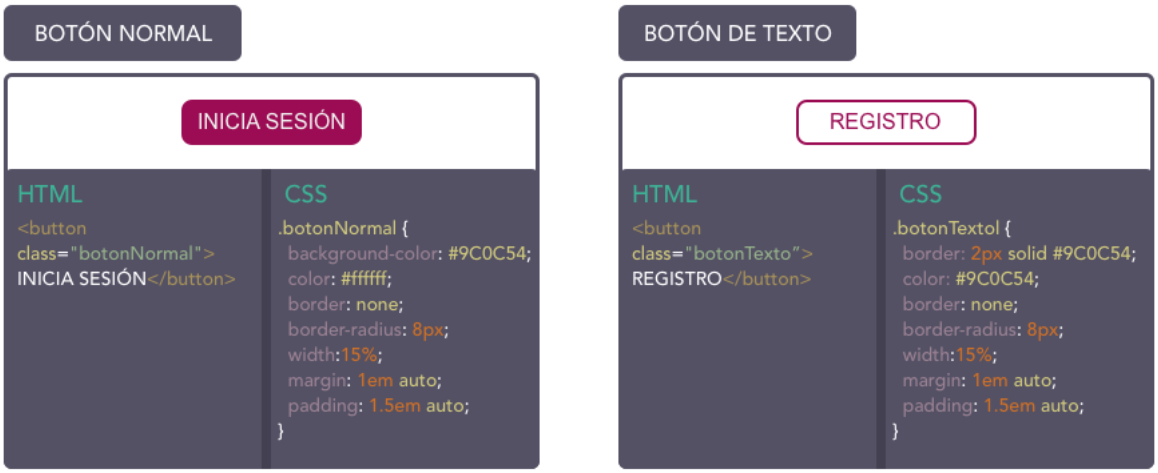


Figura 26. Ejemplo átomos con botones

Si componetizamos también los subelementos que componen un formulario, como pueden ser los campos de texto, los checkbox o las listas desplegables, cuando queramos utilizar estos elementos en cualquier formulario solo tenemos que aplicar las clases para reutilizar el mismo estilo en todos los formularios del proyecto.

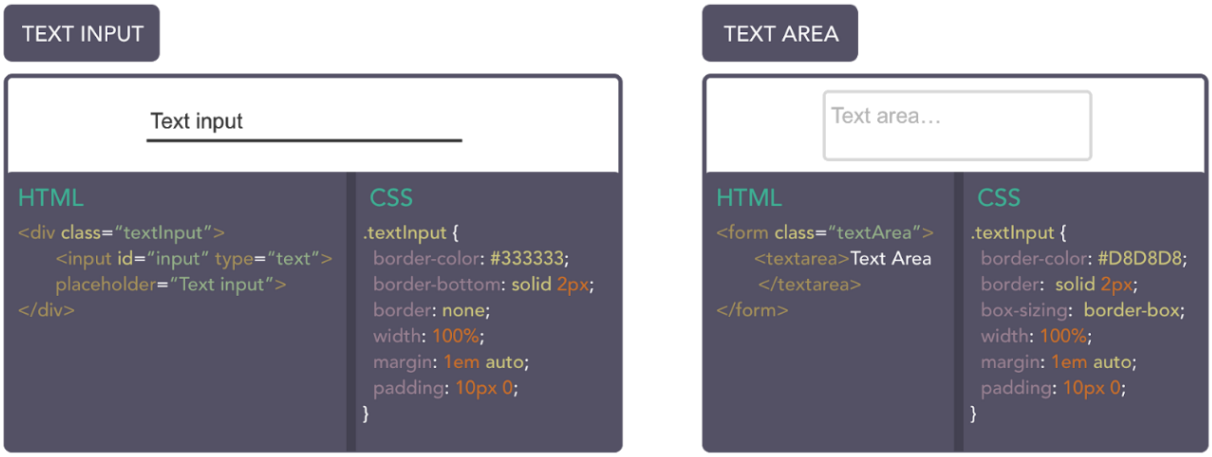


Figura 27. Ejemplo átomos con formularios

Estos pequeños átomos permiten dotar al sistema de una visión global del aspecto gráfico del proyecto, además de facilitarnos el trabajo en niveles cada vez más complejos y con posibilidad de reutilización cuando sea necesario.

2. **Moléculas**. Como hemos dicho al comienzo del apartado, las moléculas son las agrupaciones de los pequeños átomos, que se unen para dar forma a nuevos elementos cada vez más complejos, funcionando como una unidad. Esto corresponde a un menú de navegación, una tabla, etc.

   Para entenderlo mejor vamos a fijarnos en la Figura 28 donde se representa una estructura de moléculas siguiendo con los formularios. Por lados separados, el "label" y el "input" serían átomos, que juntándose forman moléculas. Entonces obtendremos un modelo más compuesto y reutilizable de formularios.

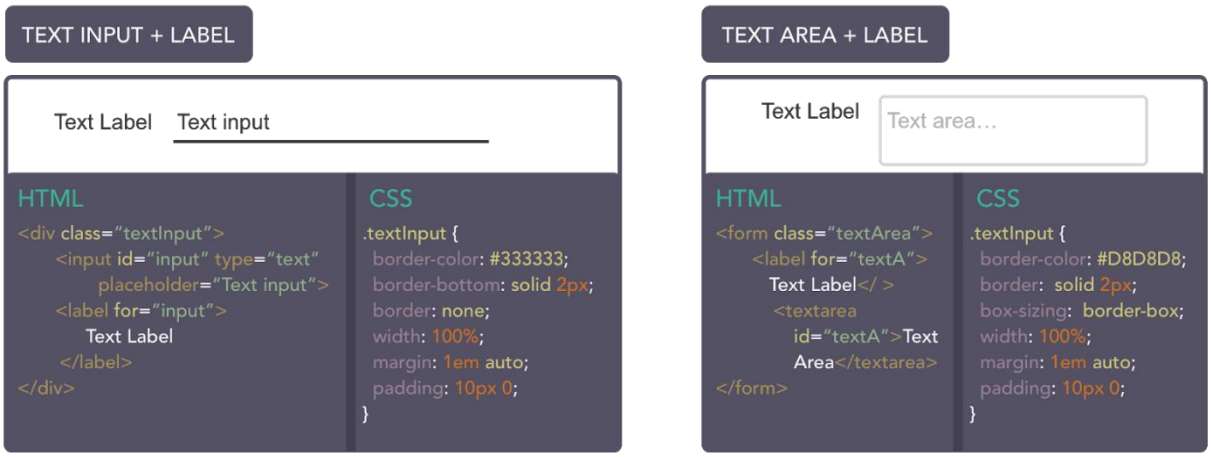


Figura 28. Ejemplo moléculas con formularios

3. **Organismos**. Estos elementos son conjuntos tanto de átomos como de moléculas que se forman para crear las diferentes secciones de la interfaz. Por ejemplo, las cabeceras serían un organismo formado por el logotipo como átomo, el menú y el buscador como molécula, etc.

Figura 29. Ejemplo de organismo con una cabecera

Un ejemplo sencillo de organismo podría ser el de una cabecera o “header”, la cual está compuesta, en el caso de la Figura 29 de dos moléculas, la imagen del logotipo y el menú de navegación. Por un lado, la primera molécula de la imagen del logotipo está conformada a su vez por dos átomos, la imagen en sí y el link que redirige a una página determinada. Por otro lado, la molécula del menú de navegación está compuesto por el menú y una lista de elementos que son las opciones con sus respectivos links.

4. **Plantillas**. Se trata de grupos de organismos que se colocan de tal forma que puede dar lugar a páginas. Es el momento en el que se empieza a dar coherencia al diseño junto con el contenido ya que es donde se crea una jerarquía y un orden de lectura a la hora de disponer los contenidos en la interfaz.

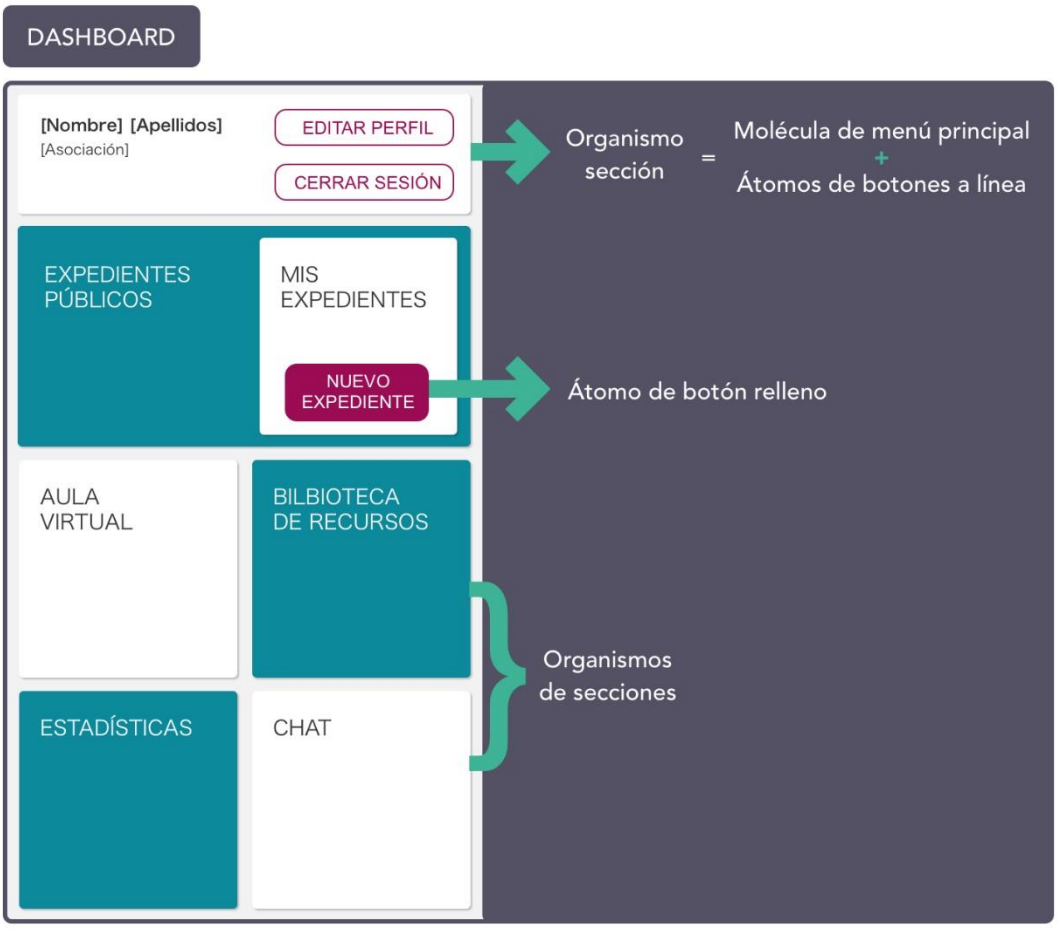


Figura 30. Ejemplo de plantilla de un menú dashboard

5. **Páginas**. Por último, tenemos las páginas, que reemplazan las estructuras o contenidos simulados con reales, con el fin de dar significado al producto final. Por lo tanto, son las más cercanas al diseño real, pudiendo simular diferentes estados del contenido, representando el paso de prueba y control de calidad del sistema, etc.

   En la Figura 31 podemos observar que una página finalmente está compuesta por diferentes elementos como puede ser el logotipo, los checkbox, botones, footer, campos de texto, imágenes, etc.

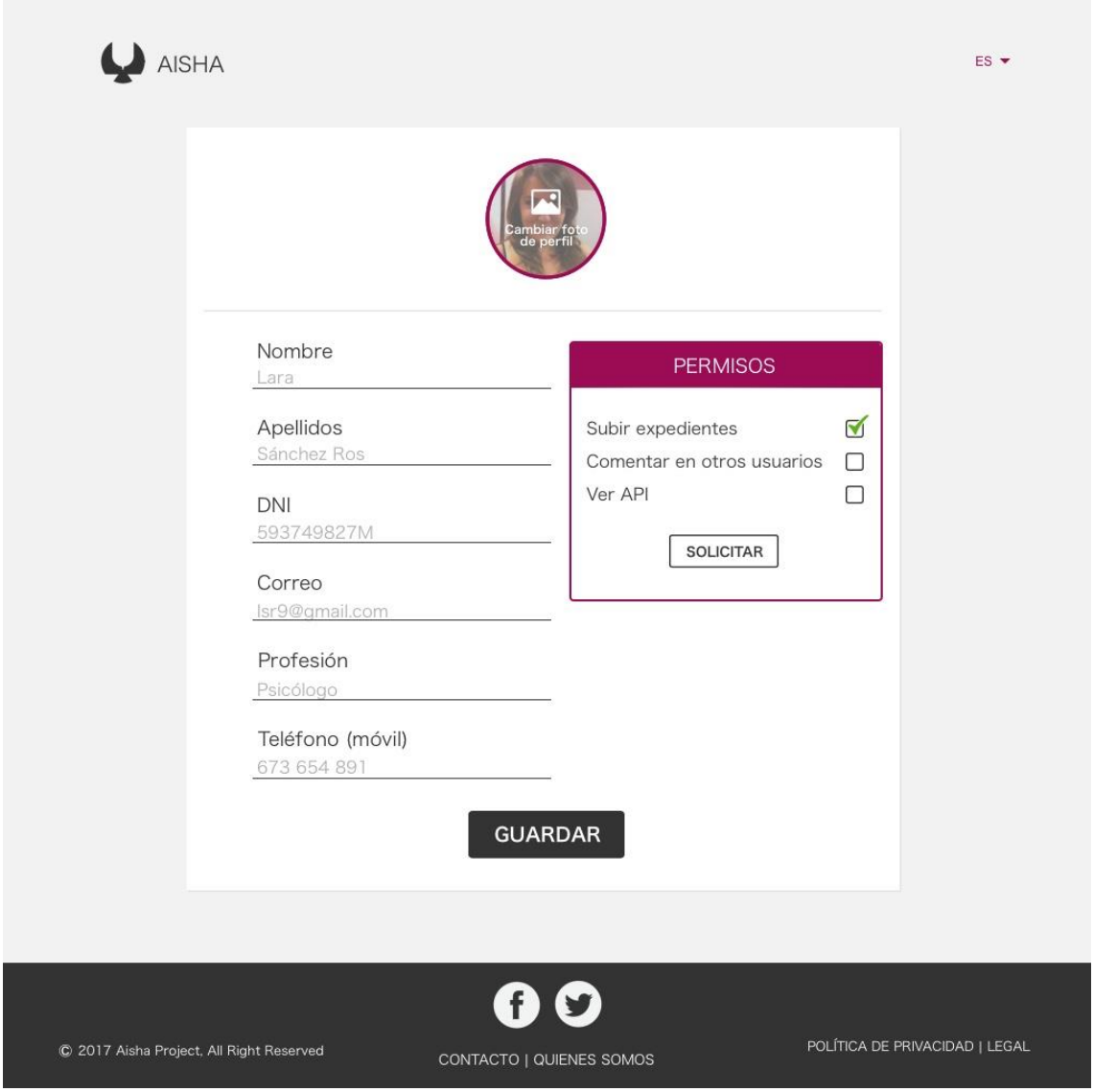


Figura 31. Ejemplo de página de Editar Perfil

Desde el punto de vista de la programación, en diferentes sistemas de implementación de código, fase que se llevará a cabo después del diseño previo, se hace uso de componentes reutilizables con sus respectivas propiedades heredadas. Gracias a la previa componentización del diseño se puede agilizar el proceso posterior de implementación de las interfaces.

Una vez ya analizados los diferentes niveles que podemos encontrar, vamos a ver las principales características que podemos extraer de Atomic Design:

- **Escalabilidad**. Se puede adaptar a cualquier dispositivo, pues uno de sus objetivos es realizar un diseño responsive o adaptativo a los diferentes tamaños de pantalla.

- **Actualización**. Es más sencillo realizar actualizaciones si es a nivel de diseño.

- **Mantenimiento**. Al estar tan dividido, realizar un pequeño cambio es mucho más sencillo.

- **Reutilización**. Es reutilizable y reaprovecharle, ya que cada elemento cada vez más pequeño es creado para adaptarse a diferentes contextos.

Este sistema sería más recomendable a proyectos web más complejos, ya que una web sencilla o con poco contenido puede utilizar soluciones alternativas más sencillas que dan buenos resultados como plantillas. Sin embargo, si hablamos de un proyecto web con mucho más contenido y un sistema más consistente que deba mantenerse en el tiempo, es más útil utilizar esta metodología para agilizar cambios y facilitar el mantenimiento del diseño y del sistema.

4. Antecedentes

Mi propuesta surge a raíz de realizar el proyecto AISHA en el que tuvimos que estar, tanto el equipo como el tutor del proyecto, en constante contacto con el cliente. En la primera reunión que tuvimos con la asociación con la que trabajaríamos, es decir, el cliente, fue una primera toma de contacto donde nos explicó de qué iba la asociación. Nos mostraron vídeos explicativos, nos dieron material, información y pudimos hablar con profesionales del campo de la mutilación genital femenina para conocer todo lo posible. Luego nos reunimos el equipo y realizamos un estudio con todo el material que se nos proporcionó para sacar las primeras ideas.

A partir de esas ideas realizamos diferentes tareas que repartimos entre los miembros del equipo. Yo me encargaba del front-end y diseño. Todo esto se recogió en un documento funcional en forma de texto, pero sin contemplar nada del diseño.

La siguiente fase fue realizar absolutamente todos los wireframe de las interfaces que compondrían el proyecto y para ello utilizamos la herramienta MockFlow [12].

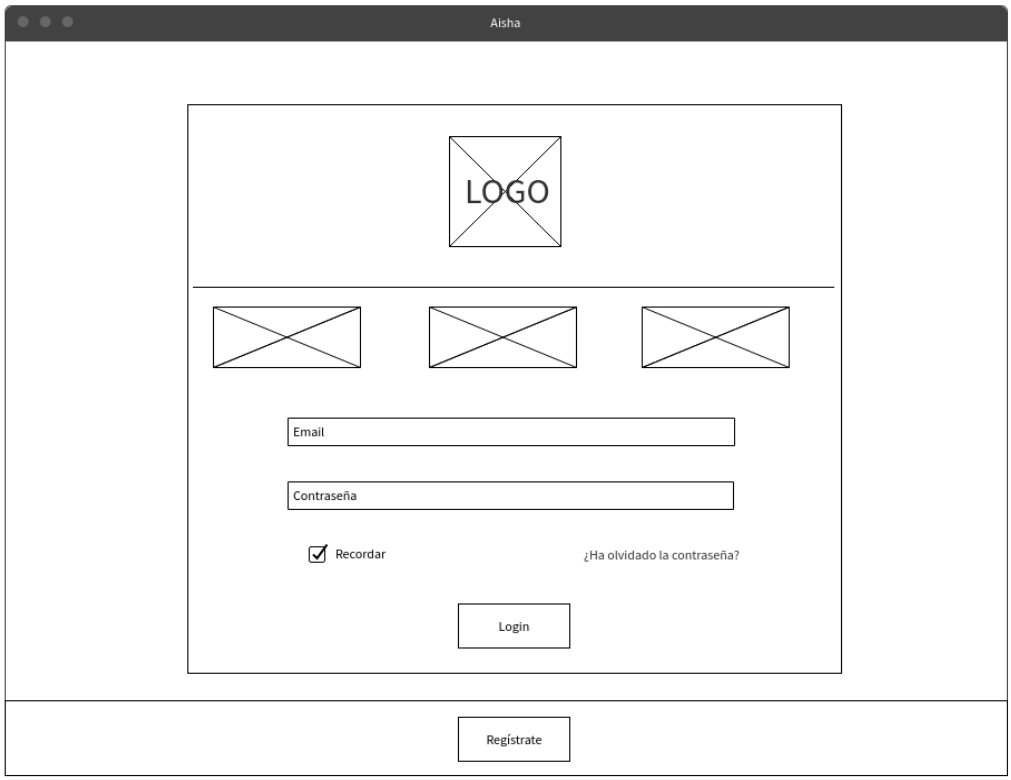


Figura 32. Ejemplo de Login con MockFlow

Una vez realizados los mockups sin apenas revisarlos, los empezamos a programar. El hecho de no haber establecido ningún criterio de diseño ni guía de estilos generó en el equipo desconcierto y por culpa de la desorientación tardamos el triple en implementar todas las interfaces estimadas ya que íbamos eligiendo los colores y las formas según la marcha, discutiendo en el momento con todo el equipo cómo hacer todos los detalles que podríamos haber establecido desde un principio.

Es entonces cuando decidimos volver a reunirnos con el cliente para enseñarles prácticamente toda la web que llevábamos hasta el momento, a falta de funcionalidades más específicas, y nos dimos de frente con la exigencia real de un proyecto real. El cliente nos comentó mil funcionalidades que quería modificar, colores que no veía muy acorde con su asociación y otras mil funcionalidades que quería incluir. No dábamos crédito a lo que estaba sucediendo, tanto trabajo tirado por el suelo.

Cansada de rehacer constantemente código y diseño, opté por realizar las interfaces de nuevo, partiendo de la principal como base para toma de decisiones de elementos básicos. Pero esta vez, interfaces a color, con elementos, tipografías, formularios tal cual serían implementados más tarde con la ayuda de la herramienta Sketch.

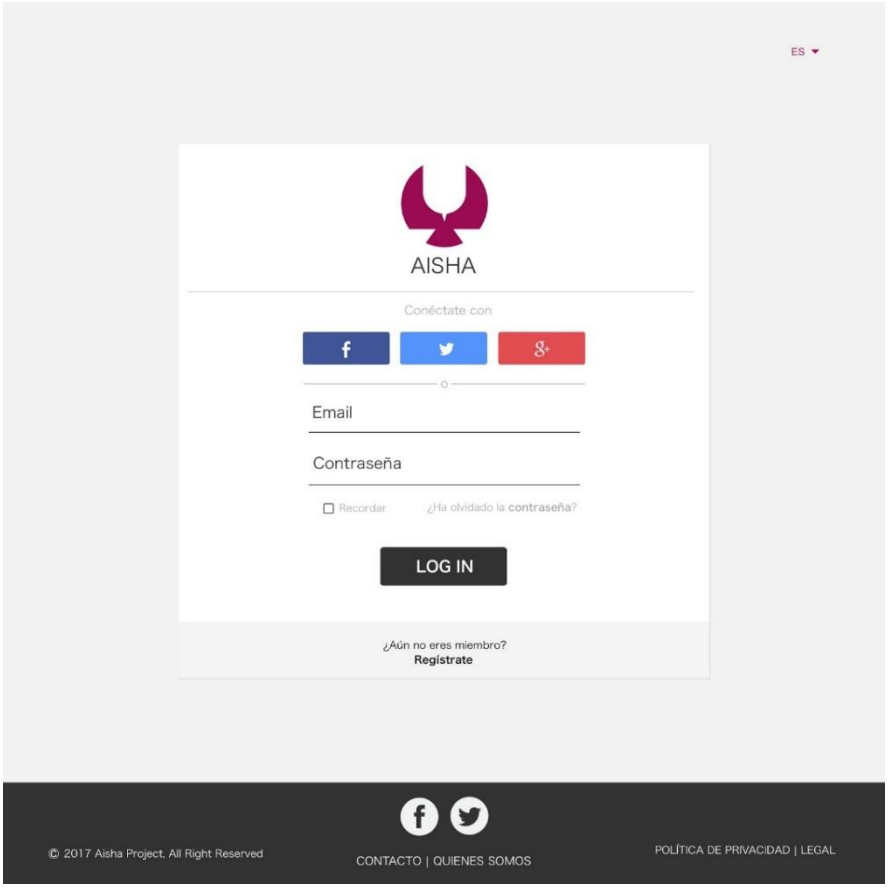


Figura 33. Ejemplo de Login con Sketch

Todo el tiempo invertido en el desarrollo de interfaces se redujo a más de la mitad gracias a esta forma de trabajo. Entonces comprendí lo importante que era establecer una base sólida, verificada y comprobada por el cliente, antes de llevar a cabo el desarrollo. Toda nuestra forma de trabajo volcó y dio un nuevo giro, donde ganamos productividad, tiempo y eficacia. Conseguimos llegar a todos los plazos de entrega previstos y trabajar en un ambiente más productivo en el equipo.

5. Metodología

5.1. Herramientas de metodología

Partiendo de querer combinar trabajo, estudios y ocio, opté por la mejor forma de organizarme el tiempo para poder desarrollar el TFG. Para ello hice con Excel una planificación dividiendo cada hoja de Excel en los meses donde llevaría a cabo el desarrollo y las horas estimadas que dedicaría cada día, haciendo constantemente cambios para llegar a las horas deseadas. Si no realizaba las horas que tenía estimadas, sumaba esas horas a las previstas de la siguiente semana, así hasta conseguir el objetivo. De esta forma, he podido visualizar de un simple vistazo, el calendario mensual teniendo marcados eventos importantes, festivos y el desarrollo hecho cada día.

	OCTUBRE										
	LUNES	MARTES	MIERCOLES	JUEVES	VIERNES	SABADO	DOMINGO	HORAS			
	1	2	3	4	5	6	7	SEMANALES	PREVISIÓN	RESULTADO	HORAS
DESCRIPCIÓN	Citas	Tutoría	Motivación		Introducción	Introducción Planificación					
DURACIÓN	0:30	1:00	1:00		1:00	5:30	6:00	15:0	15:00	OK!	
	LUNES	MARTES	MIERCOLES	JUEVES	VIERNES	SABADO	DOMINGO				
	8	9	10	11	12	13	14				
DESCRIPCIÓN											
DURACIÓN								0:0	15:00	FALTAN	15:0
	LUNES	MARTES	MIERCOLES	JUEVES	VIERNES	SABADO	DOMINGO				
	15	16	17	18	19	20	21				
DESCRIPCIÓN											
DURACIÓN								0:0	15:00	FALTAN	15:0
	LUNES	MARTES	MIERCOLES	JUEVES	VIERNES	SABADO	DOMINGO				
	22	23	24	25	26	27	28				
DESCRIPCIÓN											
DURACIÓN								0:0	15:00	FALTAN	15:0
	LUNES	MARTES	MIERCOLES								
	29	30	31								
DESCRIPCIÓN											
DURACIÓN								0:0	15:00	FALTAN	15:0
TOTAL								15:0	75:0	60:0	
										FALTAN	

	PREVISTO M	RESULTADO M	RESULTADO		PREVISTO T	RESULTADO T	RESULTADO
	75:0	15:0	FALTAN		300:0	27:0	FALTAN
			60:0				273:0

Figura 34. Planificación Excel del TFG por horas semanales

Para organizar los apartados del documento, me creé un tablero en Trello basado en metodologías ágiles, en concreto en *Scrumban* (Scrum y Kanban) donde clasificaba en varias columnas el proceso mediante tarjetas. Para dar prioridad a unas tarjetas más que a otras, las colocaba al comienzo de la columna o pila en cada parte. Las columnas las personalizaba yo y las diferenciaba en:

- **Backlog**. Es la columna donde primero puse todas las tarjetas de cada apartado a realizar, cada esquema que diseñar, cada tabla, etc.
- **To-Do**. Era la siguiente columna donde movía las tarjetas que quería hacer en un periodo de tiempo, como una iteración, en relación con la planificación del Excel.
- **En proceso**. Cuando comenzaba una tarea movía la tarjeta a esta columna para tener siempre localizado mi trabajo y no perderme.
- **Revisar**. Cuando releía apartados y encontraba incoherencias o cosas a modificar, movía las tarjetas a esta columna. Además, señalaba los apartados en el documento para localizarlos luego más fácilmente. La columna la iba vaciando conforme avanzaba en el proyecto, según era conveniente.
- **Corregir**. En esta columna se situaban las tareas terminadas o modificadas que quería que mi tutor me tenía que revisar y verificar.
- **Hecho**. Finalmente, las tareas corregidas las pasaba a esta columna. Es decir, las tarjetas ya terminadas y, además, corregidas por el tutor.

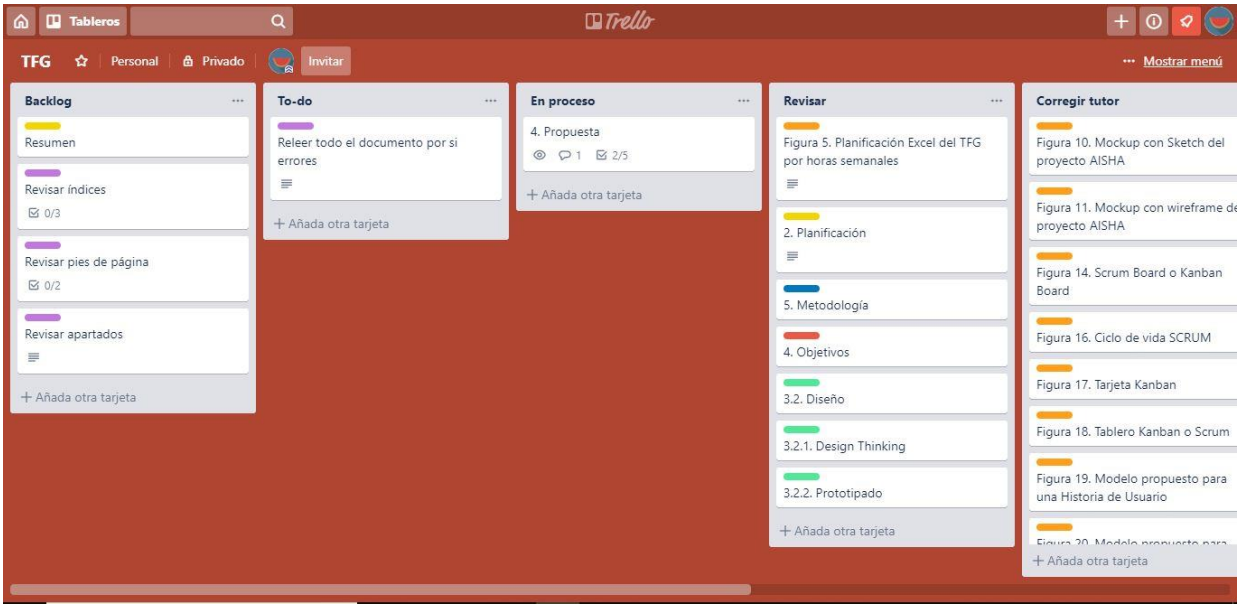


Figura 35. Planificación Trello

5.2. Herramientas software

Mediante el uso de herramientas software he podido realizar todos los esquemas, llevar un recuento de las horas dedicadas y documentarlo todo:

- **Word**. Es un programa informático orientado al procesamiento de textos donde he desarrollado todo el TFG.

- **Excel**. Programa para crear y trabajar con hojas de cálculo donde he realizado la planificación de las horas y tareas diarias.

- **Toggl**. Herramienta que cronometra el tiempo exacto que se dedica a cada tarea [13]. Gracias a esta herramienta he podido llevar un seguimiento del tiempo que dedicaba a cada tarea y estimaba la planificación de manera semanal para poder realizar todo en su plazo estimado.

- **Sketch**. Es un programa de diseño vectorial y prototipado con opciones muy intuitivas que dan flexibilidad y ligereza a la hora de realizar esquemas y prototipos. Mediante este programa he realizado numerosos esquemas y diseños de interfaces.

- **Adobe Illustrator**. Es un editor de gráficos vectoriales donde he realizado esquemas que requerían de más precisión y la iconografía utilizada en el proyecto.

5.3. Recopilación de información

Antes de todo el desarrollo llevado a cabo, desde el primer apartado, se realizó un proceso de investigación de todos los temas tratados para recoger la suficiente cantidad de información consistente y fiable que daría soporte más tarde a mi propuesta. La información ha sido extraída de todo tipo de fuentes, ya sean libros, artículos o publicaciones de autores con experiencia en el campo de las metodologías y los desarrollos web, etc. Todas las bibliografías, webgrafías y referencias están documentadas en el apartado de Referencias. También, tuve la suerte de contar con el apoyo de mi responsable de proyecto en la empresa en la que estoy trabajando actualmente, ya que es Scrum Master en desarrollo de proyectos web. Por lo que ha ido aconsejándome y revisando cada apartado del TFG.

6. Objetivos

El objetivo del proyecto es incorporar adecuadamente la fase de diseño en el desarrollo de proyectos web. Gracias a esto, el proyecto estará dotando de una funcionalidad y un diseño en sinergia, reduciendo costes, aumentando productividad y eficiencia. Conforme mejor conozcamos al cliente o usuario, los procesos y las funcionalidades, mejor entenderemos cómo puede afectar al diseño y así anticiparnos a las necesidades presentes y futuras del usuario.

Para conseguir esto, antes necesitamos definir unos objetivos más concisos que ayudarán a conseguir el objetivo principal.

- **Planificación**. Antes de comenzar es importante establecer un orden, un esquema auxiliar que nos ayude a orientar la idea, analizando lo que debemos estudiar previamente y cómo compaginar cada punto.

- **Metodología**. Una vez establecido el orden, podemos comenzar el desarrollo orientado por una metodología que nos ayudará con los plazos de finalización de apartados en el tiempo.

- **Estado del arte**. Antes de comenzar con el desarrollo de la idea, es necesario formarnos más profundamente en el contexto donde se desarrolla. Desde metodologías de gestión de proyectos a herramientas que ayudan a incorporar el diseño en el desarrollo.

- **Identificación de problemas**. Durante el estudio del punto anterior, se identifican y analizan los puntos exactos donde se encuentra el problema que propongo solucionar, es decir, las partes donde no se cuenta con la fase del diseño en las que propongo su incorporación, así como ventajas y desventajas de ello.

- **Propuestas de soluciones**. Una vez identificado los problemas, propongo una serie de soluciones en cada punto acorde con la necesidad y la naturaleza de la metodología o herramienta analizada.

7. Propuesta

El concepto de diseño aplicado a web no es realizar lo que uno mismo considera o le apetece aplicar. El diseño nace a partir de un análisis, creando un equilibrio entre estética y funcionalidad. El valor del diseño actualmente coge cada vez más y más peso pues no solo se trata de usabilidad y accesibilidad, velocidad o programación, va más allá. Una página web atractiva genera sensaciones y transmite emociones como puede ser confianza o rechazo.

Pero, para desarrollar un proyecto web en poco tiempo y, además, que sea atractivo visualmente, usable y accesible es importante hacer uso de metodologías. Una vez realizado el estudio de las metodologías existentes podemos corroborar que no existe ninguna que contemple el diseño en sus etapas, es decir, el diseño no es ágil en el desarrollo de un proyecto web.

La necesidad continua de realizar modificaciones, añadidos o eliminados en el proyecto debido a su naturaleza, requiere de un método iterativo en el desarrollo. Por lo tanto, mi propuesta es formalizar todo ese proceso que llevábamos a cabo con AISHA, como explicaba en el apartado 3.3. Antecedentes, en una metodología ágil donde la fase de diseño también lo sea. Después de realizar todo el estudio del Estado del Arte podemos observar que, por una parte, Waterfall es una metodología que contempla la fase de diseño, pero no es ágil, y, por otra parte, las metodologías que sí son ágiles como Scrum, Kanban o Scrumban, no contemplan el diseño ya que son tareas implícitas en las historias.

Aunque Waterfall contenga esta fase de diseño, el proceso más adecuado para llevar a cabo ésta práctica considero que es el de la Metodología Ágil ya que con una Metodología Tradicional estamos perjudicando al flujo cambiante de un proyecto. En concreto, Scrumban es de las que mejor encaja en mi idea ya que complementa Scrum, que proporciona el orden necesario aceptando los cambios naturales de un proyecto de la forma más eficaz, con Kanban, que mediante la gestión visual del desarrollo facilita el proceso y ayuda a abordar el trabajo y los colapsos que puedan surgir. Junto con Atomic Design para dividir y dar prioridad a las tareas que realizar. Por lo tanto, me basaré en Scrumban y Atomic Design para plantear mi idea.

La idea de las iteraciones o entregas parciales encaja mejor en la parte creativa ya que se orienta a la componentización incremental de elementos y no a su totalidad en la fase inicial, como ocurre en las metodologías tradicionales, donde no sería posible realizar modificaciones una vez establecido el diseño. Pero en su contra, trabajar de esta manera puede generar productos incoherentes o descoordinados en diseño, por lo que debe realizarse de manera cuidadosa y con mucha comunicación entre los miembros del equipo para ponerse de acuerdo en cada decisión. Además, debido al proceso cambiante, sería interesante componentizar los elementos del proyecto como propone Atomic Design para cualquier posible modificación por parte del cliente, poder realizarla con más facilidad y de este modo, evitar que afecte a todo lo que ya está producido.

Para agilizar el diseño, se necesita de entregas parciales de resultados tempranos donde el cliente pueda verificar y dar por finalizado las distintas fases de diseño. De esta forma se reducirán los cambios a realizar y potenciar la productividad. Por otra parte, Design Thinking aporta un gran valor a esta idea, ya que conocer lo que realmente necesita el cliente, cada mínimo detalle ayuda en cualquier parte de diseño que se pueda crear para él. Por ello, es el método que mejor encajaría en el proceso creativo defendido hasta ahora.

La dinámica que defiendo es la de realizar prototipos en base a las necesidades y una vez dado el visto bueno, pasar a la implementación. La metodología que propongo consta de dos iteraciones. La *primera fase* será fija ya que debe realizarse sí o sí pues es la parte más importante de la metodología donde se recoge la información y se analiza, si esa parte no queda clara desde un principio, no será un desarrollo consistente y organizado. Después, la *segunda fase* es de carácter variable, ya que su duración va en función de una serie de factores como son la estimación del equipo, la demanda del cliente, el presupuesto y demás.

Para entender mejor este proceso se pueden observar cada una de las actividades y tareas en el siguiente esquema, que después desglosaremos con más detalle para cada uno de los puntos:

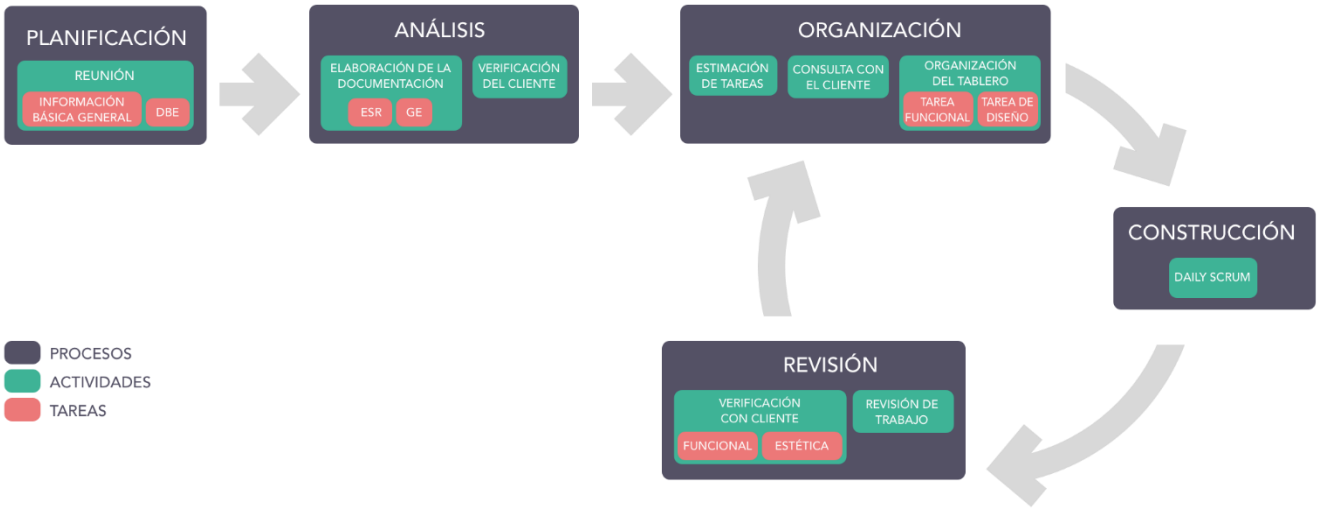


Figura 36. Ciclo de vida propuesto

Para llevar a cabo esta práctica, vamos a analizar los tres distintos elementos que podemos encontrar en el esquema del a Figura 36. En el primer nivel, de color azul oscuro, tenemos los procesos, los cuales comprenden un conjunto de actividades como segundo nivel, en color verde, relacionadas entre sí. Estas actividades son un conjunto de tareas en el tercer y último nivel, que se encargan de producir, es decir, la tarea es la labor con la que se produce un recurso.

7.1. Planificación

En el primer proceso se realizará una reunión con el cliente en la que básicamente, se recoja toda la información posible de su necesidad y entorno de trabajo como sucede en Scrum, en concreto haría referencia al *Scrum Planning*. Esta reunión duraría mínimo 2 horas, pero con un margen de dos días de ampliación para posibles improvistos y dudas, ya que teóricamente, una vez realizada la reunión con el cliente, no se deberían tener más dudas que las resueltas en dicha reunión, pero si lo llevamos a la práctica, la realidad es que después de la reunión con el cliente, surgen numerosas dudas e incoherencias que deben ser resueltas antes de comenzar el desarrollo. Por lo tanto, el primer día estaría enfocado a la reunión y los otros dos días el equipo se dedicaría a pulir todo y resolver esas dudas o incoherencias que puedan surgir. De la semana que dura esta primera iteración, tres días los ocupamos con la primera etapa de planificación.

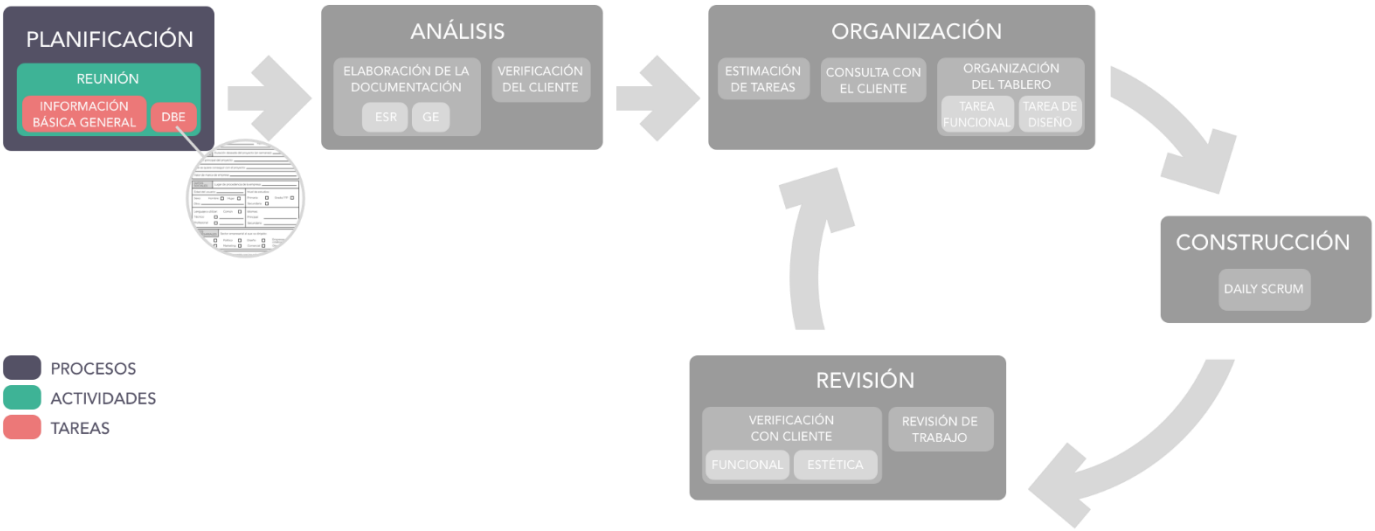


Figura 37. Ciclo de vida con planificación

Básicamente, en esta reunión es donde el cliente especifica las funcionalidades que quiere y se resuelven todas las dudas que puedan surgir. Esto forma parte de la primera tarea de "Información básica general", donde cuanta más información recojamos, mejor vamos a entender al cliente, y, por lo tanto, más vamos a conseguir cubrir sus necesidades. Por eso es la fase crucial, donde lo más importante es documentarse de forma correcta para comenzar el desarrollo adecuadamente.

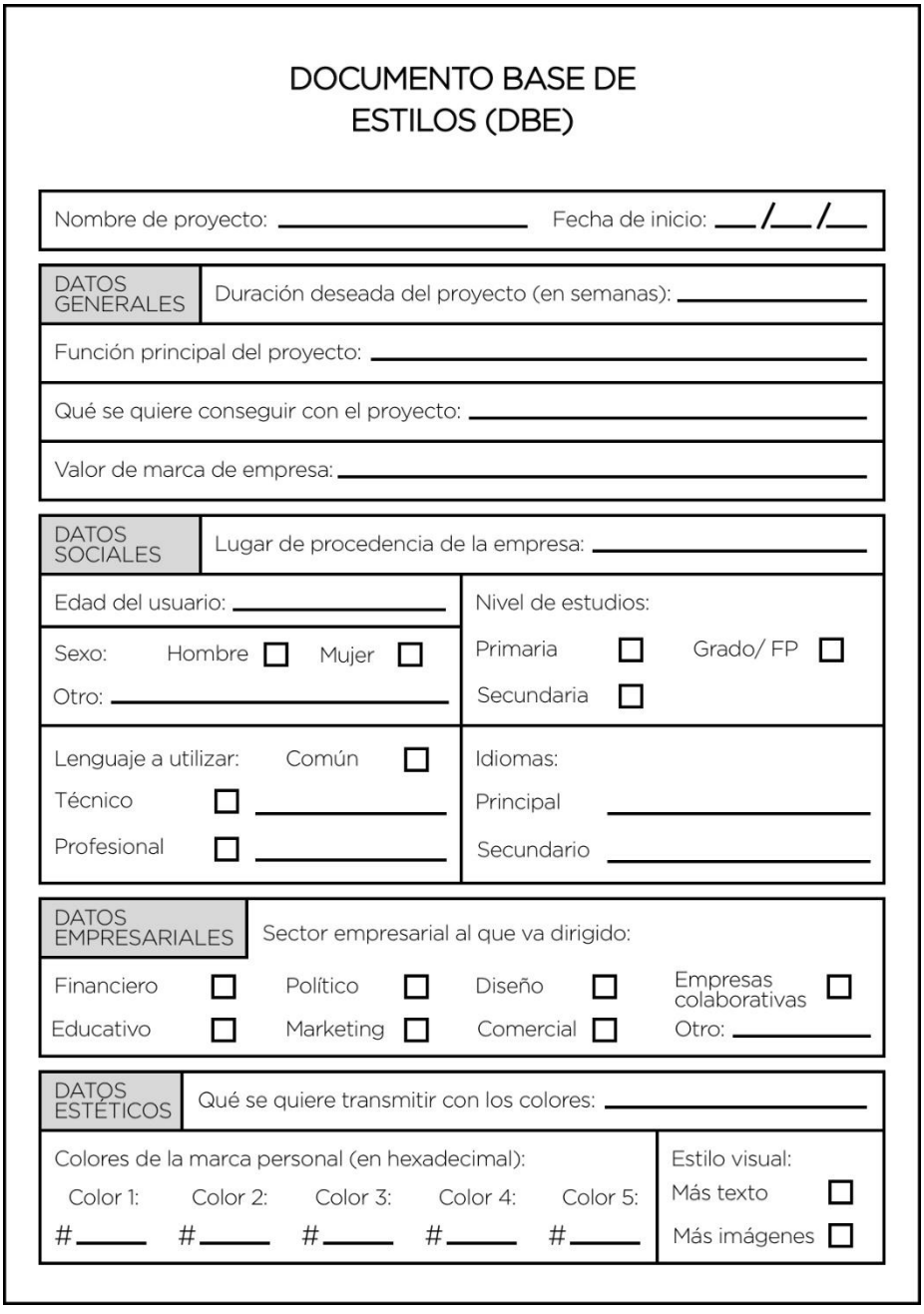
Como punto diferenciador a Scrum, incluiría en mi propuesta el uso de una plantilla de recogida de información el cual he denominado *Documento de Base de Estilos* (*DBE*), recogiendo los datos mínimos que ayudará más tarde en el siguiente proceso a no estar desorientado con ciertos aspectos estéticos. En la Figura 38 podemos ver un ejemplo que he realizado de una plantilla, recogiendo:

- **Datos generales**. Estos datos ayudarán a reflejar en el diseño el valor de la marca con una estimación previa con la que partir, aunque luego se realice la estimación real con las tareas reales que se llevarán a cabo.

- **Datos sociales**. El público al que va dirigido un producto puede variar mucho en su estética, pues si va dirigido a niños, se hará uso de más colores, más saturados y con una estética más infantil. Pero en cambio, si va dirigido a un público más adulto, se hará uso de colores más neutros y un estilo más minimalista.

- **Datos empresariales**. El sector financiero al que se dirige el proyecto también ofrece muchas pistas al diseño acerca de qué elementos utilizar o no, colores a utilizar en determinados sitios, etc.

- **Datos estéticos**. Por último, los datos estéticos con los colores de la marca personal son los que condicionan el estilo de todo el proyecto, pues si el logotipo de una empresa y su marca tiene como color principal el naranja, si se crea una web con el mismo color naranja, al usuario le transmitirá confianza y enseguida relaciona el producto con su empresa, o bien sea para marketing.



DOCUMENTO BASE DE ESTILOS (DBE)

Nombre de proyecto: _____ Fecha de inicio: __/__/__

DATOS GENERALES
Duración deseada del proyecto (en semanas): _____
Función principal del proyecto: _____
Qué se quiere conseguir con el proyecto: _____
Valor de marca de empresa: _____

DATOS SOCIALES
Lugar de procedencia de la empresa: _____
Edad del usuario: _____
Nivel de estudios:
Sexo: Hombre ☐ Mujer ☐
Otro: _____
Primaria ☐ Grado/ FP ☐
Secundaria ☐
Lenguaje a utilizar: Común ☐
Técnico ☐ _____
Profesional ☐ _____
Idiomas:
Principal _____
Secundario _____

DATOS EMPRESARIALES
Sector empresarial al que va dirigido:
Financiero ☐ Político ☐ Diseño ☐ Empresas colaborativas ☐
Educativo ☐ Marketing ☐ Comercial ☐ Otro: _____

DATOS ESTÉTICOS
Qué se quiere transmitir con los colores: _____
Colores de la marca personal (en hexadecimal):
Color 1: # _____ Color 2: # _____ Color 3: # _____ Color 4: # _____ Color 5: # _____
Estilo visual:
Más texto ☐
Más imágenes ☐

Figura 38. Documento base de estilos

7.2. Análisis

Continuando con la *primera fase*, nos encontramos con el siguiente proceso, el del análisis. En este proceso es necesario definir correctamente el problema para establecer un equilibrio en el desarrollo y llegar a la solución adecuada. A su vez, se divide de dos actividades que vamos a desarrollar en detalle a continuación.

7.2.1. Elaboración de la documentación.

Una vez recogida toda la información, es de gran importancia documentar por un lado la parte de diseño y por otro la de funcionalidad, y realizar un análisis y estudio acerca de lo que se pide o cómo es el cliente. Esta documentación sería mínima, como establece el principio de Agile, que luego se irá complementando conforme se desarrolle.

- **Documento funcional**. El documento funcional recogerá la información técnica en función de las necesidades del cliente, independiente del diseño. Este documento se utiliza habitualmente en desarrollo, conocido como *Documento de Especificación de Requerimientos de Software* (*ERS* en español y *SRS* en inglés, *Software Requirement Specification*). En este documento se recogerán datos como: descripciones de datos, de operadores en pantallas, de los flujos de trabajo, reportes del sistema, tipos de usuario que podrán o no ingresar datos en el sistema, cómo el sistema cumple los reglamentos, etc.

- **Documento de guía de estilo**. Para documentar el diseño, mediante el uso de la información recogida en la plantilla en el proceso anterior, propongo la elaboración de una guía que servirá de ayuda a lo largo del desarrollo, la cual he denominado *Guía de Estilos (GE)*. Ésta recogerá la **información básica del diseño**, como son los códigos de color, las tipografías para determinados textos, el estilo de los formularios, etc. Además, se realiza un estudio de las pantallas que se necesitan y se establecen los **mockup** correspondientes a línea, que determinará los elementos existentes.

  Para poder trabajar de forma independiente, tiene que establecerse una conexión entre las tareas funcionales y de diseño. Este nexo de conexión son las clases o identificadores, que se asignan a los elementos en los mockup y una vez verificados, se define el estilo de las clases o identificadores correspondientes.

En resumen, en esta fase lo que se hace es diseñar un flujo técnico donde se vea claramente cómo será el movimiento entre las distintas pantallas y las funcionalidades generales que se trabajarán.

Figura 39. Documento de guía de estilos

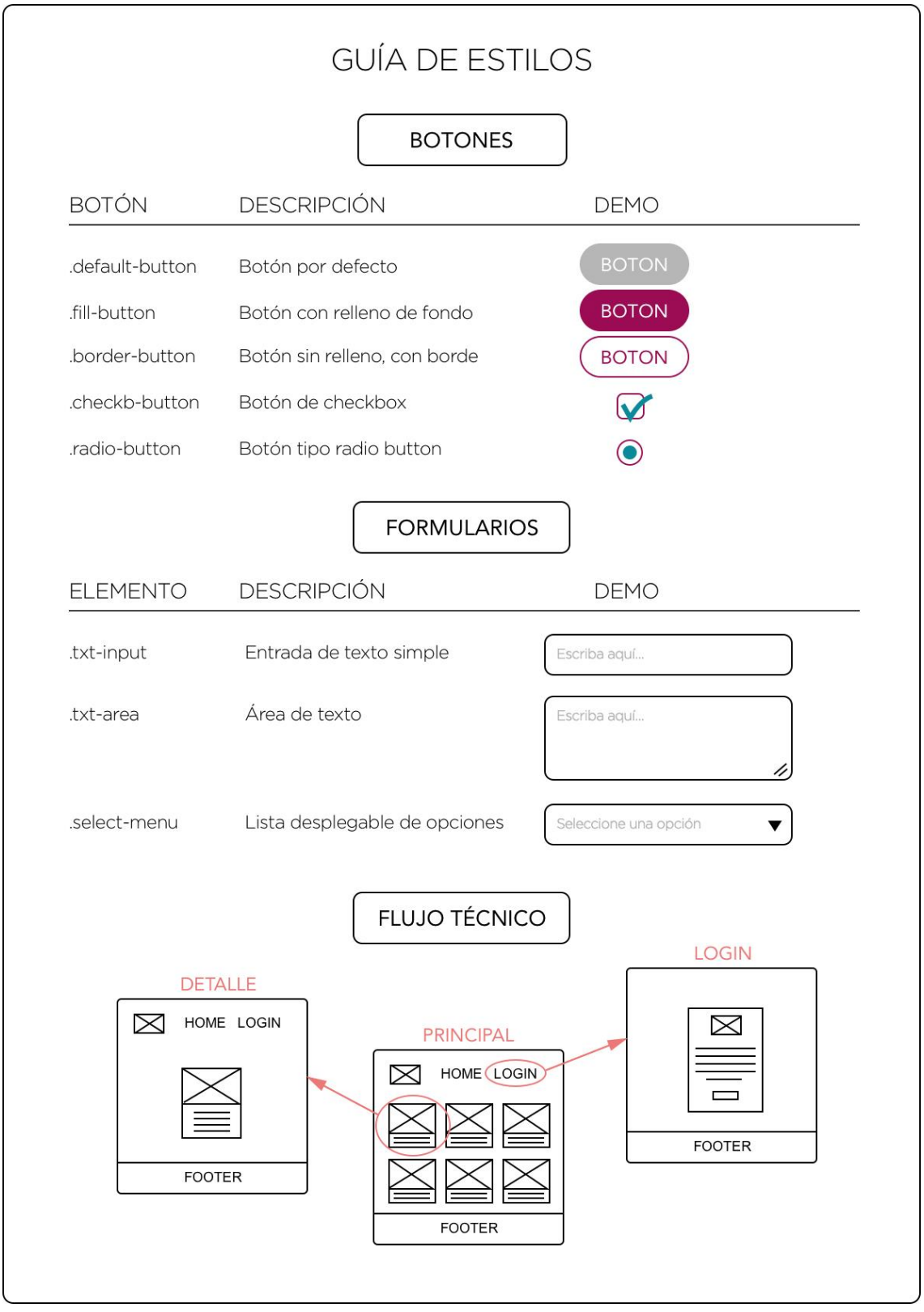


Figura 40. Documento de guía de estilos 2

Dentro del documento *GE*, se situará un apartado de *wireframe* de las pantallas del proyecto. Esto significa, bocetos de las interfaces, sin nivel de detalle, que permitan a los desarrolladores y diseñadores del equipo, identificar cada elemento existente en el proyecto mediante un nombre específico y dónde situarlo. Este identificador se podría realizar concatenando, por ejemplo, el tipo de elemento junto con un nombre que lo identifique, es decir, el nombre de la interfaz a la que pertenezca. Para comprenderlo mejor vamos a observar la Figura 41:

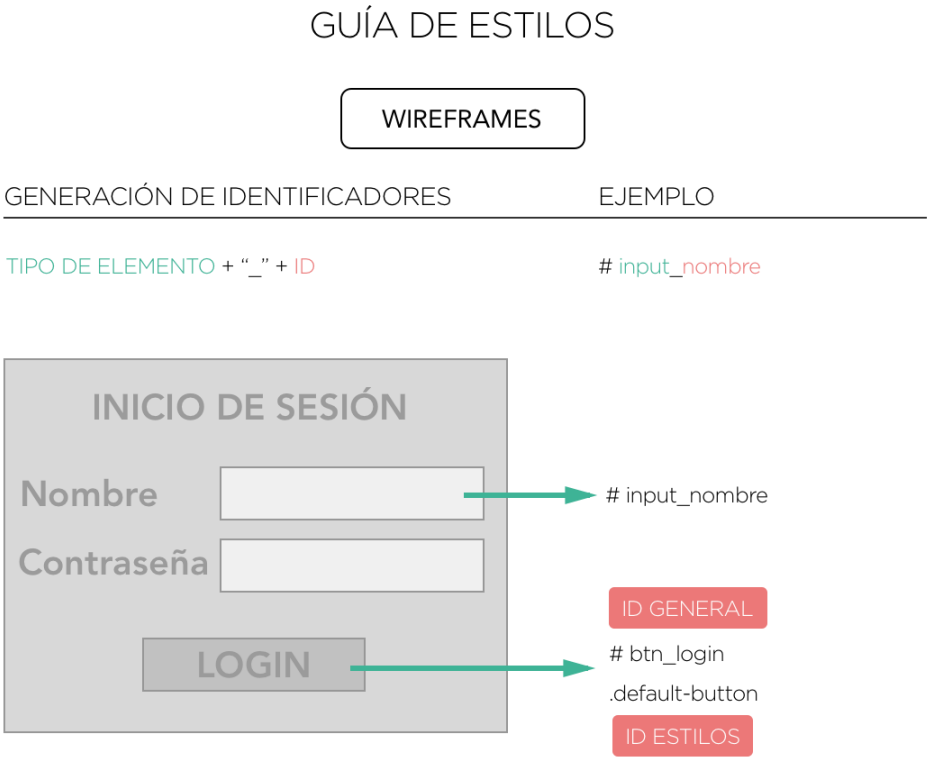


Figura 41. Apartado de Wireframe de la guía de estilos

De este modo, si cada elemento tiene un identificador, el desarrollador y el diseñador podrán acceder durante el desarrollo, siempre que lo necesiten, al documento para localizar los elementos pertinentes, sin tener que estar constantemente en contacto y así, trabajar de una forma más ágil y totalmente desacoplados.

7.2.2. Verificación del cliente

Antes de proceder con la organización de las tareas definidas, se reúne con el cliente, tanto el equipo como el responsable, y se verifica la guía de estilos y de funcionalidades. Esta actividad correspondería con el *Scrum Review*, de duración de 1 hora, que es totalmente necesario para agilizar más tarde todas las tareas que se realicen, pues el estilo y la funcionalidad base estará bien definida y consistente.

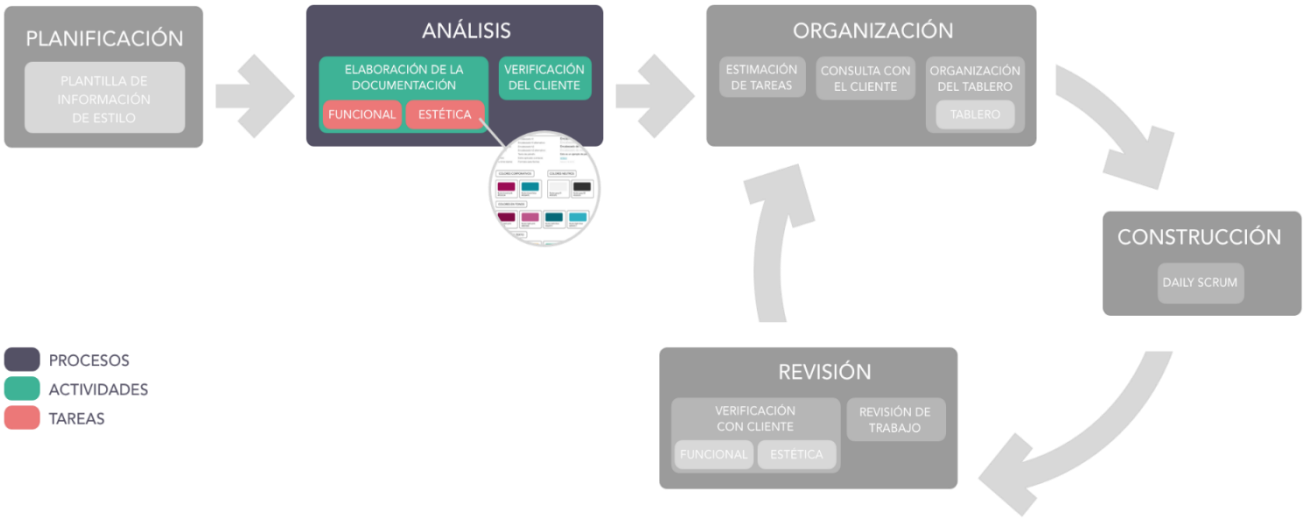


Figura 42. Ciclo de vida con análisis

7.3. Organización

Una vez tenemos todas las tareas de las funcionalidades y diseño que el cliente quiere se organizan de forma visual en el tablero. Para ello se realizan tres actividades, que se desarrollan a continuación.

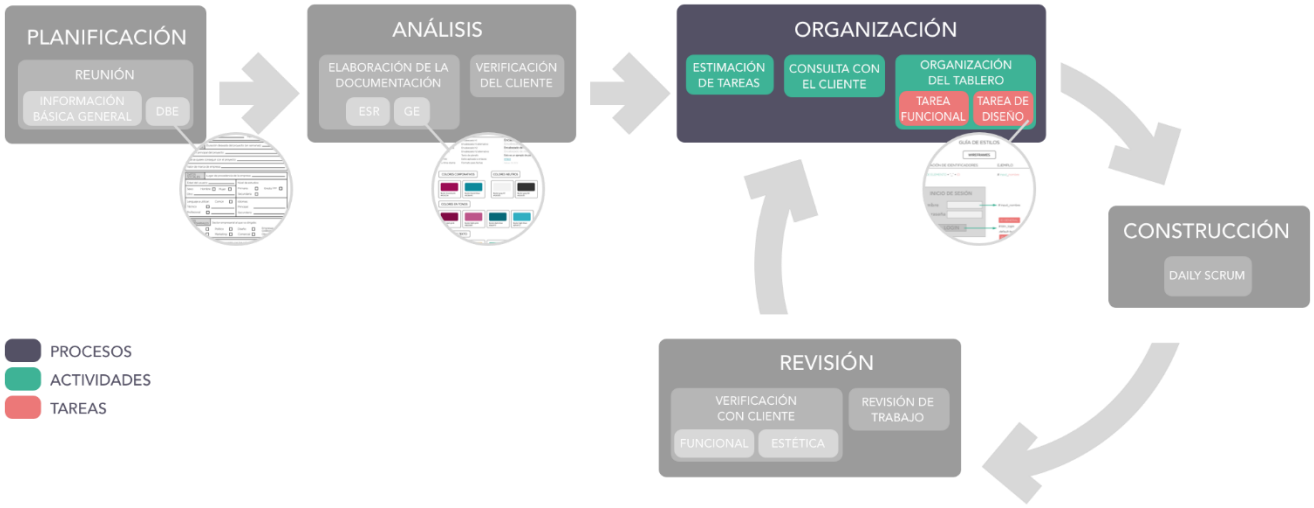


Figura 43. Ciclo de vida con organización

7.3.1. Estimación de tareas

Una vez realizado el análisis, el equipo se reúne y decide la lista de tareas de la iteración a realizar, estimando el esfuerzo y delegando las tareas entre los miembros del equipo. Este proceso sería como el Sprint Planning, con duración de 2 horas donde se delimita el contexto de trabajo en función a aquello que el equipo conoce o desconoce, al alcance que tiene y a los objetivos que hay que cumplir. La estimación se realiza de forma independiente, donde cada miembro del equipo puntúa en días lo que tardaría en realizar cada una de las tareas asignadas y se realiza una media para asignar una duración a la tarea.

Una vez estimada cada tarea cada tarea, se asigna a cada una prioridad, dependiendo del conjunto de tareas que forman el desarrollo, pues en caso de aproximarse a la fecha de finalización, habrá tareas que deban estar obligatoriamente y otras que no son de real importancia o simplemente sean detalles estéticos. El grado de prioridad se asigna como:

- **Bajo**. Las tareas de este grado suelen ser detalles estéticos sencillos fáciles de hacer, o funcionalidades simples.

- **Medio**. Por otro lado, las tareas de grado medio son tareas no estrictamente prioritarias, pero más necesarias en el desarrollo cuanto antes.

- **Alto**. Por último, las de grado alto, son tareas que se recomienda tener cuanto antes para agilizar el desarrollo y poder realizar el resto de las tareas. Son las más importantes y cuanto antes se hagan, mejor organizado estará el proyecto.

Además de la prioridad, cada tarea técnica está ligada a una tarea de diseño por un identificador (id) que el enlaza esa tarea de diseño con la tarea técnica. El número de identificador de la tarea técnica corresponde con el número de tarjeta de la tarea de diseño y viceversa. Cada tarea necesitará un estilo determinado para cada input del formulario, para cada botón, para colores, tipografías, etc.

Una vez estimadas las tareas y establecidos los identificadores correspondientes entre éstas, se procede con la organización del *sprint*, con duración de una semana, donde puede haber tareas de una pantalla en concreto, y luego, otras tareas de la misma pantalla, se pueden añadir al siguiente sprint. Es decir, no es necesario que en cada sprint finalice una pantalla, pues al terminar hay que entregar funcionalidades hechas, no pantallas hechas.

7.3.2. Consulta con el cliente

Esta actividad trata de una reunión con el cliente, a la que he denominado *Consultation Review*, ya que no corresponde a ninguna establecida en Scrum. En esta reunión, mediante la estimación realizada por el equipo, se da al cliente una macroestimación del desarrollo. Si la estimación es baja, y todo el equipo está de acuerdo, se pueden meter más historias en el *sprint*. Si la estimación es correcta y el cliente está de acuerdo, se comienza con la organización del desarrollo.

Esta reunión considero que es de real importancia, porque la comunicación entre el equipo y el cliente es prioritaria, pero en un desarrollo el cliente no es quien tiene la decisión completa en el proyecto, si no el equipo, también es partícipe, ya que es quien lo va a desarrollar. El hecho de que el equipo transmita su opinión respecto al tiempo que le llevará realizar las distintas tareas del desarrollo conlleva a una mejor planificación y una estimación más adecuada y coherente para las entregas.

7.3.3. Organización del tablero

Una vez tenemos ya las historias o tareas, su estimación y su prioridad, se limita *el Sprint Backlog o Pila Backlog* del tablero en función de las historias que haya puntuado el equipo. Estas historias se dividirán en "tareas técnicas" de las que se encargará el desarrollador, y "tareas de diseño", que se encargará el diseñador.

A continuación, podemos observar en la Figura 44 el tablero de mi propuesta, derivadado de la metodología previamente explicada Scrumban. La diferencia con un tablero Scrumban, es la incorporación de dos filas en una misma pantalla, para poder poner las tareas de diseño y técnicas de dicha funcionalidad que estén relacionadas. Es decir, cada interfaz se divide en el panel con sus historias correspondientes, las cuales se dividen en dos filas, una para tareas de diseño y otras para tareas técnicas. Si no hubiera una tarea de diseño asignada a una técnica, el tablero seguría el flujo normal de un tablero Scrumban.

El tablero sigue el diseño de Scrumban, con seis columnas, cuyas tarjetas de las tareas se desplazan de izquierda a derecha.

- **Sprint Backlog**. Esta columna contiene las tareas a realizar en el sprint correspodiente. Si no se realizan las tareas estimadas, se añaden al siguiente sprint. Esta columna y la siguiente las gestionan los responsables del proyecto.

- **Por hacer**. A esta columna se mueven las tareas con más prioridad para comenzar el desarrollo.

- **En curso**. En el momento en el que se asigna una tarea a un miembro del equipo, pasa a formar parte de la siguiente columna de en curso o en proceso.

- **Bloqueo**. Si la tarea sufre un bloqueo se mueve a esta columna, para que el project manager o scrum master, pueda encargarse de desbloquear la tarea y el equipo no tenga que preocuaprse por el desbloqueo y pueda continuar con otra tarea del desarrollo.

- **Hecho**. Una vez realiza la tarea, se mueve a esta columna, para que el tester tenga mejor localizadas las tareas que tiene que testear. Si la tarea contiene errores, se moverá a la columna “Por hacer”, si no, se mueve a “Test ok”.

- **Test ok**. Es la columna que da por finalizada y revisada una tarea del sprint. En cuanto un miembo del equipo sitúa una tarea en esta columna. Se asigna otra de la columna “Por hacer”.

La diferencia viene cuando sí que hay dos tareas relacionadas, y una es de diseño y otra técnica. Para agilizar el desarrollo del proyecto, el movimiento de estas tarjetas se realizará de manera independiente, pues estarán relacionadas mediante un identificador y dentro de su fila para que el equipo pueda ver en todo momento el estado de cada tarea, bloqueos, etc.

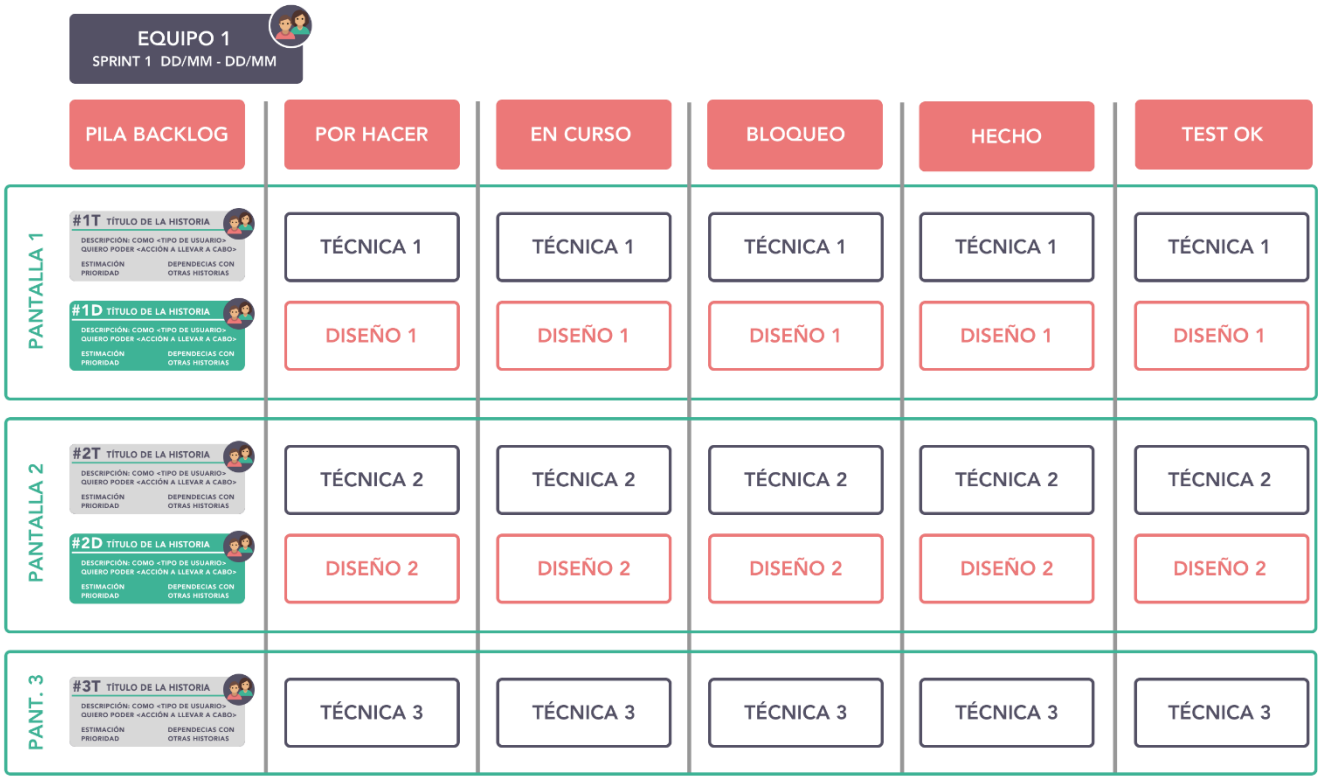


Figura 44. Tablero propuesta

Para diferenciar de forma visual unas tareas técnicas de unas de diseño, se utilizaría otro color distinto de casilla dentro de cada funcionalidad. Las tareas se moverían de la pila de tareas o Backlog al tablero en función de la prioridad asignada en su estimación, y habrá tantas tareas como se estime convenientemente en el *sprint*.

7.4. Construcción

Para no empezar a implementar a ciegas, se realiza el prototipo base, partiendo de la disposición de elementos establecido en el mockup y el documento realizado en la fase de análisis, como puede ser el Login, donde se defina los componentes más básicos (átomos) que compondrán el proyecto como establece la guía de estilos, como pueden ser los colores, las tipografías, los botones, los formularios, etc. Una vez establecido el prototipo se enseña al cliente para que verifique el estilo y la disposición de elementos. Una vez verificado se lleva a cabo la implementación del prototipo ya definido. De esta forma se consigue agilidad y productividad en el desarrollo.

En esta actividad, se realizarían diariamente los *Daily Scrum*, con una duración de 15 minutos, con posibilidad de alargarse lo que se necesite ya que llevar un seguimiento del trabajo, promueve el trabajo el equipo y genera compromiso entre los miembros. Estas reuniones son importantes, ya que el equipo puede resolver dudas o bloqueos que tengan y dar las correspondientes soluciones. En caso de que una tarea se haya estimado de forma incorrecta o surjan problemas inesperados, la tarea se moverá al siguiente *sprint*, pero nunca se podrá volver a planificar el *sprint*.

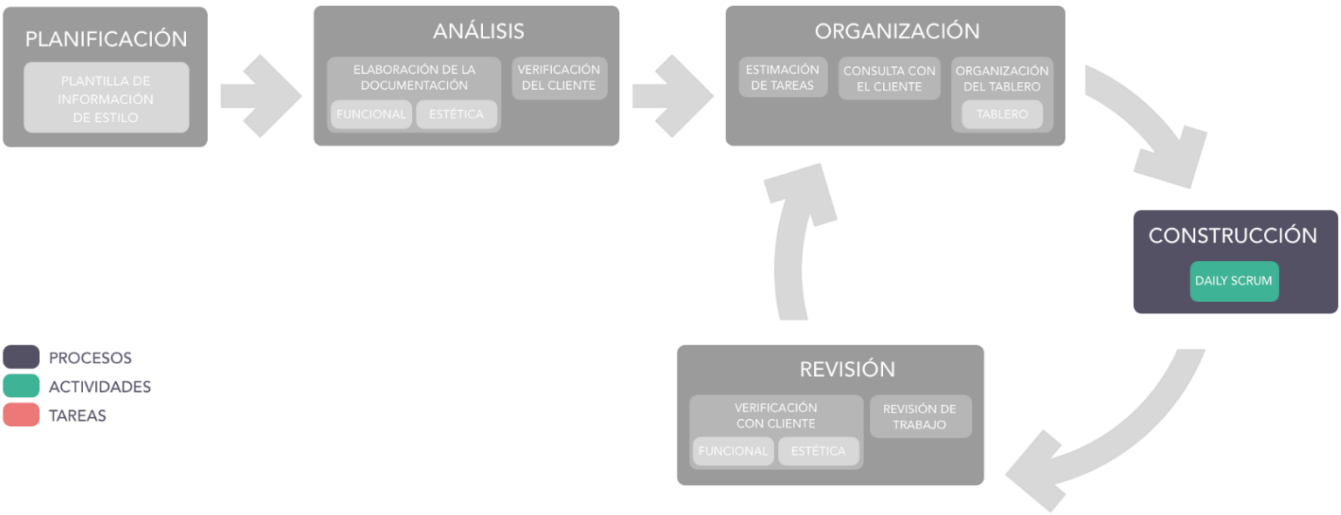


Figura 45. Ciclo de vida con construcción

Pongamos un ejemplo sencillo para entender este mecanismo. Supongamos que tenemos una funcionalidad que es la pantalla de login del usuario como podemos observar en la Figura 46, en la que podemos tener una tarea técnica que sea un formulario, menú de navegación y opciones varias. En este caso, tenemos los mensajes de éxito o de error, en función de si el usuario y la contraseña están correctos o no. Cada tarea de diseño está ligada a su correspondiente tarea técnica mediante el identificador que se establece en el proceso de estimación de tareas por parte del equipo.

Para que se pueda implementar de manera independiente al diseño y no tener bloqueos esperando a que se haga primero una y luego otra tarea, se utilizaría el documento realizado en la fase de Análisis llamado *GE*, el cual se construía a partir de los wireframe de estructura de los elementos que se deseen sus identificadores o clases asignados. Estos identificadores sirven para que luego, al realizar el desarrollo de ese elemento, con ponerle el nombre de la clase correspondiente se aplicaría el estilo.

Figura 46. Ejemplo de Login con tablero

Si Juan con su tarea de "Comprobar usuario" en la base de datos, se bloquea por cualquier motivo, Pepa puede continuar trabajando de manera independiente ya que sólo necesita conocer el nombre de la clase, especificado en el documento *GE* previamente nombrado, que identifica cada elemento sobre el que programa Juan.

Figura 47. Ejemplo de Login con tablero con bloqueo

Para finalizar con la explicación, vamos a analizar dos Figuras esquemáticas, la Figura 48 y la Figura 49 que reflejan la tarea de Juan, como programador o técnico, y de Pepa como diseñadora.

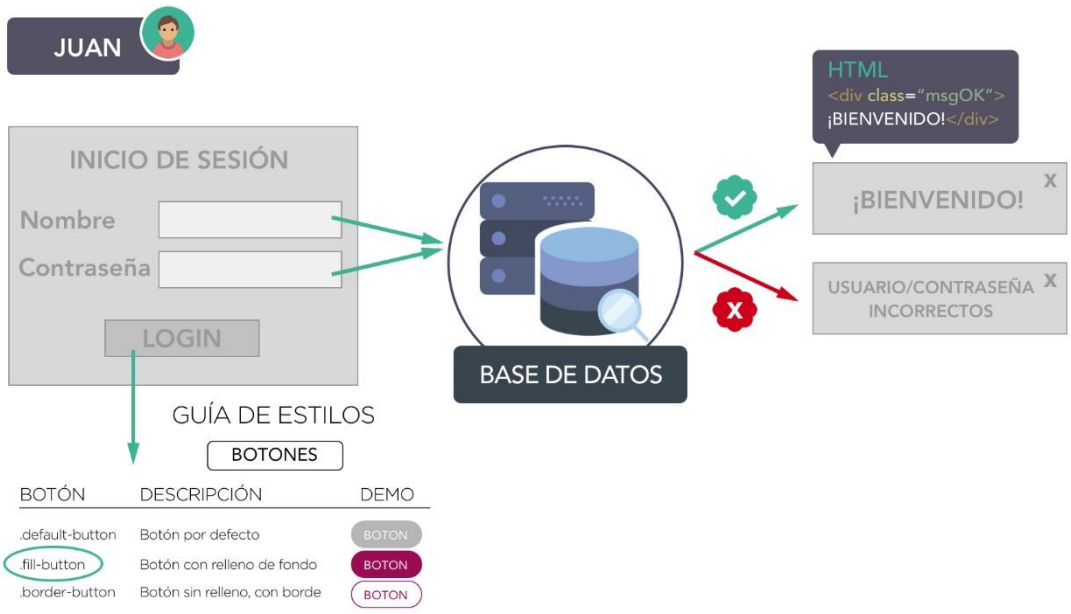


Figura 48. Esquema de función de programador

En la Figura 48 podemos observar que la función del programador, en este caso de Pepe, es dar funcionalidad a la página, sin tocar el diseño, pero utilizando las clases correspondientes según la guía de estilos para cada elemento, ya sea botones o mensajes emergentes. Dependiendo de un resultado u otro obtenido en la base de datos, el mensaje será uno u otro, es decir, que el control de los mensajes a nivel funcional o técnico, es del programador, no del diseñador.

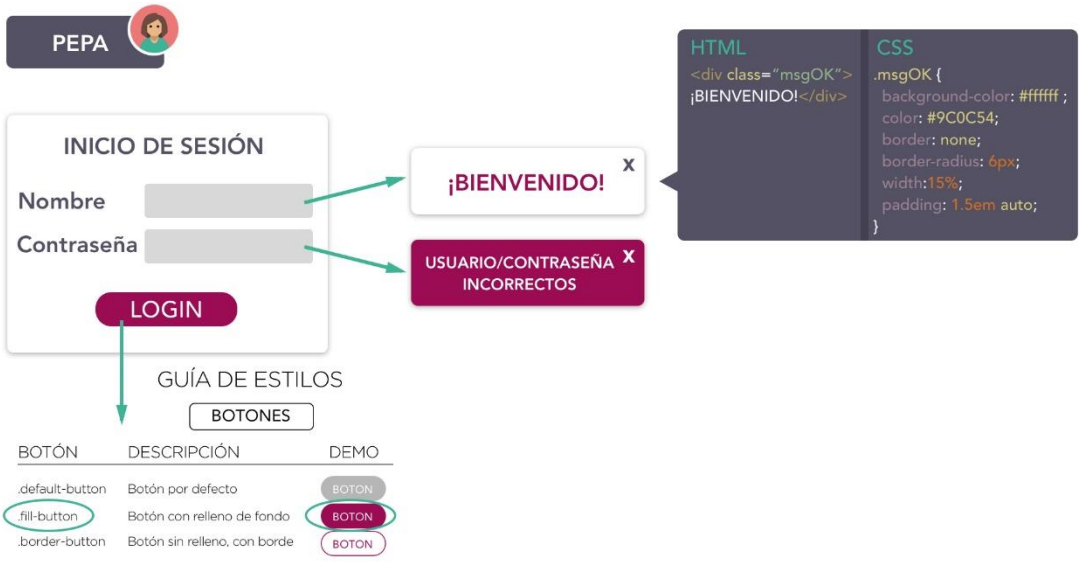


Figura 49. Esquema de función de diseñador

Por otro lado, en la Figura 49 podemos ver que el diseñador, en este caso Pepa la diseñadora, se encarga de utilizar las clases de la guía de estilos para darles color y estilo, sin tocar nada de funcionalidad.

7.5. Revisión

Por último, en este proceso, como su nombre indica, se realizan las revisiones finales del desarrollo que se ha llevado a cabo en el *sprint*. Éste, a su vez, se divide en otras dos actividades, que vamos a ver en detalle a continuación.

7.5.1. Verificación con cliente

Esta reunión correspondería a la Sprint Review, con duración de 2 horas donde el equipo presenta al cliente su proceso y ésta verifica lo que se ha realizado. Esta actividad está dividida en dos tareas:

- **Revisión del documento funcional**. El cliente da el visto bueno a las funcionalidades especificadas en el documento, tal y como se especificó en el primer proceso de planificación.

- **Revisión del documento de estilos**. En caso de proponer cambios en el diseño, al estar componetizado, no supondría gran esfuerzo, pues simplemente habría que cambiar el valor de los atributos de las clases definidas, como explicamos en el apartado de Atomic Design.

7.5.2. Revisión de trabajo

Una segunda actividad de la revisión sería a nivel de equipo, donde los miembros se reúnen con el responsable del proyecto para evaluar la metodología como Scrum utiliza el *Sprint Retrospective*, con duración de 2 horas. El feedback constante es muy importante para poder evolucionar y mejorar en la medida de lo posible, y de este modo crear el ambiente de trabajo adecuado y mantener al equipo motivado. El equipo deberá contestar una serie de preguntas como pueden ser:

- ¿Cómo fue la iteración pasada?
- ¿Qué problemas surgieron?

- ¿Qué se hizo bien?
- ¿Qué se quiere mejorar?
- ¿Qué hacer para conseguir esa mejora?

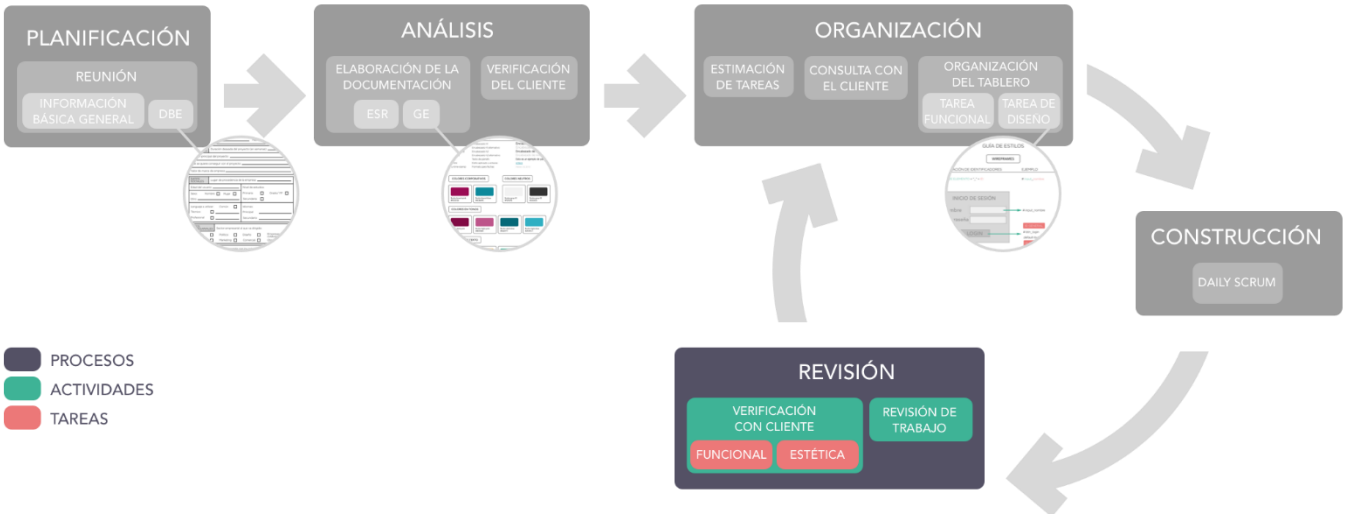


Figura 50. Ciclo de vida con revisión

7.6. Roles

Una vez establecido el proceso principal con los procesos, con sus actividades y tareas, vamos a analizar los diferentes roles que compondrían la metodología para su correcto desarrollo. Estos serían muy parecidos a los que tienen Scrum o Kanban:

1. **Cliente**. En Scrum sería el Product Owner el encargado de que el producto desarrollado cumpla con las expectativas del cliente, pero en este caso, sería mejor omitir este rol y dejar actuar al cliente frente al Project Manager y al equipo. El cliente va a exponer su necesidad y va a proporcionar la información que el equipo necesite.

2. **Project Manager**. Es el perfil encargado de todo tipo de comprobaciones y seguimiento de desarrollo. En Scrum sería como el Scrum Master, también encargado de desbloquear las tareas que no permitan al equipo avanzar y apoyar en lo que sea necesario. Estará desde el principio, es decir, la toma de requisitos y la primera reunión con el cliente y el equipo, hasta el final, con la reunión de revisión del producto una vez esté finalizado.

En la reunión de revisión, debe guiar dicha reunión, pero sin participar en las opiniones y tratar los temas cuando sea necesario.

3. **Equipo**. Se trataría de un equipo multidisciplinar, como Scrum propone, para poder asignar las tareas de forma que cada miembro realice las que mejor se considere capaz de afrontar o más facilidades tenga.

   Dentro del equipo, habrá dos perfiles diferenciados, el de desarrollador o técnico y el de diseñador, ya que la metodología que propongo trata de agilizar la parte de diseño del desarrollo de un proyecto.

   - **Técnico**. El perfil técnico, es el encargado de toda la parte funcional, es decir, desde la recogida de información hasta la implementación, éste opina acerca de cómo implementar lo que el cliente pide, independientemente del diseño que tenga el proyecto.

   - **Diseñador**. Este rol, sin embargo, no necesita saber de programación, sólo maquetará y tomará decisiones acerca de la parte estética del proyecto. Desde las primeras reuniones de recogida de información, hasta los mínimos detalles estéticos, es el perfil que opinará y decidirá acerca del diseño.

8. Pruebas y validación

Una vez ya está desarrollada la propuesta de metodología para agilizar la fase de diseño en proyectos web, falta pasar por la validación del método que verifique su fiabilidad. Para validar la metodología, vamos a analizar un desarrollo sencillo con todas las diferentes partes que lo componen, sus documentos y su planificación.

El proyecto de ejemplo es una aplicación web de consulta de mensajería, como un buzón de mensajes. Esta aplicación la usamos en la empresa en la que actualmente trabajo y considero que es un buen ejemplo ya que no requiere de una cantidad de interfaces tan grande como para usarlo de prueba. La empresa proporciona un nombre de usuario, por lo que no requiere de una interfaz de registro, solamente servirá la de inicio de sesión. Una vez el usuario inicia sesión, se conducirá a la segunda y última interfaz donde se pueden visualizar los mensajes recibidos a ese correo o usuario, sin poder contestarlos.

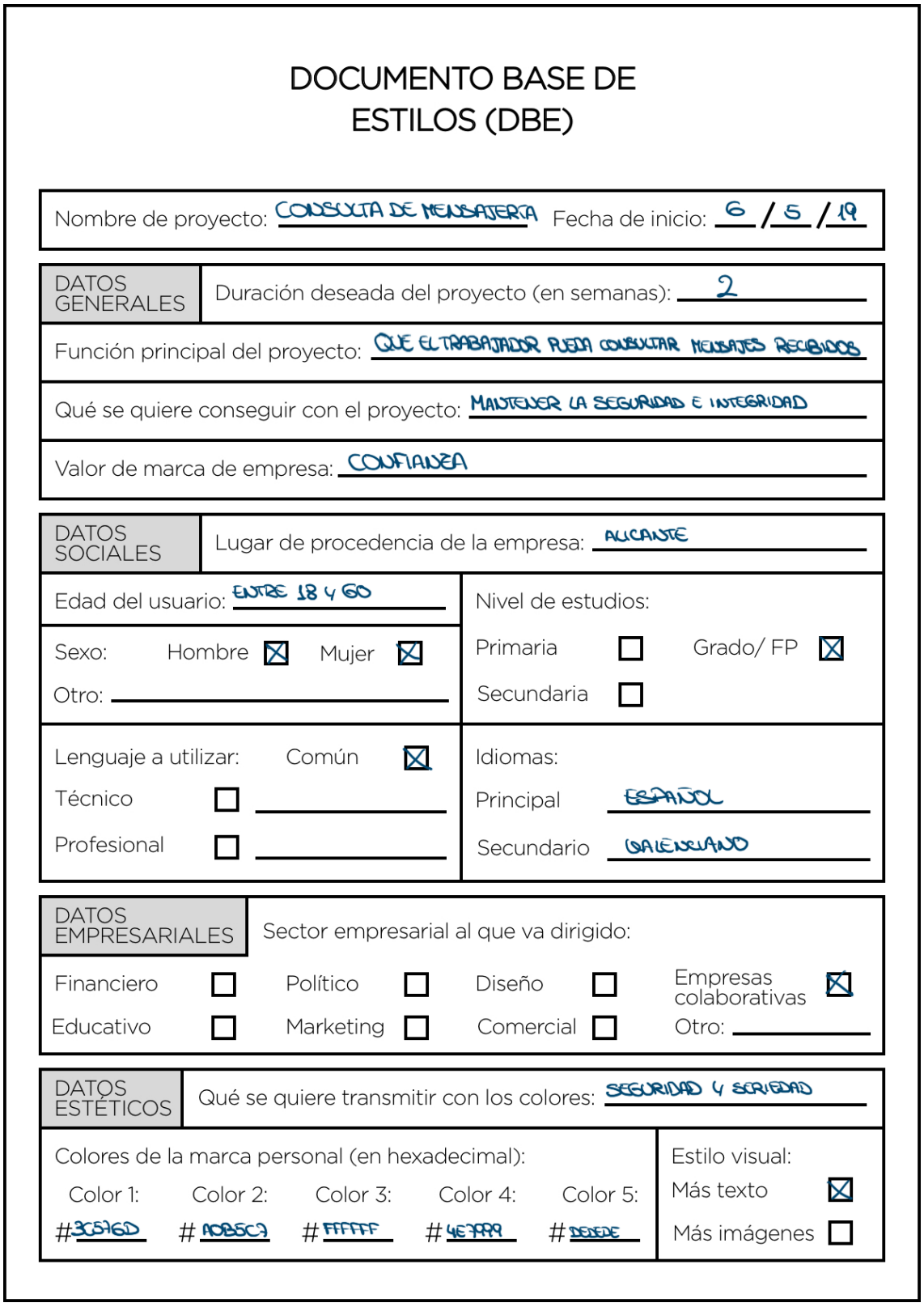
El equipo estará formado por un desarrollador técnico y un diseñador para diseñar la parte estética del proyecto. El desarrollo constará de una iteración.

8.1. Planificación

Día 1, 06 de abril de 2019

Figura 51. Día 1 de planificación

En primer lugar, se realiza la reunión con el cliente con una duración de 2h. En esta reunión el cliente nos cuenta acerca de su empresa y el objetivo que tiene con el proyecto. La empresa es una consultora informática de desarrollo de proyectos web, donde tiene trabajadores de diseño, marketing, etc. El proyecto que quiere realizar esta empresa es una pequeña aplicación en la que el trabajador pueda entrar con un nombre ya proporcionado por ésta y acceda directamente a una interfaz de mensajería. Mientras tanto, el equipo recoge toda la información que podemos ver reflejada en el *DBE* de la Figura 52.



DOCUMENTO BASE DE ESTILOS (DBE)

Nombre de proyecto: CONSULTA DE MENSAJERIA Fecha de inicio: 6 / 5 / 19

DATOS GENERALES

Duración deseada del proyecto (en semanas): 2

Función principal del proyecto: QUE EL TRABAJADOR PUEDA CONSULTAR MENSAJES RECIBIDOS

Qué se quiere conseguir con el proyecto: MANTENER LA SEGURIDAD E INTEGRIDAD

Valor de marca de empresa: CONFIANZA

DATOS SOCIALES

Lugar de procedencia de la empresa: ALICANTE

Edad del usuario: ENTRE 18 Y 60

Sexo: Hombre ☒ Mujer ☒

Otro: _____

Nivel de estudios:

Primaria ☐ Grado/ FP ☒

Secundaria ☐

Lenguaje a utilizar: Común ☒

Técnico ☐ _____

Profesional ☐ _____

Idiomas:

Principal ESPAÑOL

Secundario GALENCIANO

DATOS EMPRESARIALES

Sector empresarial al que va dirigido:

Financiero ☐ Político ☐ Diseño ☐ Empresas colaborativas ☒

Educativo ☐ Marketing ☐ Comercial ☐ Otro: _____

DATOS ESTÉTICOS

Qué se quiere transmitir con los colores: SEGURIDAD Y SERIEDAD

Colores de la marca personal (en hexadecimal):

Color 1: #305960 Color 2: #A0B5C9 Color 3: #FFFFFF Color 4: #4E7999 Color 5: #DEDEDE

Estilo visual:

Más texto ☒

Más imágenes ☐

Figura 52. Documento base de estilos de ejemplo

Día 2, 07 de abril de 2019

Figura 53. Día 2 de planificación

El segundo día, el equipo repasa la documentación recogida del día anterior y revisa cada apartado. Es entonces, cuando al ver que el cliente quiere el proyecto para dentro de 2 semanas, no se verían con tiempo suficiente para el desarrollo, por lo que contactan con el cliente y solicitan una ampliación del plazo a 3 semanas. El proyecto ahora pasa a desarrollarse en 3 semanas.

8.2. Análisis

8.2.1. Elaboración de la documentación

Día 3, 08 de abril de 2019

Figura 54. Día 3 de análisis

Una vez finalizada el proceso de *planificación*, con toda la información recogida de lo que se quiere tener, se da paso al proceso de *análisis*. Los dos primeros días, sirven para realizar los documentos funcional *ERS* y la guía de estilos *GE*. El documento funcional recoge los aspectos más técnicos que los desarrolladores necesitarán, como puede ser la gestión de las llamadas a la base de datos, así como la recogida de datos, el funcionamiento de los diferentes componentes de las interfaces, validación de datos, etc.

Pero vamos a hacer más hincapié en la parte estética, donde se documenta la *GE*. El documento de guía de estilos, como podemos observar en la Figura 55 contendrá:

- Las **tipografías** de los textos para los mensajes y encabezados para los títulos que tendrá la aplicación.

- Los **colores** que se utilizarán para el diseño serán tanto los corporativos de la empresa como neutros para determinadas funciones, en tonalidades para jugar con el diseño y de texto para mensajes informativos al usuario como puede ser de error o validación. La información que recogimos anteriormente por parte del cliente especifica que la empresa de éste tiene como valor de empresa la confianza y la seguridad y, por lo tanto, es lo que desea trasmitir con la aplicación web. Es por ello, que una paleta de colores fríos trasmite esa seriedad al diseño, en concreto, el azul claro, da la sensación de sinceridad y confianza.

- Los **botones** tendrán por defecto un estilo determinado, pero si el diseñador quiere asignar otro estilo diferente dependiendo de la función del botón, se puede elegir entre los establecidos en este documento.

- El único elemento de tipo formulario que habrá, es un **input** para introducir el nombre del usuario en el inicio de sesión.

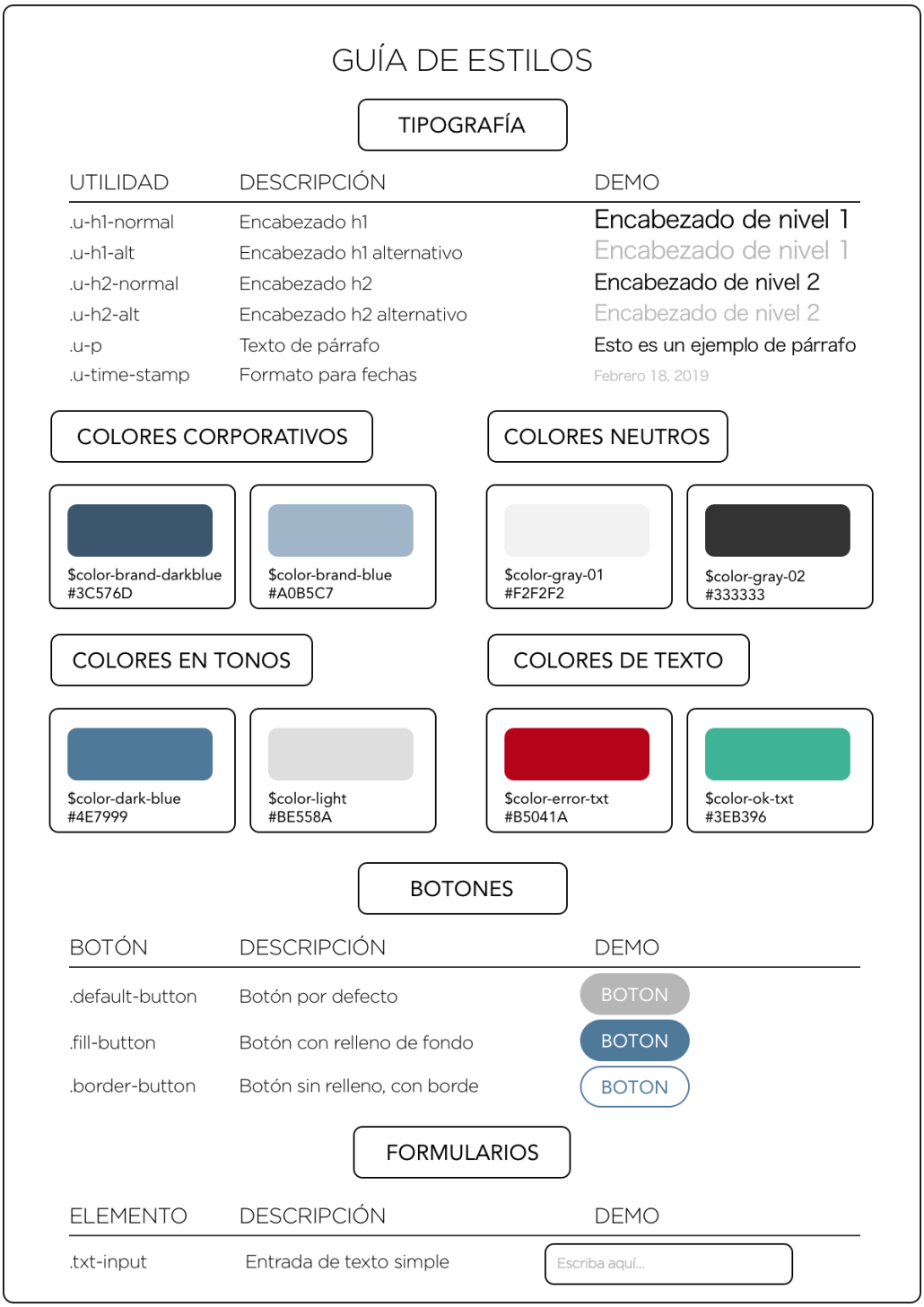


Figura 55. Documento guía de estilos de ejemplo

La primera parte del documento se refiere más a lo relacionado con colores y tipografías de los distintos elementos que tendrán las interfaces, en cambio, la segunda parte del documento se enfoca más a nivel de interfaz y su distribución como se puede observar en la Figura 56.

El diseño de la aplicación será minimalista, es decir, con pocos elementos, ya que el público al que va dirigido es muy amplio, desde jóvenes que están más acostumbrados al uso de aplicaciones web como gente mayor, que no tiene tanta destreza. Por lo que lo ideal es una aplicación lo más intuitiva posible.

La segunda parte del documento contiene el flujo técnico y las interfaces en wireframe, es decir, sin un alto nivel de detalle.

- El **flujo técnico** es sencillo, ya que sólo hay dos pantallas y de la pantalla de inicio de sesión se accede a la de mensajes y viceversa.

- Los **wireframes** también son sencillos e intuitivos. Además, cada elemento tiene su identificador si es único o su nombre identificativo de tipo clase, si hay más de un elemento igual.

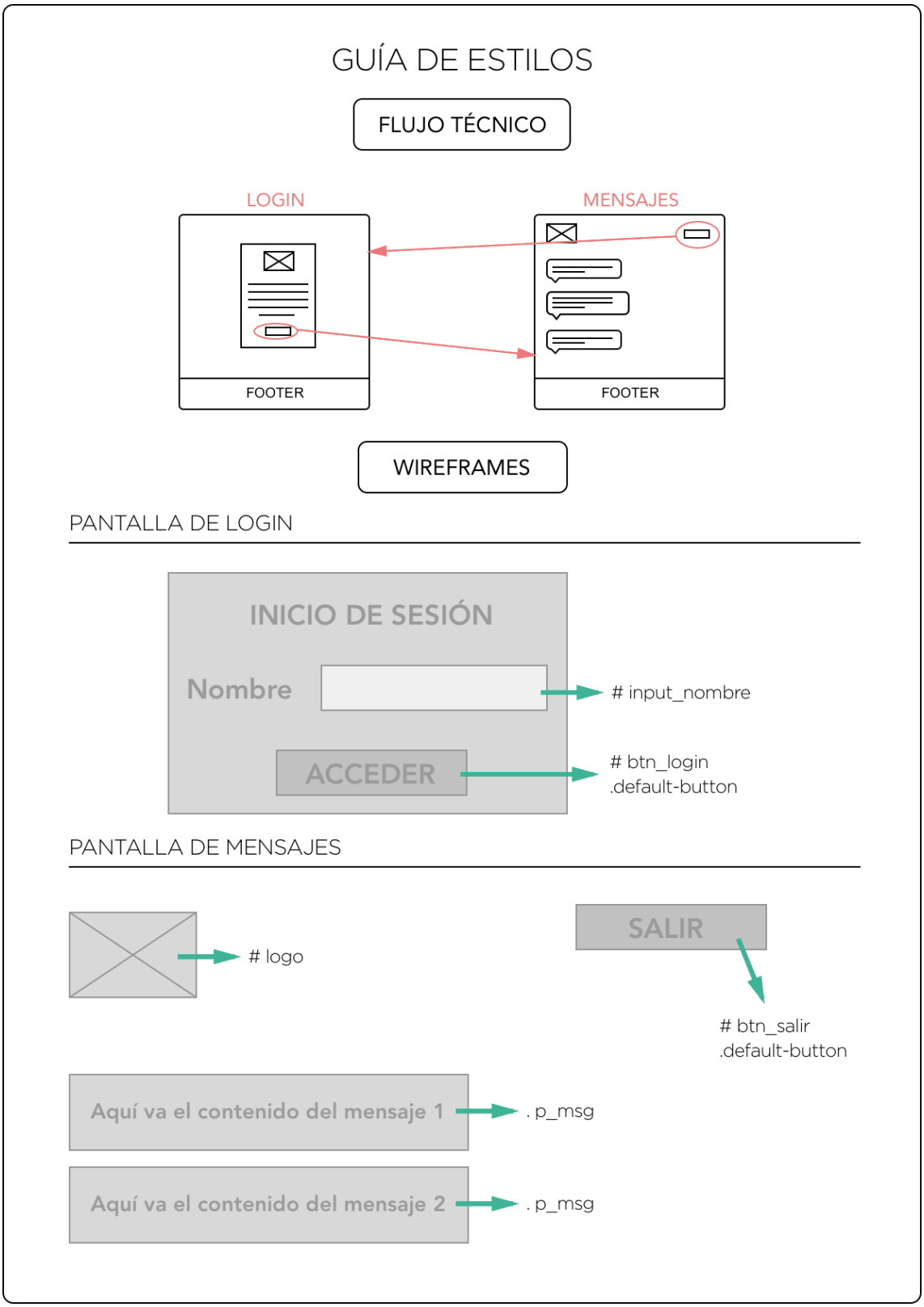


Figura 56. Documento guía de estilos de ejemplo 2

Día 4, 09 de abril de 2019

Figura 57. Día 4 de análisis

El siguiente día, se dedicará también a la elaboración de los dos tipos de documentos, *ERS* y *GE*, resolviendo posibles dudas y detallando cada apartado.

8.2.2. Verificación con el cliente

Día 5, 10 de abril de 2019

Figura 58. Día 10 de análisis

El último día de la semana, es el último día dedicado al proceso de *análisis*. Una vez finalizados los documentos funcionales y de diseño, el equipo se reúne con el cliente y se da paso a la *Scrum Review*. En esta reunión se enseña todo lo que se ha documentado al cliente, y si a éste le parece bien se continúa con el siguiente proceso, si se tiene que realizar algún cambio, se realiza a lo largo del día, ya que esta reunión sólo dura una hora. Si fuera preciso, se puede ampliar un día más para los cambios necesarios en la documentación, pero en este caso, el cliente da el visto bueno y no hay que modificar nada.

8.3. Organización

8.3.1. Estimación de tareas

Día 6, 13 de abril de 2019

Figura 59. Día 6 de organización

Una vez tenemos toda la información recogida y documentada, damos paso a la organización, donde se desglosa cada funcionalidad y se distingue entre técnica y de diseño.

La estimación la realiza el equipo en la reunión de Sprint Planning donde cada miembro, en este caso el desarrollador y la diseñadora, estiman cada tarea ya desglosada en función del tiempo que los va a llevar realizarla. La estimación por tarea se evalúa en días, y después se saca la media entre la estimación de un miembro del equipo y otro, como podemos observar en la Figura 60.

La tabla contiene tres columnas, una para la descripción de la tarea técnica o de diseño, otra para su duración estimada en días, como se ha explicado, y otra para el identificador. Este identificador se asignará a las tareas técnicas y de diseño que tengan relación entre sí para después, poder organizar el tablero.

ESTIMACIÓN DE TAREAS

LOGIN

TAREA TÉCNICA	DURACIÓN (EN DÍAS)	ID	PRIORIDAD
#1T_Llamadas a BBDD	1	-	Alta
#2T_Comprobar usuario si es correcto, y mostrar mensajes éxito/error	1	#9D	Media
#3T_Redirigir botón a pantalla de mensajes	0.5	#10D	Baja
TOTAL DE DURACIÓN (EN DÍAS)	2.5		
MEDIA DE DURACIÓN (EN DÍAS)	**3**		

TAREA DE DISEÑO	DURACIÓN (EN DÍAS)	ID	PRIORIDAD
#7D_Maquetar formulario	0.5	-	Media
#8D_Realizar archivo de estilo común del GE	2	-	Alta
#9D_Colores de mensajes	0.5	#2T	Baja
#10D_Cambiar estilo de botón al pulsar y sin pulsar	0.5	#3T	Baja
	3.5		

MENSAJES

TAREA TÉCNICA	DURACIÓN (EN DÍAS)	ID	PRIORIDAD
#4T_Llamadas a BBDD	1	-	Alta
#5T_Mostrar mensajes y su fecha	0.5	#13D	Media
#6T_Redirigir botón a pantalla de login	0.5	#14D	Baja
TOTAL DE DURACIÓN (EN DÍAS)	2		
MEDIA DE DURACIÓN (EN DÍAS)	**2**		

TAREA DE DISEÑO	DURACIÓN (EN DÍAS)	ID	PRIORIDAD
#11D_Maquetar textos de mensajes	0.5	-	Media
#12D_Maquetar mensajes	0.5	-	Media
#13D_Colocar loco	0.5	#5T	Media
#14D_Cambiar estilo de botón al pulsar y sin pulsar	0.5	#6T	Baja
	2		

Figura 60. Estimación de tareas

8.3.2. Consulta con el cliente

Por lo estimado, se lleva a la conclusión de que el desarrollo constará solamente de un *sprint* con una duración de una semana contando con el inicio de desarrollo al día siguiente de organizar el tablero, pues son tres días para la pantalla de login y dos días más para la pantalla de mensajes.

Los resultados de esta reunión se consultan con el cliente en la reunión de *Consultation Review*. En esta reunión se da la estimación en tiempo y una breve explicación de lo que se va a desarrollar y el cliente queda informado de la duración del proyecto.

8.3.3. Organización del tablero

Día 7, 14 de abril de 2019

Figura 61. Día 7 de organización

Con las tareas ya definidas, viene la organización del tablero en función de los *sprints*. En este caso sólo hay un sprint.

En primer lugar, se divide el tablero en filas, según las pantallas que existan, en este caso son dos: la pantalla de login y la pantalla de mensajes. Dentro de cada fila correspondiente a su pantalla, se introducen en la columna de "Pila Backlog" todas las tareas a realizar, con distintas tarjetas para las tareas técnicas de las de diseño. El orden se establece de izquierda a derecha hasta validar la tarea e introducirla en el desarrollo general del proyecto. Para escoger qué tarea hacer en primer lugar, se observará el orden de prioridad que tenga asignada la tarea y se escogerá de mayor a menor prioridad. En la Figura 62 se puede observar cómo sería el estado inicial del tablero.

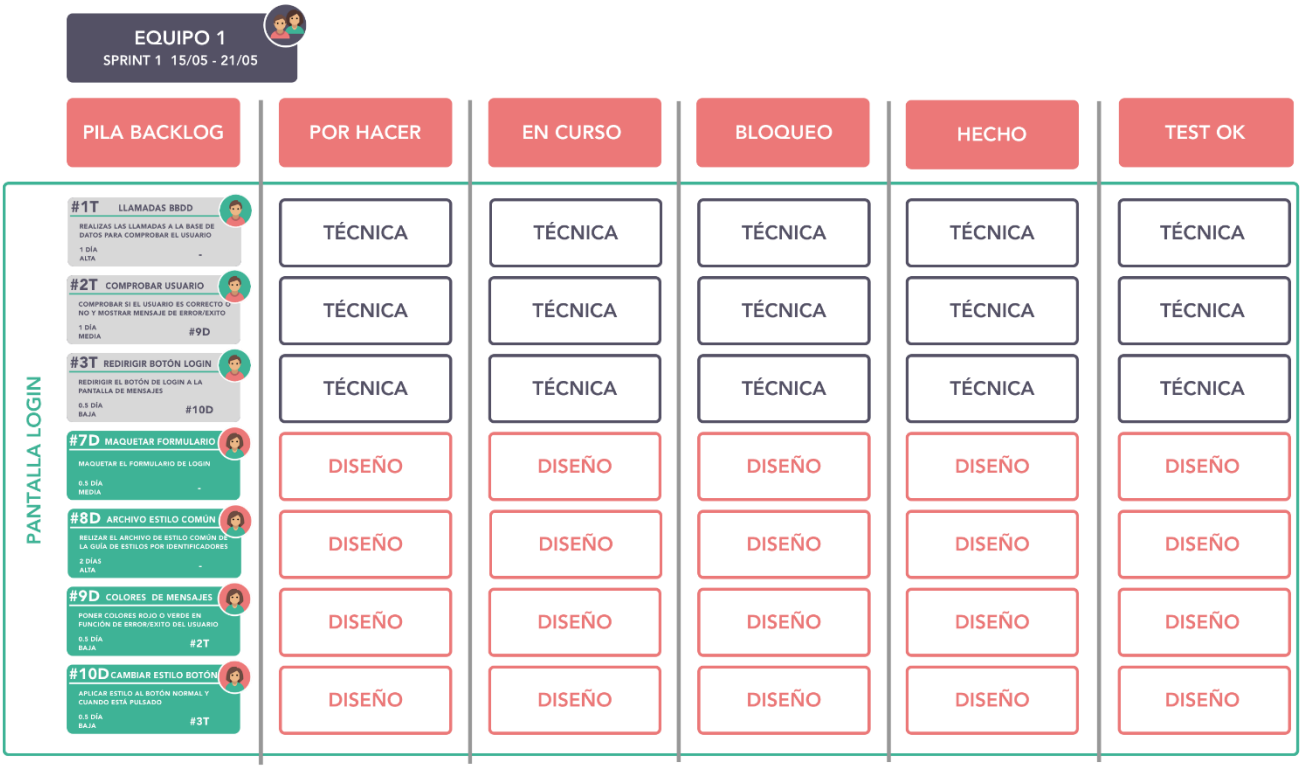


Figura 62. Tablero de validación

8.4. Construcción

Día 8, 15 de abril de 2019

Figura 63. Día 8 de construcción

Es el día del comienzo del *sprint*. Las primeras tareas para realizar son las de prioridad más alta, en el caso del diseñador, la tarea más prioritaria es la de elaborar un archivo de estilo común basado en la *GE*, ya que, una vez elaborado cada identificador con su respectivo estilo, sólo habrá que coger la clase que se desee y aplicarla al elemento.

Según la estimación realizada, la tarea de dicho archivo de estilo común, ocupa todo el día porque lo que es la única tarea para realizar en el día de hoy. En la Figura 64 se puede observar la evolución del estado del tablero.

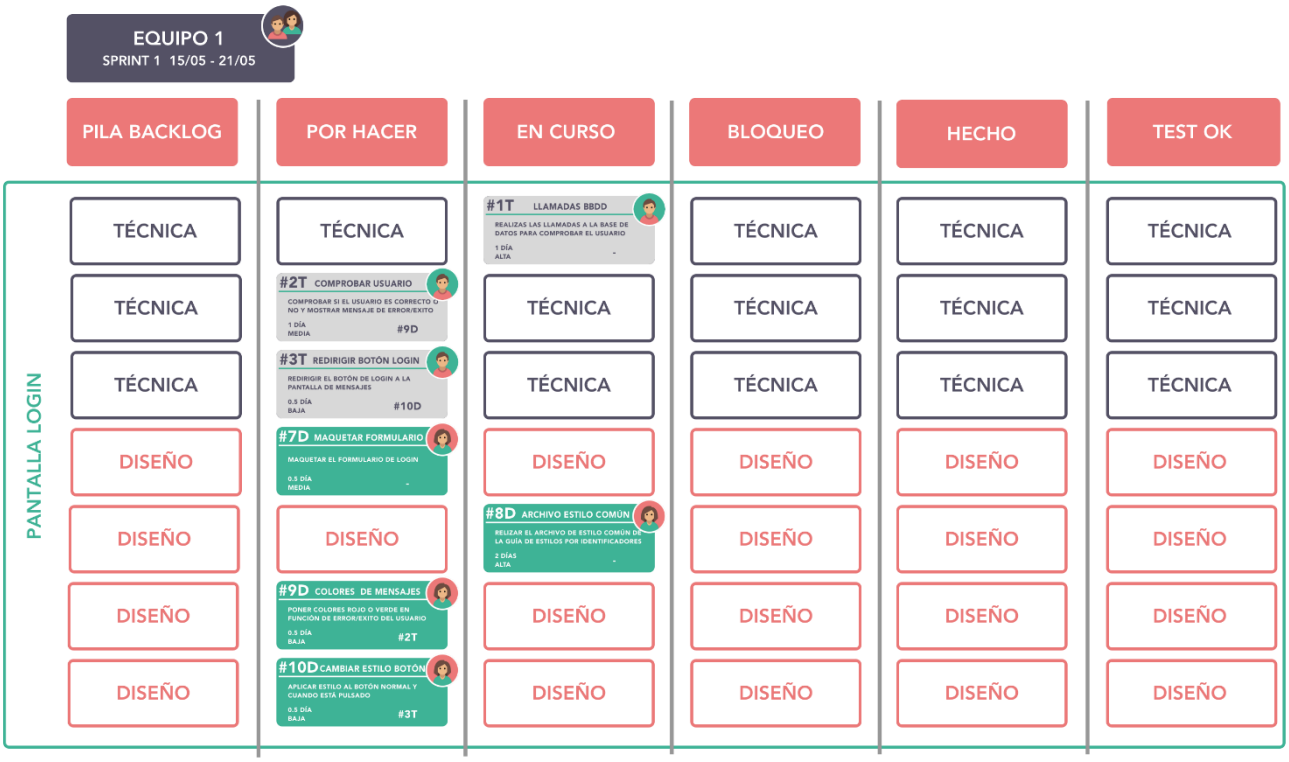


Figura 64. Evolución del tablero de validación

Día 9, 16 de abril de 2019

Figura 65. Día 9 de construcción

Tal y como se estimó, la tarea de diseño de elaborar el archivo de estilo común, ocupa dos días enteros por lo que el siguiente día se dedica a la misma tarea.

Día 10, 17 de abril de 2019

Figura 66. Día 10 de construcción

El archivo de diseño ya está finalizado y es consistente para ser utilizado a partir de ahora para elaborar de manera más ágil el diseño del resto de elementos. La evolución del proyecto se puede observar en la Figura 67.

Figura 67. Pantalla de login de validación

Los elementos modificados son los que se visualizan con estilo aplicado referente al especificado en la *GE*. El técnico por un lado implementa la validación del usuario que se introduce en el campo de entrada "nombre" y si es correcto o no, muestra un mensaje de error u otro. Pero esto no concierne al diseñador, que aplicara el estilo correspondiente a ambos, por ejemplo, si el usuario es correcto, verde y si es incorrecto, rojo. Pero ambos pueden trabajar sobre el mismo identificador gracias a la documentación en la GE y de este modo, se aplica esa filosofía al resto de elementos.

Día 11, 20 de abril de 2019

Figura 68. Día 11 de construcción

Una vez finalizada la pantalla de inicio de sesión, se comienza con la de mensajes. El estado de la interfaz resultante es el de la Figura 69.

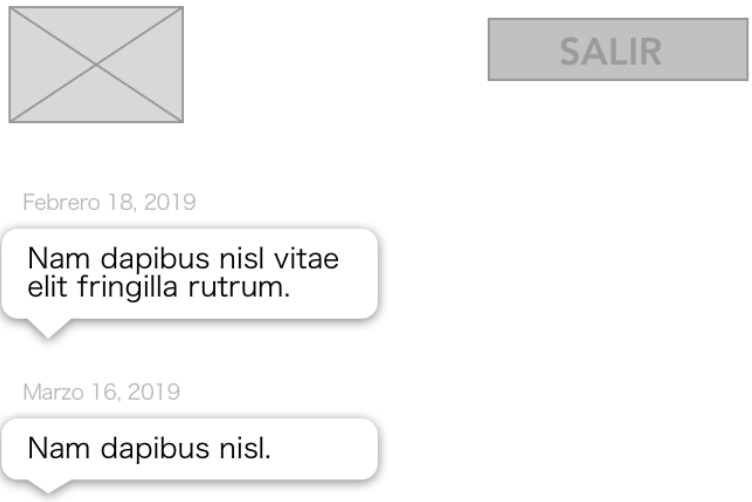


Figura 69. Pantalla de mensajes de validación

Las tareas de la pantalla de login se quedan todas en la columna de "Hecho", esto significa que, para el final del *sprint,* se procederá con el test para dar las tareas por finalizadas y revisadas una vez se han implementado. Una vez pasadas el test, es cuando se añaden al desarrollo general del proyecto.

Día 12, 21 de abril de 2019

Figura 70. Día 12 de construcción

Este es el último día en el que se desarrolla el proyecto, dando paso a la revisión de lo que se ha elaborado. Todas las tareas pasan a la columna de finalizado en el tablero y se procede con su revisión o testeo. En la Figura 71 podemos observar el resultado del desarrollo.

Figura 71. Pantalla de mensajes finalizada

8.5. Revisión

8.5.1. Verificación con el cliente

Día 14, 23 de abril de 2019

Figura 72. Día 14 de revisión

Los tres últimos días de la semana serán dedicados a la revisión. En primer lugar, está la reunión con el cliente, que corresponde con *la Sprint Review*, para verificar el desarrollo que se ha llevado a cabo.

El cliente da el visto bueno a todo lo que se ha desarrollado y se queda satisfecho con el trabajo, pero indica al equipo que le gustaría un color diferente para los botones. El equipo recoge el código del color que solicita el cliente y procede con el cambio.

Gracias a que se contempla cada identificador y cada clase en el archivo de estilo base realizado al comienzo y sobre el que se ha basado el resto de los elementos del proyecto, basta con cambiar la propiedad de color referente a la petición del cliente y se modifica todo el estilo automáticamente.

8.5.2. Revisión de trabajo

Día 15, 24 de abril de 2019

Figura 73. Día 15 de revisión

Una vez finalizado el desarrollo, y validado por el cliente, se procede con la última reunión, la Sprint Retrospective. Esta reunión es diferente, ya que no se enfoca a la relación del cliente con el equipo, sino del equipo entre los diferentes componentes y la forma de trabajo llevada a cabo. Las preguntas son:

- **¿Qué problemas surgieron?** No ha surgido ningún problema ya que el desarrollo se ha realizado de forma paralela y así se ha conseguido agilidad y eficacia.
- **¿Qué se hizo bien?** Lo mejor fue realizar el archivo de estilo base ya que para los cambios que luego se realizaron a petición de cliente, resultaron más sencillos, pues bastaba con cambiar una propiedad del identificador correspondiente y automáticamente se aplicaban los cambios a los elementos con ese identificador.

En la Figura 74 se puede observar cómo ha quedado el desarrollo total del proyecto reflejado en el calendario, desde el primer día con la planificación de la información recogida hasta la construcción día a día y su revisión una vez finalizado.

Figura 74. Calendario general

9. Resultados

Los resultados han sido mejor de lo esperado gracias a la motivación y a la involucración tanto mía como autora, como de mi tutor y amigos con conocimientos de metodologías ágiles para desarrollo de proyectos que me han aconsejado y orientado adecuadamente.

Tras haber dedicado tanto tiempo a estudios de las diferentes asignaturas y haber experimentado en el mundo laboral la aplicación de estas asignaturas al mundo real, quería aportar mi grano de arena a algo tan útil como son las metodologías en lo que concierne al desarrollo de proyectos web, pues es muy importante planificar, organizar y establecer un orden a la hora de enfrentarse a un proyecto.

El trabajo ha sufrido grandes cambios conforme su desarrollo, pues hasta que no ejemplificas un marco teórico, no es posible confirmar su utilidad y su veracidad. Constantemente surgían dudas, que se han ido resolviendo a lo largo de la investigación y las consultas en todo lo que abarcan las metodologías.

Existen numerosas metodologías para llevar a cabo un proyecto, pero yo he propuesto una adaptación que no he visto contemplada en estas metodologías. Un proyecto web se compone de diseño y funcionalidad, y ambos son iguales de importantes y ambos deben agilizarse en el desarrollo.

Espero que la metodología que propongo sea útil y pueda ayudar a muchos equipos de desarrollo para agilizar el proyecto como me ha ayudado a mí.

10. Conclusiones y trabajo futuro

Durante toda la carrera nos han enseñado que el perfil de ingeniero es el encargado de dar solución a problemas existentes. Con este proyecto no sólo se da solución a un problema necesariamente desde el punto de vista técnico, si no desde un punto de vista estético complementario a éste, pues toda aplicación web tiene una parte de diseño, más visual y atractiva que complementa la funcionalidad.

El método de validación utilizado para probar mediante un sencillo ejemplo el uso de ésta metodología ágil de diseño en proyectos web no es un ejemplo de desarrollo real, pero es un ejemplo sencillo de comprender y fácil de aplicar a una aplicación real ya existente.

En el mundo laboral y educacional, se utilizan numerosas metodologías ágiles que funcionan y dan buenos resultados. Con este trabajo propongo una mejora de estas metodologías ya existentes y con un uso extendido, por lo que los resultados aplicados a un proyecto real considero que pueden ser bastante óptimos.

Como trabajo futuro me gustaría llegar a aplicar la metodología a un proyecto más grande, con un equipo multidisciplinar que se organice conjuntamente para desarrollarlo de forma ágil y eficaz. De este modo, surgen mejoras conforme se prueba su funcionamiento, y se puede llegar a crear una metodología de desarrollo de proyectos web inexistente hasta el momento, que pueda ayudar a muchos equipos.

Tanto la organización y la planificación como la parte visual y estética de los proyectos, siempre me ha llamado mucho la atención y he buscado mucha información para mejorar mi formación todo lo posible. Este trabajo me ha ayudado a formarme en este ámbito y conocer otras nuevas metodologías que se utilizan, así como formas de organización de código y diseño que podré aplicar en proyectos míos o de empresas y, sobre todo, a mi futuro profesional.

Referencias

[1] Metodología ágil Scrum https://blog.conectart.com/metodologias-agiles/

[2] 10 errores comunes en la Gestión de Proyectos https://inventtatte.com/10-errores-comunes-en-la-gestion-de-proyectos/

[3] Stone,D. C. Jarrett, M. Woodroffe, S.Minocha. User Interface Design and Evaluation. Morgan Kaufmann publishers, 2005

[4] Sketch https://www.sketchapp.com/

[5] Trabajo de Investigación de Administración de Proyectos con tema en Metodología en Cascada realizado en la Universidad Estatal de Milagro por Gisella Serrano Silva http://es.scribd.com/doc/35015019/Metodologia-en-Cascada

[6] Página enfocada a Proyectos Ágiles, proyectosagiles.org https://proyectosagiles.org/que-es-scrum/

[7] Control predictivo vs empírico https://proyectosagiles.org/control-predictivo-control-empirico/

[8] Metodología Kanban https://iswugkanbansite.wordpress.com/inicio/roles/

[9] *Kanban vs Scrum boards: 11 major differences* de *RealtimeBoard* https://miro.com/blog/scrum-kanban-boards-differences/

[10] Corey Ladas. Essays on Kanban System for Lean Software Development, 2009

[11] Artículo de Atomic Design de Brad Frost http://bradfrost.com/blog/post/atomic-web-design/

[12] MockFlow https://mockflow.com/

[13] Toggl https://toggl.com/